VB6.0 实用类模块和模块汇总（简单测试版）

作者：张宏

完成时间：2022 年 9 月 4 日

前言

一些 VB6.0 常用并可以通用的类模块和模块，时间跨度较大有近十年时间，所以程序格式优劣都有，业余程序员程序。大多数类模块和模块都是经常使用的，没有使用和简单测试的或者错误废弃的都有标注。算是一次简单的汇总，印刷防止内容的永远消失。

目录

VB6.0 实用类模块和模块汇总（简单测试版）

1. Array1D

详细说明：

函数或方法名称	作用
Array1DSum	一维数组求和
Array1DCDataTypeDtoL	一维数组 double 转 long
Array1DMax	一维数组最大值
Array1DMix	一维数组最小值
Array1DRemoveOneNum	一维数组中剔除一个数据
Array1DDifferentNum	一维数组得到数据中无重的数据
SubDifferentNumbe	Array1DDifferentNum 子程序
SubSameGroupsNumber	Array1DDifferentNum 子程序
Array1DSort	一维数组进行大小排序
SubSort	Array1DSort 子程序
Array1DSortWhich	一维数组进行大小排序同时返回编号
SubSortWhichPart1	Array1DSortWhich 子程序
SubSortWhichPart2	Array1DSortWhich 子程序
Array1DDifferentElementNum	一维数组返回无重复元素的数据个数
Array1DMaxWhich	一维数组最大值及其位置
Array1DMixWhich	一维数组最小值及其位置
Array1DMixWhichExcludeEmptyAnd0	一维数组中最小值去除空的和 0
Array1DNotSameLetters	一维数组中相同字的个数
SubGetNotSame	Array1DNotSameLetters 子程序
SubUsedOrNot	Array1DNotSameLetters 子程序
Array1DSameLetters	一维数组中相同字的个数
Array1DSameLettersWhichPosition	一维数组相同字母组合位置反馈
Array1DSameNumbers	一维数组中相同数字的个数

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
```

```
'*****************************************************************************
'名称 = Array1D.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 12 月 30 日
'简介 = 操作一维数组
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'添加 Array1DSum                                                    2019-5-5
'修改程序结构                                                       2020-2-26
'修改程序注解                                                       2020-3-10
'综合各个版本                                                       2021-12-30
'*****************************************************************************
```

'Array1DSum 一维数组求和

Rem 求和

```
Public Function Array1DSum(ByRef Data() As Double) As Double
Dim i As Long
   For i = LBound(Data) To UBound(Data)
   Array1DSum = Array1DSum + Data(i)
   Next i
End Function
```

'Array1DCDataTypeDtoL 一维数组 double 转 long

Rem 双精度转长整型

```
Public Function Array1DCDataTypeDtoL(ByRef DblData() As Double, _
ByRef ReturnLngData() As Long)
Dim i As Long
ReDim ReturnLngData(LBound(DblData) To UBound(DblData))
For i = LBound(DblData) To UBound(DblData)
ReturnLngData(i) = CLng(DblData(i))
Next i
End Function
```

'Array1DMax 一维数组最大值

Rem 找到最大值

```
Public Function Array1DMax(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
cData = Data
Rem  防止数组只有一个数字
If LBound(Data) = UBound(Data) Then
Array1DMax = Data(LBound(Data))
End If
Rem  找到数组的最大值赋给 Array1DMax
For i = LBound(cData()) To UBound(cData())
    If i < UBound(cData()) Then
        If cData(i) > cData(i + 1) Then
        Array1DMax = cData(i)
        cData(i + 1) = cData(i)
        Else
        Array1DMax = cData(i + 1)
        End If
    End If
Next i
End Function
```

'Array1DMix 一维数组最小值

Rem 最小值

```
Public Function Array1DMix(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
cData = Data
Rem 防止只有一个数字================================
If LBound(Data) = UBound(Data) Then
Array1DMix = Data(LBound(Data))
End If
Rem  找到数组的最小值赋给 Array1DMix====================
For i = LBound(cData()) To UBound(cData())
    If i < UBound(cData()) Then
        If cData(i) < cData(i + 1) Then
        Array1DMix = cData(i)
        cData(i + 1) = cData(i)
        Else
```

```
            Array1DMix = cData(i + 1)
            End If
        End If
Next i
End Function
```

'Array1DRemoveOneNum 一维数组中剔除一个数据

Rem 从一维数组中剔除一个数据
Rem 例如：data(-1)=1,data(0)=2,data(1)=3,data(2)=4,data(3)=5 去除 1 则是 1,2,4,5。这里的 1 是指 data(1)中的 1
Rem Data()=原始数据
Rem RemoveWhich=数组标号，数据中剔除的数据
Rem ReturnData()=返回剔除后的数据，返回数组以 1 为底
Rem Array1DRemoveOneNum=返回剔除数据

```
Public Function Array1DRemoveOneNum(ByRef Data() As Double, _
ByVal RemoveWhich As Long, ByRef ReturnData() As Double) As Double
Dim i As Long
Dim cData() As Double
Dim n As Long
If (RemoveWhich < LBound(Data) Or RemoveWhich > UBound(Data)) Then
MsgBox "去除位置超出数组范围，程序跳出 Exit Function Array1DRemoveOneNum"
Exit Function
End If
If LBound(Data) = UBound(Data) Then
MsgBox "数组只有一个空间，程序跳出 Exit Function Array1DRemoveOneNum"
Exit Function
End If

cData = Data
n = 0
For i = LBound(Data) To UBound(Data)
    If i = RemoveWhich Then
    Array1DRemoveOneNum = Data(i)
    Else
    n = n + 1
    ReDim Preserve ReturnData(n)
    ReturnData(n) = cData(i)
    'Debug.Print "r(" & n & ")=" & "cdata(" & i & ")=" & cData(i)
    End If
Next i
End Function
```

'Array1DDifferentNum 一维数组得到数据中无重的数据

Rem 得到数据中无重的数据
Rem 例如：110 和 110 是重复数据，120 和 130 不是重复数据，111 和 112 不是重复数据
Rem Data()=原始数据
Rem ReturnDifferent()=返回数据中没有重复的数据
Rem ReturnSameGroupNum()=返回数据中相同数的个数 1.1，1.1，1.1，2，3，3 返回 3，1，2
Rem Array1DDifferentNum=返回无重复元素的数据个数

```
Public Function Array1DDifferentNum(ByRef Data() As Double, _
ByRef ReturnDifferent() As Double, ByRef ReturnSameGroupNum() As Long) As Integer
Dim m As Long, n As Long
Dim Same As Boolean
Dim SameGroupNum    As Long
Dim Count() As Boolean
Dim L As Long, i As Long
Dim CountN() As Long
Same = False
For L = LBound(Data()) To UBound(Data())
            For i = L To UBound(Data())
                Rem 自动跳转================================
                If i = UBound(Data()) Then
                    If Same = True Then
                    Same = False
                    End If
                  i = L + 1
                  L = L + 1
                  If L = UBound(Data()) Then
                  GoTo a2
                  End If
               End If
                '=======================================
                  If i + 1 <= UBound(Data()) Then
                  ReDim Preserve Count(LBound(Data()) To UBound(Data()))
                  'Debug.Print "data(" & L; ")= " & Data(L) & "      data(" & i + 1; ")=    " & Data(i + 1)
& "     Count(" & L & ")= " & Count(L)
                  ReDim Preserve CountN(LBound(Data()) To UBound(Data()))
                      If Data(L) = Data(i + 1) And Count(L) = False Then
                      Same = True
                      Count(i + 1) = True
                      CountN(L) = CountN(L) + 1 '统计相同数的个数
                      End If
                 End If
```

```
            Next i
Next L
a2:
Array1DDifferentNum = SubDifferentNumber(Data, Count, ReturnDifferent)
SubSameGroupsNumber Data, Count, CountN, Array1DDifferentNum, ReturnSameGroupNum
End Function
```

Rem Array1DDifferentNum 子程序
Rem 显示不同的数
Rem Data()=传入数据
Rem Count()=传入记录数据是否相同 False=不相同
Rem ReturnDifferent()=返回由不同数据构成的数组

```
Private Function SubDifferentNumber(ByRef Data() As Double, ByRef Count() As Boolean, _
ByRef ReturnDifferent() As Double) As Long
Dim i As Long
For i = LBound(Data()) To UBound(Data())
    If LBound(Data()) <> UBound(Data()) Then '去除只有一个数的情况
        If Count(i) = False Then '不同数据
        SubDifferentNumber = SubDifferentNumber + 1
        ReDim Preserve ReturnDifferent(SubDifferentNumber)
        ReturnDifferent(SubDifferentNumber) = Data(i)
        End If
    Else '单个数的情况
    SubDifferentNumber = 1
    ReDim Preserve ReturnDifferent(1)
    ReturnDifferent(SubDifferentNumber) = Data(i)
    End If
Next i
End Function
```

Rem Array1DDifferentNum 子程序
Rem 统计相同数的个数
Rem Data()=传入数据
Rem Count()=传入记录数据是否相同 False=不相同
Rem CountN()=传入统计相同数的个数
Rem Array1DDifferentNum=传入无重复元素的数据个数
Rem ReturnSameGroupNum()=返回统计相同数的个数

```
Private Function SubSameGroupsNumber(ByRef Data() As Double, ByRef Count() As Boolean, _
ByRef CountN() As Long, ByVal Array1DDifferentNum As Long, _
ByRef ReturnSameGroupNum() As Long)
Dim o As Long: Dim i As Long
ReDim ReturnSameGroupNum(1 To Array1DDifferentNum)
For i = LBound(Data()) To UBound(Data())
    If LBound(Data()) <> UBound(Data()) Then
        If Count(i) = False Then
        o = o + 1
```

```
            ReturnSameGroupNum(o) = CountN(i) + 1
            End If
        Else
        ReturnSameGroupNum(1) = 0
        End If
Next i
End Function
```

'Array1DSort 一维数组进行大小排序

Rem 对数据进行大小排列
Rem Data()=原始数据
Rem ReturnSortLtoU()=返回数据从小到大排列
Rem ReturnSortUtoL()=返回数据从大到小排列

```
Public Function Array1DSort(ByRef Data() As Double, ByRef ReturnSortLtoU() As Double, _
ByRef ReturnSortUtoL() As Double) As Boolean
Dim i As Long, L As Long, T As Double
Dim cData()    As Double
cData = Data
For L = LBound(cData) To UBound(cData)
        For i = LBound(cData) To (UBound(cData) - L)
            If i = UBound(cData) Then
            i = LBound(cData)
            L = L + 1
                If L = UBound(cData) Then
                Exit For
                End If
            End If
            If cData(i) > cData(i + 1) Then
            T = cData(i)
            cData(i) = cData(i + 1)
            cData(i + 1) = T
            End If
        'Debug.Print "cData(" & i & ")= " & cData(i) & "    *****      " & l
        Next i
Next L
SubSort cData, ReturnSortLtoU, ReturnSortUtoL
End Function
```

Rem Array1DSort 子程序
Rem 给出排序数组
Rem cData()=传入数据
Rem ReturnSortLtoU()=返回数据从小到大排列
Rem ReturnSortUtoL()=返回数据从大到小排列

```
Private Function SubSort(ByRef cData() As Double, ByRef ReturnSortLtoU() As Double, _
ByRef ReturnSortUtoL() As Double)
ReDim ReturnSortLtoU(LBound(cData) To UBound(cData))
ReDim ReturnSortUtoL(LBound(cData) To UBound(cData))
Dim i As Long, L As Long
For i = LBound(cData) To UBound(cData)
    'Debug.Print "cData(" & i & ")= " & cData(i) & "    PaiXu up"
    ReturnSortLtoU(i) = cData(i)
Next i
L = LBound(cData) - 1
For i = UBound(cData) To LBound(cData) Step -1
    'Debug.Print "cData(" & i & ")= " & cData(i) & "    PaiXu down"
    L = L + 1
    ReturnSortUtoL(L) = cData(i)
Next i
End Function
```

'Array1DSortWhich 一维数组进行大小排序同时返回编号

Rem 一维数组排序
Rem Data()=传入数据
Rem ReturnSortLtoU()=返回由小到大的数据
Rem ReturnSortLtoUWhich()=返回由小到大的数据的编号
Rem ReturnSortUtoL()=返回由大到小的数据
Rem ReturnSortUtoLWhich()=返回由大到小的数据的编号

```
Public Function Array1DSortWhich(ByRef Data() As Double, ByRef ReturnSortLtoU() As Double, _
ByRef ReturnSortLtoUWhich() As Long, ByRef ReturnSortUtoL() As Double, _
ByRef ReturnSortUtoLWhich() As Long) As Boolean
Dim i As Long, L As Long, T As Double
Dim OldData() As Double, cData() As Double
OldData = Data
cData = Data
For L = LBound(cData) To UBound(cData)
        For i = LBound(cData) To (UBound(cData) - L)
            If i = UBound(cData) Then
            i = LBound(cData)
            L = L + 1
                If L = UBound(cData) Then
                Exit For
                End If
            End If
            If cData(i) > cData(i + 1) Then
            T = cData(i)
```

```
                cData(i) = cData(i + 1)
                cData(i + 1) = T
                End If
            'Debug.Print "cData(" & i & ")= " & cData(i) & "    *****      " & l
        Next i
Next L
SubSortWhichPart1 cData, OldData, ReturnSortLtoU, ReturnSortLtoUWhich
SubSortWhichPart2 cData, OldData, ReturnSortLtoUWhich, ReturnSortUtoL,
ReturnSortUtoLWhich
End Function
```

Rem Array1DSortWhich 子程序

Rem 排序 1 部分

Rem Data()=传入由小到大的数据

Rem ReturnSortLtoU()=返回由小到大的数据

Rem ReturnSortLtoUWhich()=返回由小到大的数据的编号

Rem ReturnSortUtoL()=返回由大到小的数据

Rem ReturnSortUtoLWhich()=返回由大到小的数据的编号

```
Private Function SubSortWhichPart1(ByRef Data() As Double, ByRef OldData() As Double, _
ByRef ReturnSortLtoU() As Double, ByRef ReturnSortLtoUWhich() As Long)
Dim i As Long, L As Long, n As Long, m As Long, o As Long
Dim S6Obj As New S6_Programrunningtime
Dim BeginT As Long, EndT As Long
S6Obj.RunningTime BeginT
ReturnSortLtoU = Data
n = 0
ReDim ReturnSortLtoUWhich(LBound(Data) To UBound(Data))
FrmNotice.TxtNotice.Text = FrmNotice.TxtNotice.Text + "SubSortWhichPart1:" +
"UBound(OldData)=" + CStr(UBound(OldData)) & vbCrLf
For i = LBound(OldData) To UBound(OldData)
        Debug.Print i
        For L = LBound(OldData) To UBound(OldData)
        If ReturnSortLtoU(i) = OldData(L) Then
        n = n + 1
        If n > UBound(OldData) Then
        Exit Function
        S6Obj.RunningTime EndT
        FrmNotice.TxtNotice.Text = FrmNotice.TxtNotice.Text + CStr(UBound(OldData)) & " 跳
出时间：" & CStr(EndT - BeginT) & vbCrLf
        End If
        ReturnSortLtoUWhich(n) = L
                Rem 剔除重复的============================
                For m = LBound(OldData) To UBound(OldData)
                        For o = LBound(OldData) To UBound(OldData)
                                If ReturnSortLtoUWhich(m) = ReturnSortLtoUWhich(o) And m <> o
```

```
And ReturnSortLtoUWhich(m) <> 0 Then
                                    n = n - 1
                                    L = L + 1
                                    GoTo a1
                                    End If
                                    'Debug.Print UBound(OldData)
                                Next o
                            Next m
a1:
            'Debug.Print "ReturnSortLtoUWhich(" & i & ")= " & l
            End If
            Next L
Next i
S6Obj.RunningTime EndT
FrmNotice.TxtNotice.Text = FrmNotice.TxtNotice.Text + CStr(UBound(OldData)) & " 时间：" &
CStr(EndT - BeginT) & vbCrLf
End Function
```

Rem Array1DSortWhich 子程序
Rem 排序 2 部分
Rem Data()=传入改变后的数据
Rem OldData()=传入原始数据
Rem ReturnSortLtoU()=返回由小到大的数据
Rem ReturnSortLtoUWhich()=返回由小到大的数据的编号
Rem ReturnSortUtoL()=返回由大到小的数据
Rem ReturnSortUtoLWhich()=返回由大到小的数据的编号

```
Private Function SubSortWhichPart2(ByRef Data() As Double, ByRef OldData() As Double, _
ByRef ReturnSortLtoUWhich() As Long, ByRef ReturnSortUtoL() As Double, _
ByRef ReturnSortUtoLWhich() As Long)
ReDim ReturnSortUtoL(LBound(Data) To UBound(Data))
Dim i As Long, L As Long: Dim A() As Long
L = LBound(Data) - 1
For i = UBound(Data) To LBound(Data) Step -1
    'Debug.Print "Data(" & i & ")= " & Data(i) & "    PaiXu down"
    L = L + 1
    ReturnSortUtoL(L) = Data(i)
Next i
L = 0
ReDim ReturnSortUtoLWhich(LBound(Data) To UBound(Data))
A = ReturnSortLtoUWhich
For L = UBound(OldData) To LBound(OldData) Step -1
    i = i + 1
    ReturnSortUtoLWhich(i) = A(L)
Next L
End Function
```

'Array1DDifferentElementNum 一维数组返回无重复元素的数据个数

Rem 得到数据中无重复元素的数据
Rem 例如：**110** 就有数字 **1** 为重复元素，**120** 就没有重复元素，**133** 就有数字 **3** 为重复元素
Rem Data()原始数据
Rem Return_Different()数据中没有重复元素的
Rem Return_Same()数据中有重复元素的
Rem Array1DDifferentElementNum 返回无重复元素的数据个数

```
Public Function Array1DDifferentElementNum(ByRef Data() As Double, ByRef Return_Different()
As Double, _
ByRef Return_Same() As Double) As Integer
Dim i As Long, L As Long
Dim Num() As String
Dim DblNum() As Double
Dim rd() As Double
Dim r() As Long
Dim n As Long, n1 As Long
Dim DNum As Long
For i = LBound(Data) To UBound(Data)
        For L = 1 To Len(CStr(Data(i)))
        ReDim Preserve Num(L) As String
        ReDim Preserve DblNum(L) As Double
        Num(L) = Mid(Data(i), L, 1)
        DblNum(L) = CDbl(Num(L))
        Next L
DNum = Array1DDifferentNum(DblNum(), rd(), r())
        If DNum = Len(CStr(Data(i))) Then
        n = n + 1
        ReDim Preserve Return_Different(n) As Double
        Return_Different(n) = Data(i)
        Array1DDifferentElementNum = n
        Else
        n1 = n1 + 1
        ReDim Preserve Return_Same(n1) As Double
        Return_Same(n1) = Data(i)
        End If
Next i
End Function
```

'Array1DMaxWhich 一维数组最大值及其位置

Rem 得到数据中最大值
Rem 例如：110，12，130，1111,1111 最大值为 1111
Rem Data()原始数据
Rem 返回 Return_Which() 最大值的位置.例如：110，12，130，1111，1111 返回 4，5
Rem Array1DMaxWhich 返回最大值数值

```
Public Function Array1DMaxWhich(ByRef Data() As Double, ByRef Return_Which() As Long) As
Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next

Rem 16-8-29 防止只有一个数字
If LBound(Data) = UBound(Data) Then
Array1DMaxWhich = Data(LBound(Data))
ReDim Preserve Return_Which(1) As Long
Return_Which(1) = LBound(Data)
Exit Function
End If

For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) > cData(i + 1) Then
Array1DMaxWhich = cData(i)
cData(i + 1) = cData(i)
Else
Array1DMaxWhich = cData(i + 1)
End If
End If
Next
'************************************************
'遍历数组求得最大值的位置
Dim n As Long
n = 0
For i = LBound(Data()) To UBound(Data())
If (Array1DMaxWhich = Data(i)) Then
n = n + 1
ReDim Preserve Return_Which(n) As Long
Return_Which(n) = i
```

```
End If
Next i
End Function
```

'Array1DMixWhich 一维数组最小值及其位置

Rem 得到数据中最小值
Rem 例如：110，12，130，1111,1111 最小值为 12
Rem Data()原始数据
Rem Array1DMaxWhich 返回最大值数值
Rem 返回 Return_Which() 最大值的位置.例如：110，12，130，1111，1111 返回 2
Rem Return_Which() 返回最小值序数，按照给出数组的 lbound 开始计算，返回值是数组，原因是有可能有 2 个相同的最小值

```
Public Function Array1DMixWhich(ByRef Data() As Double, ByRef Return_Which() As Long) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(Data()) To UBound(Data())
cData(i) = Data(i)
Next i
Rem 16-8-29 防止只有一个数字
If LBound(Data) = UBound(Data) Then
    Array1DMixWhich = Data(LBound(cData))
    ReDim Preserve Return_Which(1) As Long
    Return_Which(1) = LBound(Data)
    Exit Function
End If
For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then ' 这条程序是什么意思 可能是防止超出范围，但是现在出现数组只有一个数的情况会发生冲突。
    If cData(i) < cData(i + 1) Then
    Array1DMixWhich = cData(i)
    cData(i + 1) = cData(i)
    Else
    Array1DMixWhich = cData(i + 1)
    End If
Else
End If
Next
'*********************************************
'遍历数组求得最小值的位置
Dim n As Long
```

```
n = 0
For i = LBound(Data()) To UBound(Data())
    If (Array1DMixWhich = Data(i)) Then
    n = n + 1
    ReDim Preserve Return_Which(n) As Long
    Return_Which(n) = i
    End If
Next i
n = 0
End Function
```

'Array1DMixWhichExcludeEmptyAnd0 一维数组中最小值去除空的和 0

Rem 得到数据中最小值去除空的和 0
Rem 例如：110，12，130，1111,1111 最小值为 12
Rem Data()原始数据
Rem Array1DMaxWhich 返回最大值数值
Rem 返回 Return_Which() 最大值的位置.例如：110，12，130，1111，1111 返回 2
Rem ReturnWhich() 返回最小值序数，按照给出数组的 lbound 开始计算，返回值是数组，原因是有可能有 2 个相同的最小值
Rem 未测试 2019-4-10

```
Public Function Array1DMixWhichExcludeEmptyAnd0(ByRef Data() As Double, ByRef
ReturnWhich() As Long) As Double
Dim i As Long, L As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(Data()) To UBound(Data())
cData(i) = Data(i)
Next i
For i = LBound(Data()) To UBound(Data())
    If cData(i) = 0 Then
    L = L + 1
    End If
Next i
If L = UBound(Data()) - LBound(Data()) + 1 Then
GoTo a1:
End If
Rem 16-8-29 防止只有一个数字
If LBound(Data) = UBound(Data) Then
Array1DMixWhichExcludeEmptyAnd0 = Data(LBound(cData))
ReDim Preserve ReturnWhich(1) As Long
```

```
ReturnWhich(1) = LBound(Data)
Exit Function
End If

For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then ' 这条程序是什么意思 可能是防止超出范围，但是现在出现数组只有一个数的情况会发生冲突。
If cData(i) < cData(i + 1) And cData(i) <> 0 Then
Array1DMixWhichExcludeEmptyAnd0 = cData(i)
cData(i + 1) = cData(i)
Else
If cData(i + 1) <> 0 Then
Array1DMixWhichExcludeEmptyAnd0 = cData(i + 1)
End If
End If
Else
End If
Next
'************************************************
'遍历数组求得最小值的位置
Dim n As Long
n = 0
For i = LBound(Data()) To UBound(Data())
If (Array1DMixWhichExcludeEmptyAnd0 = Data(i)) Then
n = n + 1
ReDim Preserve ReturnWhich(n) As Long
ReturnWhich(n) = i
Debug.Print i
End If
Next i
n = 0
Exit Function
a1:
ReDim Preserve ReturnWhich(1)
ReturnWhich(1) = 0
MsgBox "全部为 0"
End Function
```

'Array1DNotSameLetters 一维数组中相同字的个数

Rem 返回 1 维数组中相同字的个数
Rem 例子 Data(1) = "A"Data(2) = "AB"Data(3) = "AC"Data(4) = "AB"Data(5) = "AB"
Rem Data(6) = "AC"Data(7) = "A"Data(8) = "AAA"Data(9) = "AAA"Data(10) = "A"

Rem NotSame()=不同的字母,为 0 的是不同的，1 是出现过的

```
Public Function Array1DNotSameLetters(ByRef Data() As String, ByRef NotSame() As Long)
Dim i As Long, L As Long, P As Long, n As Long
Dim Used As Boolean '字母对比对比过的字母标记为 true
Dim ComparisonRedimed As Boolean '对比数组是否含有数据
Dim Count As Long '记录有多少个已经对比的字符
Dim ReturnWhichNumber() As Long '返回 1 维数组中相同字的个数-1
Dim Comparison() As Long   '记录对比过的字母，不再参与对比
For i = LBound(Data) To UBound(Data)
P = P + 1
    For L = LBound(Data) To UBound(Data)
        If i <> L And i < L Then
        SubUsedOrNot i, ComparisonRedimed, Comparison, Used
                If Used = False Then
                        If Data(i) = Data(L) Then
                            n = n + 1
                            Count = Count + 1
                            ReDim Preserve ReturnWhichNumber(P)
                            ReDim Preserve Comparison(Count)
                            ComparisonRedimed = True
                            ReturnWhichNumber(P) = n
                            Comparison(Count) = L   '记录对比过的字母，不再参与对比
                        Else
                            ReDim Preserve ReturnWhichNumber(P)
                        End If
                End If
        End If
    Next L
n = 0
Used = False
Next i
SubGetNotSame Data, Comparison, NotSame
End Function
```

Rem Array1DNotSameLetters 的子程序

Rem 把数据转换成想要的数据

```
Private Sub SubGetNotSame(Data, ByRef Comparison() As Long, NotSame)
Dim i As Long, L As Long
ReDim NotSame(LBound(Data) To UBound(Data))
Dim Obj1 As New ArrayEmptyOrNot
If Obj1.ArrayEmpty1(Comparison) = False Then
    For i = LBound(Data) To UBound(Data)
        For L = LBound(Comparison) To UBound(Comparison)
            If i <> Comparison(L) Then
            NotSame(i) = 0
```

```
                Else
                NotSame(i) = 1
                Exit For
                End If
            Next L
        Next i
End If
End Sub
```

Rem　Array1DNotSameLetters 的子程序

Rem 判断字符是否已经对比过

```
Private Sub SubUsedOrNot(i, ComparisonRedimed, Comparison, ReturnUsed)
Dim q As Variant '用在 For Each 中
        If ComparisonRedimed = True Then
            For Each q In Comparison
                    If i = q Then
                    ReturnUsed = True '已经对比过
                    Exit For '加快程序
                    End If
            Next
        End If
End Sub
```

'Array1DSameLetters 一维数组中相同字的个数

Rem Array1DSameLetters 返回 1 维数组中相同字的个数

Rem ReturnWhichNumber(a)返回 1 维数组中相同字的个数，(a)是字位置,注意区分大小写

```
Public Function Array1DSameLetters(ByRef Data() As String, ByRef ReturnWhichNumber() As Long)
As Double
Dim i As Long, L As Long, n As Long, P As Long
For i = LBound(Data) To UBound(Data)
P = P + 1
    For L = LBound(Data) To UBound(Data)
        If i <> L Then
            If Data(i) = Data(L) Then
            n = n + 1
            ReDim Preserve ReturnWhichNumber(P)
            ReturnWhichNumber(P) = n
            Else
            ReDim Preserve ReturnWhichNumber(P)
            End If
        End If
    Next L
n = 0
```

```
Next i
'Array1DSameNumbers = n
End Function
```

'Array1DSameLettersWhichPosition 一维数组相同字母组合位置反馈

Rem 缺乏测试
Rem 相同字母组合位置反馈
Rem 例如：a,b,c,d,ab,bc,ac,ac,ab,ab
Rem ab:ReturnSameLetter(1,1)=5 ReturnSameLetter(1,2)=9 ReturnSameLetter(1,3)=10
Rem ac:ReturnSameLetter(2,1)=7 ReturnSameLetter(2,2)=8

```
Public Function Array1DSameLettersWhichPosition(ByRef Data() As String, ByRef
ReturnSameLetter() As Long)
Rem 用二维数组记录所有的重复字母的位置
ReDim ReturnSameLetter(1 To UBound(Data), 1 To UBound(Data))
Dim n As Long, m As Long, P As Long, i As Long, L As Long
n = 0
P = 0
m = 0
For i = 1 To UBound(Data)
    For L = 1 To UBound(Data)
            If i <> L And i < L Then
            If Data(i) = Data(L) Then
                If m <> i Then
                n = n + 1
                P = P + 1
                ReturnSameLetter(P, n) = i
                m = i
                End If
                n = n + 1
                ReturnSameLetter(P, n) = L
            End If
            End If
    Next L
    m = 0
    n = 0
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 去除多余的重复 列如[5,9,10][9,10]去除[9,10]
Dim ii As Long, ll As Long
For i = 1 To UBound(Data)
```

```
    For L = 1 To UBound(Data)
        For ii = 1 To UBound(Data)
            For ll = 1 To UBound(Data)
                If i <> ii Then
                If ReturnSameLetter(i, L) = ReturnSameLetter(ii, ll) And ReturnSameLetter(i, L)
<> 0 And ReturnSameLetter(ii, ll) <> 0 Then
                        If L > ll Then
                        ReturnSameLetter(ii, ll) = 0
                        End If
                End If
                End If
            Next ll
        Next ii
    Next L
Next i
End Function
```

'Array1DSameNumbers 一维数组相同数字的个数

Rem Array1DSameNumbers 返回 1 维数组中相同数字的个数
Rem ReturnWhichNumber(a)返回 1 维数组中相同数字的个数，(a)是数字位置

```
Public Function Array1DSameNumbers(ByRef Data() As Double, ByRef ReturnWhichNumber() As
Long) As Double
Dim i As Long, L As Long, n As Long, P As Long
For i = LBound(Data) To UBound(Data)
P = P + 1
For L = LBound(Data) To UBound(Data)
If i <> L Then
If Data(i) = Data(L) Then
n = n + 1
ReDim Preserve ReturnWhichNumber(P)
ReturnWhichNumber(P) = n
Else
ReDim Preserve ReturnWhichNumber(P)
End If
End If
Next L
n = 0
Next i
'Array1DSameNumbers = n
End Function
```

2. Array2D

详细说明：

函数或方法名称	作用
Array2DMix	二维数组最小值
Array1DMix	Array2DMix 子程序
Array2DMax	二维数组最大值
Array1DMax	Array2DMax 子程序
Array2DDifferentNumV	二维数组得到数据中无重的数据
Array2DDifferentNumIncludeEmpty	二维数据中无重的数据包括空白
Array2DSortIncludeEmpty	二维数组按大小排列数据包括空白
Array2DSum	二维数组求和
Array2DCDataTypeVtoD	二维数组数据转换
Array2DCDataTypeDtoV	二维数组数据转换
Array2DCDataTypeStoDEmptyTo0	二维数组数据转换空格变 0
Array2DRemoveEmpty	二维数组数据去除空白
Array2DRedimWidth	二维数组去掉多余的 0 项，保留最大非 0 边界
SubArray1DMax	Array2DRedimWidth 子程序

代码：

```
Option Explicit
Option Base 1
'*********************************************************************************
'名称 = Array2D.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 12 月 31 日
'简介 = 操作二维数组
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'最值是错误的,16-6-29 已经更改过，需要全方面测试
'Mix Max 没有排除特殊情况，像只有一个元素的数组                    16-8-29
'增加 Str 变成 Dbl
'增加 Array2DCDataTypeVtoS                                        19-4-10
'增加 Array2DRedimWidth 及其子程序 SubArray1DMax                   20-3-28
'修改 Array2DMix 让其不关联其他类模块                              20-8-13
'修改 Array2DMax 让其不关联其他类模块                              20-8-13
'修改 Array2DRedimWidth 让其不关联其他类模块                       20-8-13
'修改 SubArray1DMax                                               20-8-13
'*********************************************************************************
```

'Array2DMix 二维数组最小值

Rem 一个元素的没有测试，应该不能通过 16-8-29

Rem 最小值

```
Public Function Array2DMix(ByRef Data() As Double) As Double
Dim i As Long, l As Long
Dim c1Data() As Double
Dim n As Long
n = 0
ReDim c1Data(LBound(Data, 1) To UBound(Data, 1), LBound(Data, 2) To UBound(Data, 2))
For l = LBound(Data, 2) To UBound(Data, 2)
For i = LBound(Data, 1) To UBound(Data, 1)
c1Data(i, l) = Data(i, l)
'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "Data(" & i & "," & l & ")=" & Data(i, l)
Next i
Next l
Dim Part() As Double
For l = LBound(c1Data, 2) To UBound(c1Data, 2)
n = n + 1
ReDim Preserve Part(n) As Double
For i = LBound(c1Data, 1) To UBound(c1Data, 1)
If (l <= UBound(c1Data, 2)) Then
If (i < UBound(c1Data, 1)) Then
'Debug.Print "i=" & i & "l=" & l
'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "c1Data(" & i + 1 & "," & l & ")=" &
c1Data(i + 1, l)
If (c1Data(i, l) < c1Data(i + 1, l)) Then
Part(n) = c1Data(i, l)
c1Data(i + 1, l) = c1Data(i, l)
Else
Part(n) = c1Data(i + 1, l)
End If
End If
End If
Next i
Next l
n = 0
'If (Part(1) <= Part(2)) Then
  'Array2DMix = Part(1)
  'Else
  'Array2DMix = Part(2)
'End If
Array2DMix = Array1DMix(Part())
```

```
End Function
```

Rem Array2DMix 子程序

Rem 要用到的函数

```
Public Function Array1DMix(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(Data()) To UBound(Data())
cData(i) = Data(i)
Next
Rem 16-8-29 防止只有一个数字
If LBound(Data) = UBound(Data) Then
Array1DMix = Data(LBound(Data))
End If
For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) < cData(i + 1) Then
Array1DMix = cData(i)
cData(i + 1) = cData(i)
Else
Array1DMix = cData(i + 1)
End If
End If
Next
End Function
```

'Array2DMax 二维数组最大值

Rem 一个元素的没有测试，应该不能通过 16-8-29

Rem 求 2D 数据最大值

```
Public Function Array2DMax(ByRef Data() As Double) As Double
Dim i As Long, l As Long
Dim c1Data() As Double
Dim n As Long
n = 0
ReDim c1Data(LBound(Data, 1) To UBound(Data, 1), LBound(Data, 2) To UBound(Data, 2))
For l = LBound(Data, 2) To UBound(Data, 2)
For i = LBound(Data, 1) To UBound(Data, 1)
c1Data(i, l) = Data(i, l)
'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "Data(" & i & "," & l & ")=" & Data(i, l)
Next i
Next l
Dim Part() As Double
```

```
For l = LBound(c1Data, 2) To UBound(c1Data, 2)
n = n + 1
ReDim Preserve Part(n) As Double
For i = LBound(c1Data, 1) To UBound(c1Data, 1)
If (l <= UBound(c1Data, 2)) Then
If (i < UBound(c1Data, 1)) Then
'Debug.Print "i=" & i & "l=" & l
'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "c1Data(" & i + 1 & "," & l & ")=" &
c1Data(i + 1, l)
If (c1Data(i, l) > c1Data(i + 1, l)) Then
Part(n) = c1Data(i, l)
c1Data(i + 1, l) = c1Data(i, l)
Else
Part(n) = c1Data(i + 1, l)
End If
End If
End If
Next i
Next l
n = 0
Array2DMax = Array1DMax(Part())
End Function
```

Rem Array2DMax 子程序

Rem 要用到的函数

```
Public Function Array1DMax(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next
Rem 16-8-29 防止只有一个数字
If LBound(Data) = UBound(Data) Then
Array1DMax = Data(LBound(Data))
End If

For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) > cData(i + 1) Then
Array1DMax = cData(i)
cData(i + 1) = cData(i)
Else
Array1DMax = cData(i + 1)
End If
```

```
End If
Next
End Function
```

'Array2DDifferentNumV 二维数组得到数据中无重的数据

Rem 得到数据中无重的数据
Rem 例如：110 和 110 是重复数据，120 和 130 不是重复数据，111 和 112 不是重复数据
Rem Data()原始数据
Rem Return_Different()数据中没有重复的数据
Rem Array2DDifferentNumV 返回无重复元素的数据个数,变体数据

```
Public Function Array2DDifferentNumV(ByRef Data() As Variant, ByRef Return_Different() As
Variant) As Integer
    Dim m As Long, n As Long
    Dim Same As Boolean
    Dim SameGroupNum    As Long
    Dim Count() As Boolean
    Dim l As Long, i As Long, j As Long, k As Long, O As Long
    Dim Data1D() As Variant
    Same = False
    m = 0


For j = LBound(Data(), 1) To UBound(Data(), 1)
For k = LBound(Data(), 2) To UBound(Data(), 2)
If Data(j, k) <> "" Then
O = O + 1
ReDim Preserve Data1D(O)
Data1D(O) = Data(j, k)
'Debug.Print Data1D(O) & "    data1d"
End If
Next k
Next j

For l = LBound(Data1D()) To UBound(Data1D())
For i = l To UBound(Data1D())
    If i = UBound(Data1D()) Then
        '****************************************** 统计相同数的组数没有用上
        If Same = True Then                                    '*
        SameGroupNum = SameGroupNum + 1    '*
        Same = False                                                '*
          End If                                                        '*
        '******************************************
```

```
    i = l + 1
    l = l + 1

    'Debug.Print i
    If l = UBound(Data1D()) Then
    GoTo a2
    End If
    End If

  If i + 1 <= UBound(Data1D()) Then
  'Debug.Print "data(" & l; ")= " & data(l) & "      data(" & i + 1; ")=    " & data(i + 1) & "      Count("
& l & ")= " & Count(l)
  ReDim Preserve Count(LBound(Data1D()) To UBound(Data1D()))
  If Data1D(l) = Data1D(i + 1) And Count(l) = False Then
  Same = True
  Count(i + 1) = True
 End If
 End If
Next i

      '****************************************** 统计相同数的组数，没有用上
      If l = UBound(Data1D()) And Count(l) = False Then              '*
      SameGroupNum = SameGroupNum + 1    '*

     End If                                                                  '*
      '******************************************
Next l
a2:

For i = LBound(Data1D()) To UBound(Data1D())
If Count(i) = False Then
Array2DDifferentNumV = Array2DDifferentNumV + 1

ReDim Preserve Return_Different(Array2DDifferentNumV)
Return_Different(Array2DDifferentNumV) = Data1D(i)
End If
Next
For i = 1 To Array2DDifferentNumV
'Debug.Print Return_Different(i) & "Array2DDifferentNumV"
Next
'Debug.Print "Array2DDifferentNumV=" & Array2DDifferentNumV
End Function
```

'Array2DDifferentNumIncludeEmpty 二维数据中无重的数据包括空白

Rem 得到数据中无重的数据包括空白

```
Public Function Array2DDifferentNumIncludeEmpty(ByRef Data() As Variant, ByRef
Return_Different() As Variant) As Integer
    Dim m As Long, n As Long
    Dim Same As Boolean
    Dim SameGroupNum    As Long
    Dim Count() As Boolean
    Dim l As Long, i As Long, j As Long, k As Long, O As Long
    Dim Data1D() As Variant
    Same = False
    m = 0

For j = LBound(Data(), 1) To UBound(Data(), 1)
For k = LBound(Data(), 2) To UBound(Data(), 2)
O = O + 1
ReDim Preserve Data1D(O)
Data1D(O) = Data(j, k)
'Debug.Print Data1D(O) & "    data1d"
Next k
Next j

For l = LBound(Data1D()) To UBound(Data1D())
For i = l To UBound(Data1D())
    If i = UBound(Data1D()) Then
        '****************************************** 统计相同数的组数没有用上
        If Same = True Then                                    '*
        SameGroupNum = SameGroupNum + 1    '*
        Same = False                                                '*
          End If                                                        '*
        '******************************************
    i = l + 1
    l = l + 1

    'Debug.Print i
    If l = UBound(Data1D()) Then
    GoTo a2
    End If
    End If
```

```
   If i + 1 <= UBound(Data1D()) Then
   'Debug.Print "data(" & l; ")= " & data(l) & "      data(" & i + 1; ")=    " & data(i + 1) & "      Count("
& l & ")= " & Count(l)
   ReDim Preserve Count(LBound(Data1D()) To UBound(Data1D()))
   If Data1D(l) = Data1D(i + 1) And Count(l) = False Then
   Same = True
   Count(i + 1) = True
  End If
  End If
Next i

       '****************************************** 统计相同数的组数，没有用上
       If l = UBound(Data1D()) And Count(l) = False Then              '*
       SameGroupNum = SameGroupNum + 1    '*

      End If                                                                    '*
       '******************************************
Next l
a2:

For i = LBound(Data1D()) To UBound(Data1D())
If Count(i) = False Then
Array2DDifferentNumIncludeEmpty = Array2DDifferentNumIncludeEmpty + 1

ReDim Preserve Return_Different(Array2DDifferentNumIncludeEmpty)
Return_Different(Array2DDifferentNumIncludeEmpty) = Data1D(i)
End If
Next
For i = 1 To Array2DDifferentNumIncludeEmpty
'Debug.Print Return_Different(i) & "Array2DDifferentNum"
Next
'Debug.Print "Array2DDifferentNum=" & Array2DDifferentNum
End Function
```

'Array2DSortIncludeEmpty 二维数组按大小排列数据包括空白

Rem Data()原始数据
Rem Return_SortLtoU1D()返回的是由小到大重新排列的 **1** 维数组
Rem Return_SortUtoL1D()返回的是由大到小重新排列的 **1** 维数组
Rem Return_SortLtoU2D()返回的是由小到大重新排列的 **2** 维数组。维数内的数据进行排列。
Rem Return_SortUtoL2D()返回的是由大到小重新排列的 **2** 维数组。维数内的数据进行排列。

```
Public Function Array2DSortIncludeEmpty(ByRef Data() As Variant, ByRef Return_SortLtoU1D() As
Variant, ByRef Return_SortUtoL1D() As Variant, ByRef Return_SortLtoU2D() As Variant, ByRef
```

```
Return_SortUtoL2D() As Variant) As Boolean
Dim i As Long, l As Long, t As Variant
Dim j As Long, k As Long, m As Long
Dim DataC1D() As Variant
For k = LBound(Data, 1) To UBound(Data, 1)
For j = LBound(Data, 2) To UBound(Data, 2)
m = m + 1
ReDim Preserve DataC1D(m)
DataC1D(m) = Data(k, j)
Next j
Next k

For l = LBound(DataC1D) To UBound(DataC1D)
For i = LBound(DataC1D) To (UBound(DataC1D) - l) '冒泡排序这里不能用 i=L!!! 只能是
(UBound(Data) - l)
If i = UBound(DataC1D) Then
i = LBound(DataC1D)
l = l + 1
If l = UBound(DataC1D) Then
Exit For
End If
End If
If DataC1D(i) > DataC1D(i + 1) Then
t = DataC1D(i)
DataC1D(i) = DataC1D(i + 1)
DataC1D(i + 1) = t
End If
'Debug.Print "Data(" & i & ")= " & Data(i) & "    *****      " & l
Next
Next
ReDim Return_SortLtoU1D(LBound(DataC1D) To UBound(DataC1D))
ReDim Return_SortUtoL1D(LBound(DataC1D) To UBound(DataC1D))
For i = LBound(DataC1D) To UBound(DataC1D)
'Debug.Print "Data(" & i & ")= " & Data(i) & "    PaiXu up"
Return_SortLtoU1D(i) = DataC1D(i)
Next
l = LBound(DataC1D) - 1
For i = UBound(DataC1D) To LBound(DataC1D) Step -1
'Debug.Print "Data(" & i & ")= " & Data(i) & "    PaiXu down"
l = l + 1
Return_SortUtoL1D(l) = DataC1D(i)
Next i
m = 0
ReDim Return_SortLtoU2D(LBound(Data, 1) To UBound(Data, 1), LBound(Data, 2) To
```

```
UBound(Data, 2))
ReDim Return_SortUtoL2D(LBound(Data, 1) To UBound(Data, 1), LBound(Data, 2) To
UBound(Data, 2))
For k = LBound(Data, 1) To UBound(Data, 1)
For j = LBound(Data, 2) To UBound(Data, 2)
m = m + 1
Return_SortLtoU2D(k, j) = DataC1D(m)
Next j
Next k

m = 0
For k = UBound(Data, 1) To LBound(Data, 1) Step -1
For j = UBound(Data, 2) To LBound(Data, 2) Step -1
m = m + 1
Return_SortUtoL2D(k, j) = DataC1D(m)
Next j
Next k

End Function
```

'Array2DSum 二维数组求和

Rem 2D 数据求和

```
Public Function Array2DSum(ByRef Data() As Double) As Double
Dim k As Long, j As Long
For k = LBound(Data, 1) To UBound(Data, 1)
For j = LBound(Data, 2) To UBound(Data, 2)
Array2DSum = Array2DSum + Data(k, j)
Next j
Next k
End Function
```

'Array2DCDataTypeVtoD 二维数组数据转换

Rem 数据类型转换

```
Public Function Array2DCDataTypeVtoD(ByRef VarData() As Variant, ByRef Return_DblData() As
Double) As Boolean
Dim i As Long, l As Long
ReDim Return_DblData(LBound(VarData, 1) To UBound(VarData, 1), LBound(VarData, 2) To
UBound(VarData, 2))
For i = LBound(VarData, 1) To UBound(VarData, 1)
For l = LBound(VarData, 2) To UBound(VarData, 2)
```

```
  Return_DblData(i, l) = CDbl(VarData(i, l))
Next l
Next i
Array2DCDataTypeVtoD = True
End Function
```

'Array2DCDataTypeDtoV 二维数组数据转换

Rem 数据类型转换

```
Public Function Array2DCDataTypeDtoV(ByRef DblData() As Double, ByRef Return_VarData() As Variant) As Boolean
Dim i As Long, l As Long
ReDim Return_VarData(LBound(DblData, 1) To UBound(DblData, 1), LBound(DblData, 2) To UBound(DblData, 2))
For i = LBound(DblData, 1) To UBound(DblData, 1)
For l = LBound(DblData, 2) To UBound(DblData, 2)
Return_VarData(i, l) = CDbl(DblData(i, l))
Next l
Next i
Array2DCDataTypeDtoV = True
End Function
```

'Array2DCDataTypeVtoS 二维数组数据转换

Rem 数据类型转换

```
Public Function Array2DCDataTypeVtoS(ByRef VarData() As Variant, ByRef Return_StrData() As String) As Boolean
Dim i As Long, l As Long
ReDim Return_StrData(LBound(VarData, 1) To UBound(VarData, 1), LBound(VarData, 2) To UBound(VarData, 2))
For i = LBound(VarData, 1) To UBound(VarData, 1)
For l = LBound(VarData, 2) To UBound(VarData, 2)
Return_StrData(i, l) = CDbl(VarData(i, l))
Next l
Next i
Array2DCDataTypeVtoS = True
End Function
```

'Array2DCDataTypeStoD 二维数组数据转换

Rem 数据类型转换

```
Public Function Array2DCDataTypeStoD(ByRef StrData() As String, ByRef Return_DblData() As
Double) As Boolean
Dim i As Long, l As Long
ReDim Return_DblData(LBound(StrData, 1) To UBound(StrData, 1), LBound(StrData, 2) To
UBound(StrData, 2))
For i = LBound(StrData, 1) To UBound(StrData, 1)
For l = LBound(StrData, 2) To UBound(StrData, 2)
  Return_DblData(i, l) = CDbl(StrData(i, l))
  'Debug.Print "Return_DblData(" & i & "," & l & ")=" & Return_DblData(i, l)
Next l
Next i
Array2DCDataTypeStoD = True
End Function
```

'Array2DCDataTypeStoDEmptyTo0 二维数组数据转换空格变 0

Rem 数据类型转换

Rem string 为空的时候，转化为 0

```
Public Function Array2DCDataTypeStoDEmptyTo0(ByRef StrData() As String, ByRef
Return_DblData() As Double) As Boolean
Dim i As Long, l As Long
ReDim Return_DblData(LBound(StrData, 1) To UBound(StrData, 1), LBound(StrData, 2) To
UBound(StrData, 2))
For i = LBound(StrData, 1) To UBound(StrData, 1)
      For l = LBound(StrData, 2) To UBound(StrData, 2)
            If StrData(i, l) <> "" Then
            Return_DblData(i, l) = CDbl(StrData(i, l))
            Else
            Return_DblData(i, l) = 0
            End If
            'Debug.Print "Return_DblData(" & i & "," & l & ")=" & Return_DblData(i, l)
      Next l
Next i
Array2DCDataTypeStoDEmptyTo0 = True
End Function
```

'Array2DRemoveEmpty 二维数组数据去除空白

Rem 未测试

Rem 2 维数组把不全为 0 的行前移。例如：0,1,0 0,0,0 1,0,0 变成 0,1,0 1,0,0 0,0,0

Rem NoEmpty=True 那么 。例如：0,1,0 0,0,0 1,0,0 变成 0,1,0 1,0,0

```
Private Sub Array2DRemoveEmpty(ByRef Data2D(), ByVal NoEmpty As Boolean, ByRef
```

```
ReturnData2D())
Dim i As Long, l As Long, m As Long, n As Long
Dim Data2DOld()
Dim cData2D()
Data2DOld = Data2D
For i = 1 To UBound(Data2D, 1)
        For l = 1 To UBound(Data2D, 2)
            If Data2DOld(i, l) <> 0 Then
            n = n + 1
                If n = 1 Then
                m = m + 1
                End If
            ReturnData2D(m, n) = Data2DOld(i, l)
            End If
        Next l
        n = 0
Next i
If NoEmpty = True Then
    cData2D = ReturnData2D
    ReDim ReturnData2D(1 To m, 1 To UBound(Data2D, 2))
    For i = 1 To m
            For l = 1 To UBound(Data2D, 2)
                ReturnData2D(i, l) = cData2D(i, l)
            Next l
    Next i
End If
End Sub
```

'Array2DRedimWidth 二维数组去掉多余的 0 项，保留最大非 0 边界

Rem 去掉多余的 0 项，保留最大非 0 边界

Rem 未测试

```
Private Sub Array2DRedimWidth(ByRef Data2D() As Double)
Dim i As Long, l As Long
Dim Max() As Double
    For l = 1 To UBound(Data2D, 1)
        For i = 1 To UBound(Data2D, 2)
            If Data2D(l, i) = 0 Then
            ReDim Preserve Max(l)
            Max(l) = i
                If i = 1 Then '第一个为 0
```

```
                    '需要剔除这一条嘛？
                    End If
                Exit For
                End If
            Next i
        Next l
For i = 1 To UBound(Data2D, 1)
ReDim Data2D(i, SubArray1DMax(Max))
Next i
End Sub
```

Rem Array2DRedimWidth 子程序

```
Public Function SubArray1DMax(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next i
Rem 16-8-29 防止只有一个数字
If LBound(Data) = UBound(Data) Then
SubArray1DMax = Data(LBound(Data))
End If

For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) > cData(i + 1) Then
SubArray1DMax = cData(i)
cData(i + 1) = cData(i)
Else
SubArray1DMax = cData(i + 1)
End If
End If
Next
End Function
```

3. Array3D

详细说明：

函数或方法名称	作用
Array3DMax	三维数组最大值
Array3DCDataTypeBtoV	三维数组类型转换
Array3DMaxWhich	三维数组最大值及其位置
Array3DCDataTypeDtoL	三维数组类型转换
Array3DCDataTypeLtoD	三维数组类型转换

代码：

```
Option Explicit
Option Base 1
'********************************************************************************
'名称 = Array3D.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 12 月 31 日
'简介 = 操作三维数组
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
' Array3DMaxWhich 要用到 Array1D001
'********************************************************************************
```

'Array3DMax 三维数组最大值

Rem 简单测试通过

```
Public Function Array3DMax(ByRef Data() As Double) As Double
Dim i As Long, l As Long, m As Long
Dim c1Data() As Double
Dim n As Long
n = 0
ReDim c1Data(LBound(Data, 1) To UBound(Data, 1), LBound(Data, 2) To UBound(Data, 2),
LBound(Data, 3) To UBound(Data, 3))
For m = LBound(Data, 3) To UBound(Data, 3)
For l = LBound(Data, 2) To UBound(Data, 2)
For i = LBound(Data, 1) To UBound(Data, 1)
c1Data(i, l, m) = Data(i, l, m)
'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "Data(" & i & "," & l & ")=" & Data(i, l)
```

```
Next i
Next l
Next m
Dim Part() As Double

For m = LBound(c1Data, 3) To UBound(c1Data, 3)
For l = LBound(c1Data, 2) To UBound(c1Data, 2)
n = n + 1
ReDim Preserve Part(n) As Double
For i = LBound(c1Data, 1) To UBound(c1Data, 1)
If (l <= UBound(c1Data, 2)) Then
If (i < UBound(c1Data, 1)) Then
'Debug.Print "i=" & i & "l=" & l
'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "c1Data(" & i + 1 & "," & l & ")=" &
c1Data(i + 1, l)
If (c1Data(i, l, m) > c1Data(i + 1, l, m)) Then
Part(n) = c1Data(i, l, m)
c1Data(i + 1, l, m) = c1Data(i, l, m)
Else
Part(n) = c1Data(i + 1, l, m)
End If
End If
End If
Next i
Next l
Next m
n = 0
Dim obj As New Array1D001
Array3DMax = obj.Array1D_Max(Part())
Set obj = Nothing
End Function
```

'Array3DCDataTypeBtoV 三维数组类型转换

Rem 3 维数组类型改变

```
Public Function Array3DCDataTypeBtoV(ByRef BoolData() As Boolean, ByRef Return_VarData() As
Variant) As Boolean
Dim i As Long, l As Long, m As Long
ReDim Return_VarData(LBound(BoolData, 1) To UBound(BoolData, 1), LBound(BoolData, 2) To
UBound(BoolData, 2), LBound(BoolData, 3) To UBound(BoolData, 3))
For i = LBound(BoolData, 1) To UBound(BoolData, 1)
For l = LBound(BoolData, 2) To UBound(BoolData, 2)
For m = LBound(BoolData, 3) To UBound(BoolData, 3)
```

```
Return_VarData(i, l, m) = CVar(BoolData(i, l, m))
Next m
Next l
Next i
Array3DCDataTypeBtoV = True
End Function
```

'Array3DMaxWhich 三维数组最大值及其位置

Rem 简单测试通过（如果有两个相同最大值可能有问题）
Rem Data()=数据
Rem R_WhichBound1():R_WhichBound2():R_WhichBound3() 3 维度数据的 3 个限
Rem SameMaxNum=相同最大值个数

```
Public Function Array3DMaxWhich(ByRef Data() As Double, ByRef R_WhichBound1() As Long,
ByRef R_WhichBound2() As Long, ByRef R_WhichBound3() As Long, ByRef SameMaxNum As Long)
As Double
Dim i As Long, l As Long, m As Long
Dim c1Data() As Double
Dim n As Long
n = 0
ReDim c1Data(LBound(Data, 1) To UBound(Data, 1), LBound(Data, 2) To UBound(Data, 2),
LBound(Data, 3) To UBound(Data, 3))
For m = LBound(Data, 3) To UBound(Data, 3)
    For l = LBound(Data, 2) To UBound(Data, 2)
        For i = LBound(Data, 1) To UBound(Data, 1)
        c1Data(i, l, m) = Data(i, l, m)
        'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "Data(" & i & "," & l & ")=" &
Data(i, l)
        Next i
    Next l
Next m
Dim Part() As Double

For m = LBound(c1Data, 3) To UBound(c1Data, 3)
    For l = LBound(c1Data, 2) To UBound(c1Data, 2)
    n = n + 1
    ReDim Preserve Part(n) As Double
        For i = LBound(c1Data, 1) To UBound(c1Data, 1)
            If (l <= UBound(c1Data, 2)) Then
                If (i < UBound(c1Data, 1)) Then
                'Debug.Print "i=" & i & "l=" & l
                'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "c1Data(" & i + 1 &
"," & l & ")=" & c1Data(i + 1, l)
```

```
                        If (c1Data(i, l, m) > c1Data(i + 1, l, m)) Then
                        Part(n) = c1Data(i, l, m)
                        c1Data(i + 1, l, m) = c1Data(i, l, m)
                        Else
                        Part(n) = c1Data(i + 1, l, m)
                        End If
                    End If
                End If
            Next i
        Next l
Next m
n = 0
Dim obj As New Array1D001
Array3DMaxWhich = obj.Array1D_Max(Part())
For m = LBound(Data, 3) To UBound(Data, 3)
        For l = LBound(Data, 2) To UBound(Data, 2)
            For i = LBound(Data, 1) To UBound(Data, 1)
            If Array3DMaxWhich = Data(i, l, m) Then
            n = n + 1
            ReDim Preserve R_WhichBound1(n)
            ReDim Preserve R_WhichBound2(n)
            ReDim Preserve R_WhichBound3(n)
            R_WhichBound1(n) = i
            R_WhichBound2(n) = l
            R_WhichBound3(n) = m
            SameMaxNum = n
            End If
            Next i
        Next l
Next m

n = 0
'Debug.Print LBound(R_WhichBound1)
                        Set obj = Nothing

End Function
```

'Array3DCDataTypeDtoL 三维数组类型转换

Rem 类型转换

```
Public Function Array3DCDataTypeDtoL(ByRef DblData() As Double, ByRef Return_LngData() As Long) As Boolean
Dim i As Long, l As Long, m As Long
```

```
ReDim Return_LngData(LBound(DblData, 1) To UBound(DblData, 1), LBound(DblData, 2) To
UBound(DblData, 2), LBound(DblData, 3) To UBound(DblData, 3))
For i = LBound(DblData, 1) To UBound(DblData, 1)
For l = LBound(DblData, 2) To UBound(DblData, 2)
For m = LBound(DblData, 3) To UBound(DblData, 3)
Return_LngData(i, l, m) = CLng(DblData(i, l, m))
Next m
Next l
Next i
Array3DCDataTypeDtoL = True
End Function
```

'Array3DCDataTypeLtoD 三维数组类型转换

Rem 类型转换

```
Public Function Array3DCDataTypeLtoD(ByRef LngData() As Long, ByRef Return_DblData() As
Double) As Boolean
Dim i As Long, l As Long, m As Long
ReDim Return_DblData(LBound(LngData, 1) To UBound(LngData, 1), LBound(LngData, 2) To
UBound(LngData, 2), LBound(LngData, 3) To UBound(LngData, 3))
For i = LBound(LngData, 1) To UBound(LngData, 1)
For l = LBound(LngData, 2) To UBound(LngData, 2)
For m = LBound(LngData, 3) To UBound(LngData, 3)
Return_DblData(i, l, m) = CDbl(LngData(i, l, m))
Next m
Next l
Next i
Array3DCDataTypeLtoD = True
End Function
```

4. Array4D

详细说明：

函数或方法名称	作用
Array4DMax	四维数组最大值及其位置

代码：

```
Option Explicit
Option Base 1
'**************************************************************************
'名称 = Array4D.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 12 月 30 日
'简介 = 操作一维数组
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'**************************************************************************
```

'Array4DMaxWhichLng 四维数组最大值及其位置

```
Rem 未测试
Rem Data()=数据
Rem R_WhichBound1():R_WhichBound2():R_WhichBound3():R_WhichBound4()  4 维度数据的 4 个限
Rem SameMaxNum=相同最大值个数
Public Function Array4DMaxWhichLng(ByRef Data() As Long, ByRef R_WhichBound1() As Long, _
ByRef R_WhichBound2() As Long, ByRef R_WhichBound3() As Long, ByRef R_WhichBound4() As Long, _
ByRef SameMaxNum As Long) As Double
Dim i As Long, l As Long, m As Long, p As Long
Dim c1Data() As Long
Dim n As Long
n = 0
ReDim c1Data(LBound(Data, 1) To UBound(Data, 1), LBound(Data, 2) To UBound(Data, 2), LBound(Data, 3) To UBound(Data, 3), LBound(Data, 4) To UBound(Data, 4))
For p = LBound(Data, 4) To UBound(Data, 4)
For m = LBound(Data, 3) To UBound(Data, 3)
For l = LBound(Data, 2) To UBound(Data, 2)
For i = LBound(Data, 1) To UBound(Data, 1)
```

```
c1Data(i, l, m, p) = Data(i, l, m, p)
'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "Data(" & i & "," & l & ")=" & Data(i, l)
Next i
Next l
Next m
Next p
Dim Part() As Double
For p = LBound(c1Data, 4) To UBound(c1Data, 4)
For m = LBound(c1Data, 3) To UBound(c1Data, 3)
For l = LBound(c1Data, 2) To UBound(c1Data, 2)
n = n + 1
ReDim Preserve Part(n) As Double
For i = LBound(c1Data, 1) To UBound(c1Data, 1)
If (m <= UBound(c1Data, 3)) Then
   If (l <= UBound(c1Data, 2)) Then
     If (i < UBound(c1Data, 1)) Then
'Debug.Print "i=" & i & "l=" & l
'Debug.Print "c1Data(" & i & "," & l & ")=" & c1Data(i, l) & "c1Data(" & i + 1 & "," & l & ")=" &
c1Data(i + 1, l)
          If (c1Data(i, l, m, p) > c1Data(i + 1, l, m, p)) Then
          Part(n) = c1Data(i, l, m, p)
          c1Data(i + 1, l, m, p) = c1Data(i, l, m, p)
          Else
          Part(n) = c1Data(i + 1, l, m, p)
          End If
     End If
  End If
End If
Next i
Next l
Next m
Next p

n = 0
Dim obj As New Array1D001
Array4DMaxWhichLng = obj.Array1D_Max(Part())

For p = LBound(Data, 4) To UBound(Data, 4)
For m = LBound(Data, 3) To UBound(Data, 3)
For l = LBound(Data, 2) To UBound(Data, 2)
For i = LBound(Data, 1) To UBound(Data, 1)
If Array4DMaxWhichLng = Data(i, l, m, p) Then
n = n + 1
ReDim Preserve R_WhichBound1(n)
```

```
ReDim Preserve R_WhichBound2(n)
ReDim Preserve R_WhichBound3(n)
ReDim Preserve R_WhichBound4(n)
R_WhichBound1(n) = i
R_WhichBound2(n) = l
R_WhichBound3(n) = m
R_WhichBound4(n) = p
SameMaxNum = n
End If
Next i
Next l
Next m
Next p
n = 0
                    Set obj = Nothing

End Function
```

5. ArrayEmptyOrNot

详细说明：

函数或方法名称	作用
ArrayEmptyB	数组是否为空 boolean 数据
ArrayEmptyD	数组是否为空 double 数据
ArrayEmpty1	数组是否为空 Long 数据
ArrayEmpty2	数组是否为空 String 数据
ArrayEmptyL	数组是否为空 Long 数据
ArrayEmptyS	数组是否为空 String 数据
ArrayEmptyV	数组是否为空 Variant 数据

代码：

```
'**************************************************************************
'名称 =  ArrayEmptyOrNot.Cls
'来源 = https://www.cnblogs.com/mwming/p/4864955.html?ivk_sa=1024320u
'来源 =渔者.(2015-10-9).[2022-1-1].
'最后修改时间 = 2022 年 9 月 1 日
'简介 = 数组是否为空
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'**************************************************************************
```

'ArrayEmptyB 数组是否为空 boolean 数据

Rem 一: 利用错误捕获功能判断

```
Public Function ArrayEmptyB(ByRef ArrayS() As Boolean) As Boolean
Dim i As Long
On Error GoTo Z
If UBound(ArrayS) > -1 Then
    ArrayEmptyB = False
    Exit Function
End If
Z:
    ArrayEmptyB = True
  ' MsgBox "数组空"
End Function
```

'ArrayEmptyD 数组是否为空 double 数据

```
Public Function ArrayEmptyD(ByRef ArrayS() As Double) As Boolean
Dim i As Long
On Error GoTo Z
If UBound(ArrayS) > -1 Then
    ArrayEmptyD = False
    Exit Function
End If
Z:
    ArrayEmptyD = True
  ' MsgBox "数组空"
End Function
```

'ArrayEmpty1 数组是否为空 Long 数据

Rem 一: 利用错误捕获功能判断

```
Public Function ArrayEmpty1(ByRef ArrayS() As Long) As Boolean
Dim i As Long
On Error GoTo Z
If UBound(ArrayS) > -1 Then
    ArrayEmpty1 = False
    Exit Function
End If
Z:
    ArrayEmpty1 = True
  ' MsgBox "数组空"
End Function
```

'ArrayEmpty2 数组是否为空 String 数据

Rem 二: Join 方法:

```
Public Function ArrayEmpty2(ByRef ArrayS() As String) As Boolean
    If (CStr(Join(ArrayS, ""))) = "" Then
        ArrayEmpty2 = True
        'MsgBox "为空"
    Else
        ArrayEmpty2 = False
    End If
End Function
```

'ArrayEmptyL 数组是否为空 Long 数据

Rem 利用错误捕获功能判断数组是否存在

```
Public Function ArrayEmptyL(ByRef ArrayL() As Long) As Boolean
Dim i As Long
On Error GoTo Z
If UBound(ArrayL) > -1 Then
    ArrayEmptyL = False
    Exit Function
End If
Z:
    ArrayEmptyL = True
  ' MsgBox "数组空"
End Function
```

'ArrayEmptyS 数组是否为空 String 数据

Rem 利用错误捕获功能判断数组是否存在

```
Public Function ArrayEmptyS(ByRef ArrayS() As String) As Boolean
Dim i As Long
On Error GoTo Z
If UBound(ArrayS) > -1 Then
    ArrayEmptyS = False
    Exit Function
End If
Z:
    ArrayEmptyS = True
  ' MsgBox "数组空"
End Function
```

'ArrayEmptyV 数组是否为空 Variant 数据

Rem 利用错误捕获功能判断数组是否存在

```
Public Function ArrayEmptyV(ByRef ArrayV() As Variant) As Boolean
Dim i As Long
On Error GoTo Z
If UBound(ArrayV) > -1 Then
    ArrayEmptyV = False
    Exit Function
End If
Z:
```

```
    ArrayEmptyV = True
  ' MsgBox "数组空"
End Function
```

6. BitBlt

详细说明：

函数或方法名称	作用
BitBltShowTransparencyPic	显示透明图片到指定控件上
BitBltShieldPic	(未成功)

代码：

```
Option Explicit
Option Base 1
```

Rem WHITENESS 使用白色填充目标区域
Rem BLACKNESS 使用黑色填充目标区域

Rem SRCCOPY 直接复制源设备区域到目标设备中
Rem NOTSRCCOPY 复制源设备区域的反色到目标设备中

Rem DSTINVERT 目标矩阵区域颜色取反
Rem MERGECOPY 使用与运算组合原设备矩形区域的颜色和目标设备的画刷
Rem MERGEPAINT 使用或运算将反向的源矩形区域的颜色和目标矩形区域的颜色合并

Rem NOTSRCERASE 使用或运算组合源设备区域与目标设备区域的颜色，然后对结果颜色取反
Rem PATCOPY 复制源设备当前选中的画刷到目标设备
Rem PATINVERT 使用异或运算组合目标设备选中的画刷和目标设备区域的颜色
Rem PATPAINT 通过或运算组合目标区域当前选中的画刷和源设备区域反转的颜色
Rem SRCAND 使用与运算组合源设备和目标设备区域的颜色. 将 Picture1 中颜色与 Picture2 中颜色合并后，再复制到 Picture2 中

Rem SRCERASE 使用与运算组合目标设备区域的反色与源设备区域的颜色
Rem SRCINVERT 使用异或运算组合源设备区域颜色和目标设备区域颜色
Rem SRCPAINT 使用或运算组合源设备区域颜色和目标设备区域颜色

```
'********************************************************************************
'名称 = BitBlt .Cls
'作者 = 张宏
'最后修改时间 = 2018 年 11 月 24 日
'简介 = 显示透明位图
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'********************************************************************************
'Const BLACKNESS = &H42
```

```
'Const DSTINVERT = &H550009
'Const MERGECOPY = &HC000CA
'Const MERGEPAINT = &HBB0226
'Const NOTSRCCOPY = &H330008
'Const NOTSRCERASE = &H1100A6
'Const PATCOPY = &HF00021
'Const PATPAINT = &HFB0A09
'Const PATINVERT = &H5A0049
'Const SRCAND = &H8800C6
'Const SRCCOPY = &HCC0020
'Const SRCERASE = &H440328
'Const SRCINVERT = &H660046
'Const SRCPAINT = &HEE0086
'Const WHITENESS = &HFF0062
Private Declare Function BitBlt& Lib "gdi32" (ByVal hDestDC As Long, _
                                              ByVal X As Long, _
                                              ByVal Y As Long, _
                                              ByVal nWidth As Long, _
                                              ByVal nHeight As Long, _
                                              ByVal hSrcDC As Long, _
                                              ByVal xSrc As Long, _
                                              ByVal ySrc As Long, _
                                              ByVal dwRop As Long)
```

'BitBltShowTransparencyPic 显示透明图片到指定控件上

Rem 显示透明图片到指定控件上
Rem hDCBack 背景句柄，hDCWhite 白底图片句柄，hDCBack 黑底图片句柄
Rem ShowAtX ShowAtY 透明图片显示在背景上的位置
Rem ShowWidth ShowHeight 显示透明图片的宽度和高度
Rem ShowLeft ShowTop 透明图片的显示框左上角坐标点，坐标点是相对于透明图片

```
Function BitBltShowTransparencyPic(ByVal hDCBack As Long, ByVal hDCWhite As Long, _
ByVal hDCBlack As Long, ByVal ShowAtX As Long, ByVal ShowAtY As Long, _
ByVal ShowWidth As Long, ByVal ShowHeight As Long, _
ByVal ShowLeft As Long, ByVal ShowTop As Long)
Rem SRCCOPY 直接把源位图复制到目标区域
'Call BitBlt(Picture1.hDC, 0, 0, Picture1.Width, Picture1.Height, _
                                        Picture1.hDC, 0, 0, SRCCOPY)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 下面 2 个组合成透明图像
Call BitBlt(hDCBack, ShowAtX, ShowAtY, ShowWidth, ShowHeight, hDCWhite, ShowLeft, ShowTop,
vbSrcAnd)
Call BitBlt(hDCBack, ShowAtX, ShowAtY, ShowWidth, ShowHeight, hDCBlack, ShowLeft, ShowTop,
```

```
vbSrcPaint)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'hDCBack.Refresh
End Function
```

'BitBltShieldPic(未成功)

```
Function BitBltShieldPic(ByVal hDCBack As Long, ByVal hDCWait As Long, ByVal hDCWhite As Long, _
ByVal hDCBlack As Long, ByVal hDCShield As Long, ByVal ShowAtX As Long, _
ByVal ShowAtY As Long, ByVal ShowWidth As Long, ByVal ShowHeight As Long, _
ByVal ShowLeft As Long, ByVal ShowTop As Long)
'Call BitBlt(hDCWait, 0, 0, ShowWidth, ShowHeight, hDCWhite, ShowLeft, ShowTop, vbSrcAnd)
'Call BitBlt(hDCWait, 0, 0, ShowWidth, ShowHeight, hDCBlack, ShowLeft, ShowTop, vbSrcPaint)
'Call BitBlt(hDCWait, 0, 0, ShowWidth, ShowHeight, hDCBlack, ShowLeft, ShowTop, vbDstInvert)
End Function
```

7. ControlBoxHaveORNot

详细说明：

函数或方法名称	作用
fChkControls	判断控件是否存在的函数
ClearText3	清除控件实列

代码：

```
'**********************************************************
'*函数功能：判断控件是否存在的函数
'*参　　数：frmObject　　窗体名
'* strControlsName　　待判断的控件名
'* lngIndex 控件 Index 非数组控件可省略该参数
'*返　回　值：True 存在　　False　不存在
'ClearText3 增加列子 2019-4-15
'**********************************************************
```

'fChkControls 判断控件是否存在的函数

```
'Function fChkControls(frmObject As Form, strControlsName As String, Optional lngIndex As Long = -1) As Boolean
Function fChkControls(frmObject As Form, strControlsName As String, ByVal lngIndex As Long) As Boolean
On Error GoTo Err
    Dim strContrName As String
    If lngIndex >= 0 Then
        strContrName = frmObject.Controls(strControlsName)(lngIndex).Name
    Else
        strContrName = frmObject.Controls(strControlsName).Name
    End If
    fChkControls = True
    Exit Function
Err:
    fChkControls = False
End Function
```

'ClearText3 清除控件实列

Rem 去除所有 Form6.text3(1 to 1000)的控件

```
Sub ClearText3()
Dim i As Long
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  清除重新加载
Dim obj As New ControlBoxHaveORNot
For i = 1 To 1000
        If obj.fChkControls(Form6, "Text3", i) = True Then
        Unload Form6.Text3(i)
        Else
        Exit For
        End If
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
End Sub
```

8. ControlBoxOpration

详细说明：

函数或方法名称	作用
ShowAtCenter	居中显示控件
InBoxes	鼠标是否在控件内
SameResizeWidth	控件宽度随另一个控件变化
SameResizeHeight	控件高度随另一个控件变化
SameResizeLeft	控件横坐标随原控件宽度变化
SameResizeTop	控件纵坐标随原控件高度变化
SameResizeBoxesWidth	控件数组宽度随指定控件变化
PutBoxesInBox	控件数组放在指定控件中
PublicNewArrangeAdd	按由小到大水平交换控件位置
BoxesMultiLinesOld	控件排放到另一个控件中
BoxesMultiLines	控件排放到另一个控件中
PutBoxesInBoxXY	(Commond)控件数组放在指定控件中

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
```

```
'**************************************************************************
'名称 = S1_ControlBoxOpration.Cls
'作者 = 张宏
'最后修改时间 =2022 年 1 月 4 日
'简介 = 控件操作
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'增加 S2_PublicControlBoxOpration 中的内容到这里                    2022-1-2
'增加 PutBoxesInBoxXY        2022-1-4
'**************************************************************************
```

```
Private WithEvents ObjectCmd As CommandButton
Private WithEvents ObjectPic As PictureBox
```

'ShowAtCenter 居中显示控件

```
Rem 居中显示控件
Rem ReferBox = 参照控件
Rem ShowBox = 显示控件
```

```
Public Sub ShowAtCenter(ByRef ReferBox As Object, ByRef ShowBox As Object)
ShowBox.Left = ReferBox.Width / 2 - ShowBox.Width / 2
ShowBox.Top = ReferBox.Height / 2 - ShowBox.Height / 2
End Sub
```

'InBoxes 鼠标是否在控件内

Rem 鼠标是否在控件内
Rem Box=原控件
Rem BoxFollow=跟随变化控件
Rem Interval =间隔

```
Public Function InBoxes(ByVal X As Single, ByVal Y As Single, _
ByRef Boxes As Object) As Long
Dim i As Long
For i = 1 To Boxes.UBound
    If X < Boxes(i).Left + Boxes(i).Width And X > Boxes(i).Left And _
    Y < Boxes(i).Top + Boxes(i).Height And Y > Boxes(i).Top Then
        InBoxes = i
    Else
        InBoxes = -1
    End If
Next i
End Function
```

'SameResizeWidth 控件宽度随另一个控件变化

Rem 宽度随原控件变化
Rem Box=原控件
Rem BoxFollow=跟随变化控件
Rem Interval =间隔

```
Public Sub SameResizeWidth(ByRef Box As Object, ByRef BoxFollow As Object, Optional Interval
As Double)
If Box.Width - Interval >= 0 Then '避免宽度为负数
BoxFollow.Width = Box.Width - Interval
End If
End Sub
```

'SameResizeHeight 控件高度随另一个控件变化

Rem 高度随原控件变化
Rem Box=原控件

Rem BoxFollow=跟随变化控件

Rem Interval =间隔

```
Public Sub SameResizeHeight(ByRef Box As Object, ByRef BoxFollow As Object, Optional Interval As Double)
If Box.Height - Interval >= 0 Then ' 避免高度为负数
BoxFollow.Height = Box.Height - Interval
End If
End Sub
```

'SameResizeLeft 控件横坐标随原控件宽度变化

Rem 横坐标随原控件宽度变化

Rem Box=原控件

Rem BoxFollow=跟随变化控件

Rem Interval =间隔

```
Public Sub SameResizeLeft(ByRef Box As Object, ByRef BoxFollow As Object, Optional Interval As Double)
BoxFollow.Left = Box.Width - Interval
End Sub
```

'SameResizeTop 控件纵坐标随原控件高度变化

Rem 纵坐标随原控件高度变化

Rem Box=原控件

Rem BoxFollow=跟随变化控件

Rem Interval =间隔

```
Public Sub SameResizeTop(ByRef Box As Object, ByRef BoxFollow As Object, Optional Interval As Double)
BoxFollow.Top = Box.Height - Interval
End Sub
```

'SameResizeBoxesWidth 控件数组宽度随指定控件变化

Rem 控件数组宽度随指定控件变化

Rem Box=装载 Boxes 的载体

Rem BoxesFollow=需要改变宽度的 Boxes

Rem NumberOfAddBoxes=装载总数

Rem Row =装载列数

Rem IntervalBetweenBox=空间数组各控件之间的宽度

Rem ChangeWidth=改变各控件的宽度在最右边产生空隙。效果不好有重影

```
Public Sub SameResizeBoxesWidth(ByRef Box As Object, ByRef BoxesFollow As Object, _
```

```
ByVal NumberOfAddBoxes As Long, ByVal Row As Long, Optional IntervalBetweenBox As Double,
_
Optional ChangeWidth As Double)
Dim i As Long, l As Long, n As Long
Dim Line As Long
Line = Int((NumberOfAddBoxes - 1) / Row) + 1
For l = 1 To Line
    For i = 1 To Row
        n = n + 1
        If n <= NumberOfAddBoxes Then
        SameResizeWidth Box, BoxesFollow(n), Box.Width / Row * (Row - 1) +
IntervalBetweenBox + ChangeWidth
        BoxesFollow(n).Top = BoxesFollow(0).Top + (l - 1) * BoxesFollow(0).Height
        BoxesFollow(n).Left = Box.Width / Row * (i - 1) - ChangeWidth * i
        End If
    Next i
Next l
End Sub
```

'PutBoxesInBox 控件数组放在指定控件中

Rem Commond 控件数组放在指定控件中
Rem 使用前手动添加控件数组
Rem Box=装载 Boxes 的载体
Rem Boxes=需要防止的控件数组
Rem NumberOfAddBoxes=装载总数
Rem Row =装载列数
Rem IntervalX=影响空间数组总体宽度，最右边于容器的距离

```
Public Sub PutBoxesInBox(ByRef Box As Object, ByRef Boxes As Object, _
ByVal NumberOfAddBoxes As Long, ByVal Row As Long, Optional IntervalX As Double)
Dim i As Long, l As Long, n As Long
Dim Line As Long
On Error GoTo ErrHandle
Line = Int((NumberOfAddBoxes - 1) / Row) + 1
Rem 设置 0 号控件 Boxes
Boxes(0).Top = 0: Boxes(0).Left = 0
Boxes(0).Container = Box
Boxes(0).Visible = False
Boxes(0).Width = Box.Width / Row - IntervalX
Rem 添加和设置 Boxes
For i = 1 To NumberOfAddBoxes
    Load Boxes(i)
    Boxes(i).Visible = True
```

```
    Boxes(i).Container = Box
    Boxes(i).Width = Box.Width / Row - IntervalX
Next i
Rem 防止 Boxes
For l = 1 To Line
    For i = 1 To Row
        n = n + 1
        If n <= NumberOfAddBoxes Then
        Boxes(n).Top = Boxes(0).Top + (l - 1) * Boxes(0).Height
            If i = 1 Then
            Boxes(n).Left = Boxes(i - 1).Left
            Else
            Boxes(n).Left = Boxes(i - 1).Left + Boxes(0).Width
            End If
        End If
    Next i
Next l
ErrHandle:
If Err.Number = 340 Then '没有 0 号控件
Debug.Print Box.Caption
MsgBox "没有 0 号控件"
End If
End Sub
```

'PublicNewArrangeAdd 按由小到大水平交换控件位置

Rem 按由小到大水平交换控件位置
Rem MaxNumber 最大数量
Rem ControlBoxName 控件名称
Rem From 工作界面
Rem ControlBox 控件:例子 From.PicShow

```
Public Sub PublicNewArrangeAdd(ByVal MaxNumber As Long, ByVal ControlBoxName As String, _
ByRef From As Object, ByRef ControlBox As Object)
Dim i As Long, l As Long, A As Variant, B As Variant
For l = 1 To MaxNumber
    For i = 1 To MaxNumber
        If S4Obj1.fChkControls(From, ControlBoxName, i) = True And S4Obj1.fChkControls(From,
ControlBoxName, i + l) = True Then
            If ControlBox(i).Left > ControlBox(i + l).Left Then
              If ControlBox(i).Top = ControlBox(i + l).Top Then '2021-6-2 增加，作用是同行才交
换位置
              A = ControlBox(i + l).Left
              B = ControlBox(i).Left
```

```
                ControlBox(i + l).Left = B
                ControlBox(i).Left = A
                End If
              End If
          End If
    Next i
Next l
End Sub
```

'BoxesMultiLinesOld 控件排放到另一个控件中

Rem 控件排放到另一个控件中
Rem Num 总共控件数
Rem Container 控件:例子 From.PicContainer
Rem Boxes 控件:例子 From.PicShow
Rem Add 额外系数，增加控件高度

```
Public Sub BoxesMultiLinesOld(ByVal Num As Long, ByRef From As Object, ByRef Container As
Object, _
ByRef Boxes As Object, ByVal Add As Double, ByVal BoxesName As String)
Dim Row As Long, Line As Long
Dim i As Long, l As Long, n As Long
Row = Int(Container.Width / Boxes(0).Width) '一排容纳多少控件
Line = Int(Num / (Row + 1)) + 1 '总共有多少排
For l = 1 To Line
    Boxes(n).Left = Boxes(0).Left
    Boxes(n).Top = Boxes(0).Top + (l - 1) * (Boxes(0).Height + Add)
    For i = 1 To Row
            n = n + 1
            If S4Obj1.fChkControls(From, BoxesName, n) = True Then
            Boxes(n).Left = Boxes(n - 1).Left + Boxes(n).Width
            Debug.Print "Boxes(" & n & ").Left=" & Boxes(n).Left
            End If
    Next i
Next l
End Sub
```

'BoxesMultiLines 控件排放到另一个控件中

Rem 控件排放到另一个控件
Rem Num 总共控件数
Rem Container 控件:例子 From.PicContainer
Rem Boxes 控件:例子 From.PicShow

Rem Add　　额外系数，增加控件高度

```
Public Sub BoxesMultiLines(ByVal Num As Long, ByRef From As Object, ByRef Container As Object,
_
ByRef Boxes As Object, ByVal Add As Double, ByVal BoxesName As String)
Dim Row As Long, Line As Long
Dim i As Long, l As Long, n As Long
Dim Frist As Long
Row = Int(Container.Width / Boxes(0).Width) '一排容纳多少控件
Line = Int((Num + 1) / (Row + 1)) + 1 '总共有多少排
For l = 1 To Line
    Boxes(n).Left = Boxes(0).Left
    Boxes(n).Top = Boxes(0).Top + (l - 1) * (Boxes(0).Height + Add)
    If n = 0 Then
        Frist = 1
    Else
        Frist = 1
    End If
    For i = 1 To Row - Frist
        n = n + 1
        If S4Obj1.fChkControls(From, BoxesName, n) = True Then
        Boxes(n).Left = Boxes(n - 1).Left + Boxes(n).Width
        Boxes(n).Top = Boxes(n - 1).Top
        'Debug.Print "Boxes(" & n & ").Left=" & Boxes(n).Left
        'Debug.Print "Boxes(" & n & ").Top=" & Boxes(n).Top
        End If
    Next i
    n = n + 1
Next l
End Sub
```

'PutBoxesInBoxXY(Commond)控件数组放在指定控件中

Rem Commond 控件数组放在指定控件中
Rem 使用前手动添加控件数组
Rem Box=装载 Boxes 的载体
Rem Boxes=需要防止的控件数组
Rem NumberOfAddBoxes=装载总数
Rem Row =装载列数
Rem IntervalX=影响空间数组总体宽度，最右边于容器的距离
Rem IntervalY=影响空间数组总体高度，影响行距

```
Public Sub PutBoxesInBoxXY(ByRef Box As Object, ByRef Boxes As Object, _
ByVal NumberOfAddBoxes As Long, ByVal Row As Long, Optional IntervalX As Double, _
Optional IntervalY As Double, Optional ModifyWidth As Double, Optional LeftX As Double)
```

```
Dim i As Long, l As Long, n As Long
Dim Line As Long
Line = Int((NumberOfAddBoxes - 1) / Row) + 1
Rem 设置 0 号控件 Boxes
Boxes(0).Top = 0: Boxes(0).Left = LeftX
Boxes(0).Container = Box
Boxes(0).Visible = False
Boxes(0).Width = Box.Width / Row - ModifyWidth
Rem 添加和设置 Boxes
For i = 1 To NumberOfAddBoxes
    Load Boxes(i)
    Boxes(i).Visible = True
    Boxes(i).Container = Box
    Boxes(i).Width = Box.Width / Row - ModifyWidth
Next i
Rem 防止 Boxes
For l = 1 To Line
    For i = 1 To Row
        n = n + 1
        If n <= NumberOfAddBoxes Then
        Boxes(n).Top = Boxes(0).Top + (l - 1) * Boxes(0).Height + IntervalY
            If i = 1 Then
            Boxes(n).Left = Boxes(i - 1).Left + LeftX
            Else
            Boxes(n).Left = Boxes(i - 1).Left + Boxes(0).Width + IntervalX
            End If
        End If
    Next i
Next l
End Sub
```

9. DataBackUp

详细说明：

函数或方法名称	作用
BackUp	备份
BackUp3	备份
BackUpMS	备份 MSFlexGrid 控件：MSFWork 内容
BackUpMSFAll	备份 MSFlexGrid 控件数组：MSFWork 内容
BackUp2	备份一维数组 BackUpData1D 数据
WhiteInTxt2D	把 2 维数据写入 TXT
TimeToEreaseBackUpData	文件大于 1G 提示清除
WhiteInTxt	备份内容写到创建文件
SaveDataBeforeAction	创建备份每小时一个文件夹
SaveProgramEachDay	备份每天一次程序内容到，如果存在文件夹后不再备份。

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*************************************************************************
'名称 = DataBackUp.Cls
'作者 = 张宏
'最后修改时间 =2021 年 12 月 31 日
'简介 = 数据备份
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=工程-部件引用-Micorsoft Scripting runtime
'操作历史：
'修改 WhiteInTxt 增加错误提示自行备份                    2020-6-21
'增加 BackUp2                                          2020-6-21
'修改 WhiteInTxt 增加备份提示次数最多 3 次               2020-6-23
'增加 SaveDataBeforeAction                             2022-1-4
'增加 SaveProgramEachDay                               2022-1-4
'*************************************************************************
Dim S4Obj1 As New S4_ControlBoxHaveORNot
Dim S4Obj2 As New S4_ArrayEmptyOrNot
```

'BackUp 备份

Rem 即时备份输入内容

Rem D1GstrTestNameAll 为备份对象 2 维数组

```
Public Sub BackUp()
Dim DataCopy() As Variant
Dim i As Long, n As Long, l As Long
Dim BackUpFile As String
For i = LBound(D1GstrTestNameAll, 1) To UBound(D1GstrTestNameAll, 1)
    For l = LBound(D1GstrTestNameAll, 2) To UBound(D1GstrTestNameAll, 2)
        n = n + 1
        ReDim Preserve DataCopy(n)
        DataCopy(n) = D1GstrTestNameAll(i, l)
    Next l
Next i
BackUpFile = App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd") + "\ImmediatelyBackUp.txt"
WhiteInTxt DataCopy, BackUpFile
TimeToEreaseBackUpData
End Sub
```

'BackUp3 备份

Rem BackUpData2D 为备份对象 2 维数组

Rem 备份输入内容

```
Public Sub BackUp3(ByRef BackUpData2D() As String)
Dim DataCopy() As Variant
Dim i As Long, l As Long, n As Long
Dim BackUpFile As String
Static Times As Long
Times = Times + 1
For i = LBound(BackUpData2D, 1) To UBound(BackUpData2D, 1)
    For l = LBound(BackUpData2D, 2) To UBound(BackUpData2D, 2)
        n = n + 1
        ReDim Preserve DataCopy(n)
        DataCopy(n) = BackUpData2D(i, l)
    Next l
Next i
BackUpFile = App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd") + "\" + CStr(Times) +
"BackUp.txt"
WhiteInTxt DataCopy, BackUpFile
TimeToEreaseBackUpData
End Sub
```

'BackUpMSF 备份 MSFlexGrid 控件：MSFWork 内容

Rem 备份输入内容

```
Public Sub BackUpMSF()
Dim DataCopy() As Variant
Dim i As Long, l As Long
Dim BackUpFile As String
Static Times As Long
ReDim DataCopy(1 To FrmWork.MSFWork(S3GlngChoiceSheet).Rows, 1 To _
FrmWork.MSFWork(S3GlngChoiceSheet).Cols)
For i = 1 To FrmWork.MSFWork(S3GlngChoiceSheet).Rows - 1
    For l = 1 To FrmWork.MSFWork(S3GlngChoiceSheet).Cols - 1
    DataCopy(i, l) = FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(i, l)
    Next l
Next i
Times = Times + 1
BackUpFile = App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd") + "\定时保存\" + CStr(Times)
+ "BackUp.txt"
WhiteInTxt2D DataCopy, BackUpFile
TimeToEreaseBackUpData
End Sub
```

'BackUpMSFAll 备份 MSFlexGrid 控件数组：MSFWork 内容

Rem 定时备份

```
Public Sub BackUpMSFAll()
Dim DataCopy() As Variant
Dim i As Long, l As Long, n As Long
Dim BackUpFile As String
Dim Str As String
Static Times As Long
Times = Times + 1
For n = 0 To S3GlngMaxSheet - 1
    If S4Obj1.fChkControls(FrmWork, "CmdMult", n) = True Then
        ReDim DataCopy(1 To FrmWork.MSFWork(n).Rows, 1 To FrmWork.MSFWork(n).Cols)
        For i = 1 To FrmWork.MSFWork(n).Rows - 1
            For l = 1 To FrmWork.MSFWork(n).Cols - 1
            DataCopy(i, l) = FrmWork.MSFWork(n).TextMatrix(i, l)
            If Str = "" Then '看是否存在内容
            Str = Str + DataCopy(i, l)
            End If
            DoEvents
```

```
                Next l
            Next i
        End If
        If Str <> "" Then
        BackUpFile = App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd") + "\" + CStr(Times) + "_"
+ CStr(n) + "BackUp.txt"
        WhiteInTxt2D DataCopy, BackUpFile
        End If
Next n
TimeToEreaseBackUpData
End Sub
```

'BackUp2 备份一维数组 BackUpData1D 数据

Rem 备份输入内容

```
Public Sub BackUp2(ByRef BackUpData1D() As String)
Dim DataCopy() As Variant
Dim i As Long, n As Long
Dim BackUpFile As String
Static Times As Long
Times = Times + 1
For i = LBound(BackUpData1D) To UBound(BackUpData1D)
            n = n + 1
            ReDim Preserve DataCopy(n)
            DataCopy(n) = BackUpData1D(i)
Next i
BackUpFile = App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd") + "\" + CStr(Times) +
"BackUp.txt"
WhiteInTxt DataCopy, BackUpFile
TimeToEreaseBackUpData
End Sub
```

'WhiteInTxt2D 把 2 维数据写入 TXT

Rem 把 2 维数据分行写入 TXT，以#分隔。

```
Public Function WhiteInTxt2D(ByRef Data2D() As Variant, ByVal TxtFileName As String) As
Boolean
Dim i As Long, l As Long
Dim Str As Variant
Dim Answer As Variant
Static m As Long '防止反复提示无法备份
On Error GoTo ErrLine
```

```
Debug.Print TxtFileName
Open TxtFileName For Output As #1
For i = 1 To FrmWork.MSFWork(S3GlngChoiceSheet).Rows - 1
      For l = 1 To FrmWork.MSFWork(S3GlngChoiceSheet).Cols - 1
      Str = Str & Data2D(i, l) & vbTab
      Next l
      Str = Str & vbCrLf
Next i
Print #1, Str
Close #1
Exit Function
ErrLine:
If Err.Number = 76 Then '路径未找到
   Debug.Print App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd")
   If Dir(App.Path + "\BackUp\", vbDirectory) = "" Then
      MkDir App.Path + "\BackUp\"     '创建该文件
   End If
   If Dir(App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd"), vbDirectory) = "" Then
      MkDir App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd")   '创建该文件夹
      Resume
   End If
End If
m = m + 1
If m < 4 Then '提示 3 次
MsgBox Err.Description & "数据备份出现问题，请自行保存好数据以免丢失", vbInformation
ElseIf m = 1000 Then
m = 1
End If
End Function
```

'TimeToEreaseBackUpData 文件大于 1G 提示清除

```
Public Function TimeToEreaseBackUpData()
Dim Fso1 As New FileSystemObject
Dim Folder1 As Folder
Set Folder1 = Fso1.GetFolder(App.Path + "\BackUp\")
If Folder1.Size > 1073741824 Then '备份大于 1G 提示清除
MsgBox "备份文件大于 1G，请定期清除部分备份内容", vbInformation
End If
End Function
```

'WhiteInTxt 备份内容写到创建文件

Rem 把 1 维数据分行写入 TXT，以#分隔。

```
Public Function WhiteInTxt(ByRef Data1D() As Variant, ByVal TxtFileName As String) As Boolean
Dim i As Long, l As Long
Dim Str As Variant
Dim Answer As Variant
Static m As Long '防止反复提示无法备份
On Error GoTo ErrLine
Debug.Print TxtFileName
Open TxtFileName For Output As #1
For i = LBound(Data1D, 1) To UBound(Data1D, 1)
    Str = Str & Data1D(i) & "#"
Next i
Print #1, Str
Close #1
Exit Function
ErrLine:
If Err.Number = 76 Then '路径未找到
  Debug.Print App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd")
  If Dir(App.Path + "\BackUp\", vbDirectory) = "" Then
    MkDir App.Path + "\BackUp\"    '创建该文件
  End If
  If Dir(App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd"), vbDirectory) = "" Then
    MkDir App.Path + "\BackUp\" + Format(Now, "yyyy-mm-dd")  '创建该文件夹
    Resume
  End If
End If
m = m + 1
If m < 4 Then '提示 3 次
MsgBox "数据备份出现问题，请自行保存好数据以免丢失", vbInformation
ElseIf m = 1000 Then
m = 1
End If
End Function
```

'SaveDataBeforeAction 创建备份每小时一个文件夹

Rem 创建备份每小时一个文件夹
Rem Folder=小文件夹名称
Rem 对文件夹名称有要求通用性较差

```
Public Function SaveDataBeforeAction(ByVal Folder As String)
```

```
Dim r() As String
Dim fs As Variant
Dim p As Variant
Dim file As String
file = App.Path + "\Files\" + Folder + "\DataBackUp\" + Format(Now, "yyyy-mm-dd hh") + "\"
Rem 创建文件夹
S4FN.CreateFolder (file)
Rem 获取源文件
S4FN.GetAllFilesUnderFolders App.Path + "\Files\" + Folder + "\", r, "txt"
Rem 将源文件拷贝到创建的文件夹
Set fs = CreateObject("Scripting.FileSystemObject") '创建文件系统对象 fs
For Each p In r
fs.CopyFile p, file   '使用该对象的 copyfile 方法将源文件复制到目标文件
Next
End Function
```

'SaveProgramEachDay 备份每天一次程序内容到，如果存在文件夹后不再备份。

Rem 备份 APP.Path 每天一次程序内容到，如果存在文件夹后不再备份。

```
Public Function SaveProgramEachDay()
Dim r() As String
Dim fs As Variant
Dim p As Variant
Dim file As String
Rem 创建文件夹
file = App.Path + "\备份" + Format(Now, "yyyy-mm-dd") + "\"
If S4FN.CreateFolder(file) = True Then
Rem 获取源文件
S4FN.FindAllFilesUnderTheFolder App.Path, r
Rem 将源文件拷贝到创建的文件夹
Set fs = CreateObject("Scripting.FileSystemObject") '创建文件系统对象 fs
    For Each p In r
    fs.CopyFile p, file '使用该对象的 copyfile 方法将源文件复制到目标文件
    Next
Else
End If
End Function
```

10. ElementaryMathematics

详细说明：

函数或方法名称	作用
QiuTi_V	球体体积
JuXing_V	矩形体积
YuanZhuTi_V	圆柱体体积
LengZhuTi_V	棱柱体体积
YuanZhuiTi_V	圆柱体积
SanLengZhui_V	三棱锥体积
TaiTi_V	台体体积
YuanTai_V	圆台体积
TuoQiu_V	椭球体积
SanXing_S	扇形面积
YuanXing_S	圆形面积
ShanHuan_S	扇圆面积
TuoYuan_S	椭圆面积

代码：

```
Option Explicit
'*******************************************************************************
'名称 = ElementaryMathematics.Cls
'作者 = 张宏
'最后修改时间 =2022 年 8 月 30 日
'简介 = 基础数学
'声明 = 程序中的英文只做代号，不是标准翻译
'*******************************************************************************
Const π = 3.14159265358979
```

'QiuTi_V 球体体积

Rem 球体体积

```
Public Function QiuTi_V(ByVal r As Double) As Double
QiuTi_V = 4 / 3 * π * r ^ 3
End Function
```

'JuXing_V 矩形体积

Rem 矩形体积

```
Public Function JuXing_V(ByVal a As Double, ByVal b As Double, ByVal c As Double) As Double
JuXing_V = a * b * c
End Function
```

'YuanZhuTi_V 圆柱体体积

Rem 圆柱体体积

```
Public Function YuanZhuTi_V(ByVal r As Double, ByVal h As Double) As Double
YuanZhuTi_V = π * r ^ 2 * h
End Function
```

'LengZhuTi_V 棱柱体体积

Rem 棱锥体积

```
Public Function LengZhuTi_V(ByVal s As Double, ByVal h As Double) As Double
LengZhuTi_V = s * h
End Function
```

'YuanZhuiTi_V 圆柱体积

Rem 圆柱体积

```
Public Function YuanZhuiTi_V(ByVal s As Double, ByVal h As Double) As Double
YuanZhuiTi_V = 1 / 3 * s * h
End Function
```

'SanLengZhui_V 三棱锥体积

Rem 三棱锥体积

```
Public Function SanLengZhui_V(ByVal s As Double, ByVal h As Double) As Double
SanLengZhui_V = 1 / 3 * s * h
End Function
```

'TaiTi_V 台体体积

Rem 台体体积

```
Public Function TaiTi_V(ByVal S1 As Double, ByVal S2 As Double, ByVal h As Double) As Double
TaiTi_V = 1 / 3 * (S1 + Sqr(S1 * S2) + S2) * h
End Function
```

'YuanTai_V 圆台体积

Rem 圆台体积

```
Public Function YuanTai_V(ByVal r1 As Double, ByVal r2 As Double, ByVal h As Double) As Double
YuanTai_V = 1 / 3 * π * h * (r1 ^ 2 + r1 * r2 + r2 ^ 2)
End Function
```

'TuoQiu_V 椭球体积

Rem 椭球体积

```
Public Function TuoQiu_V(ByVal a As Double, ByVal b As Double, ByVal c As Double) As Double
TuoQiu_V = 4 / 3 * π * a * b * c
End Function
```

'SanXing_S 扇形面积

Rem 扇形面积

```
Public Function SanXing_S(ByVal JiaoDu As Double, ByVal r As Double) As Double
SanXing_S = JiaoDu * π * r ^ 2 / 360
End Function
```

'YuanXing_S 圆形面积

Rem 圆形面积

```
Public Function YuanXing_S(ByVal r As Double) As Double
YuanXing_S = π * r ^ 2
End Function
```

'ShanHuan_S 扇圆面积

Rem 扇圆面积

```
Public Function ShanHuan_S(ByVal JiaoDu As Double, ByVal r1 As Double, ByVal r2 As Double) As
Double
If r1 > r2 Then
ShanHuan_S = JiaoDu * π * (r1 ^ 2 - r2 ^ 2) / 360
```

```
Else
ShanHuan_S = JiaoDu *  π  * (r2 ^ 2 - r1 ^ 2) / 360
End If
End Function
```

'TuoYuan_S 椭圆面积

Rem 椭圆面积

```
Public Function TuoYuan_S(ByVal ChangZhou As Double, ByVal DuanZhou As Double) As Double
TuoYuan_S =  π  * ChangZhou * DuanZhou
End Function
```

11. FileOpenOpration

详细说明：

函数或方法名称	作用
OpenFile	打开文本文件 txt、xls、xlsx

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
```

```
'**************************************************************************
'名称 = S3_FileOpenOpration.cls
'作者 = 张宏
'最后修改时间 =2020 年 7 月 21 日
'简介 = 文件打开操作
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'**************************************************************************
```

```
Dim S4Obj1 As New S4_FileRead190505
```

'OpenFile 打开文本文件 txt、xls、xlsx

```
Rem https://zhidao.baidu.com/question/190896228.html
Rem 匿名用户(2016-5-18)/[2020-7-19]
```

```
Public Sub OpenFile()
Dim FileNameLoad As String
Dim R2D() As Variant, R1D() As Variant
  ' 设置“CancelError”为 True
    FrmWork.CommonDialog1.CancelError = True
    On Error GoTo ErrHandler
    ' 设置标志
    FrmWork.CommonDialog1.Flags = cdlOFNHideReadOnly
    ' 设置过滤器
    FrmWork.CommonDialog1.Filter = "Txt (*.txt)|*.txt|Excel(*.xls)|*.xls|Excel(*.xlsx)|*.xlsx|"
    ' 指定缺省的过滤器
    FrmWork.CommonDialog1.FilterIndex = 2
    ' 显示“打开”对话框
    FrmWork.CommonDialog1.ShowOpen
    ' 显示选定文件的名字
    FileNameLoad = FrmWork.CommonDialog1.FileName
```

```
    S4Obj1.FileReadByLine FileNameLoad, R2D, R1D
    If Right(FileNameLoad, 3) = "txt" Then
    GetStringTxt R1D
    Else
    GetStringExcel R2D
    End If
    'GetString
    'WriteInMSFWork
    'SwichLoad = True
    Exit Sub
ErrHandler:
    ' 用户按了“取消”按钮
    'MsgBox "出现错误", vbCritical
End Sub
```

Rem OpenFile 子程序

```
Private Sub GetStringTxt(ByRef Data() As Variant)
Dim i As Long, l As Long
Dim Str() As String
For i = LBound(Data) To UBound(Data)
    Str = Split(Data(i), vbTab)
    For l = LBound(Str) To UBound(Str)
    FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(i, l + 1) = Str(l)
    Next l
Next i
End Sub
```

Rem OpenFile 子程序

```
Private Sub GetStringExcel(ByRef Data2D() As Variant)
Dim i As Long, l As Long
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
    FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(i, l) = Data2D(i, l)
    Next l
Next i
End Sub
```

12. FileOperation

详细说明：

函数或方法名称	作用
CreateNewFilePath	输入一个文件地址和文件名称，获得该位置的文件地址+名称，输入的文件地址可以带其他文件名，最终地址会是新文件名
OpenFile	打开文本文件 txt、xls、xlsx
GetFileName	获得文件路径中的文件名称
OpenFolderOrFile	用 explorer.exe 打开文件
GetFileFolder	获得装载文件的文件夹
FindAllFilesUnderTheFolder	在相应文件夹下寻找文件，可以根据后缀名查找
TreeSearch(未通过)	树形查找文件
MicrosoftScriptingRuntiomFindFolder	寻找返回文件位置
GetAllFilesUnderFolders	获得该文件夹下所有文件

代码：

```
Option Explicit
Option Base 1
'*********************************************************************************
'名称 = S4_FileOperation.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 12 月 31 日
'简介 = 文件操作
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=1.需要工程->引用->Microsoft Scripting Runtiom
'参考文献
'操作历史：
'FileNameOperation 变成 FileOperation                    2021-12-31
'添加 FolderChange                                       2021-12-31
'FolderSize                                              2021-12-31
'*********************************************************************************
Dim ReturnNameAndPathAdd() As String
```

'CreateNewFilePath 输入一个文件地址和文件名称，获得该位置的文件地址+名称，输入的文件地址可以带其他文件名，最终地址会是新文件名

Rem 注意：给的 FilePath 带另外的文件名，将会被新文件名代替！组成新的地址
Rem 输入一个文件地址和文件名称，获得该位置的文件地址+名称。
Rem 输入的文件地址可以带其他文件名，最终地址会是新文件名
Rem FilePath=一个文件地址
Rem FileName=文件名称
Rem ReturnNewFilePath=文件地址+文件名称

```
Sub CreateNewFilePath(FilePath, FileName, ReturnNewFilePath)
On Error GoTo ErrHandler
Dim FileName1() As String
Dim Name As String, Name1 As String
Dim i As Long
FileName1 = Split(FilePath, "\")
Name1 = CStr(FileName1(UBound(FileName1)))
If InStr(Name1, ".") > 0 Then
For i = LBound(FileName1) To (UBound(FileName1) - 1)
Name = Name + FileName1(i) + "\"
Next i
Else
ReturnNewFilePath = FilePath + FileName
End If
ReturnNewFilePath = Name + FileName
Exit Sub
ErrHandler:
MsgBox "FileNameOperation:CreateNewFilePath 出错"
End Sub
```

'OpenFile 打开文本文件 txt、xls、xlsx

Rem 需要 CommonDialog 控件

```
Sub OpenFile(ByRef FrmMain As Object, ByRef CommonDialog1 As Object, ByRef
ReturnFileNameLoad As String)
    FrmMain.CommonDialog1.CancelError = True ' 设置“CancelError”为 True
    On Error GoTo ErrHandler ' 设置标志
    FrmMain.CommonDialog1.Flags = cdlOFNHideReadOnly ' 设置过滤器
    FrmMain.CommonDialog1.Filter = "Excel (*.xlsx)|*.xlsx|Excel (*.xls)|*.xls" '  Text
(*.txt)|*.txt|Docs (*.doc)|*.doc 指定缺省的过滤器
```

```
    FrmMain.CommonDialog1.FilterIndex = 2 ' 显示“打开”对话框
    FrmMain.CommonDialog1.ShowOpen ' 显示选定文件的名字
    ReturnFileNameLoad = FrmMain.CommonDialog1.FileName
    Exit Sub
ErrHandler: ' 用户按了“取消”按钮
MsgBox "FileNameOperation:OpenFile 出错"
End Sub
```

'GetFileName 获得文件路径中的文件名称

Rem 获得文件路径中的文件名称
Rem FileNameAndPath 文件名称&路径
Rem ReturnName 返回名称 例子：1
Rem ReturnName 返回名称带后缀 例子：1.txt

```
Sub GetFileName(FileNameAndPath, ReturnName, ReturnNameWithSuffix)
On Error GoTo ErrHandler
Dim FileName() As String
If FileNameAndPath <> "" Then
    FileName = Split(FileNameAndPath, "\")
    ReturnName = FileName(UBound(FileName))
    ReturnNameWithSuffix = FileName(UBound(FileName))
    FileName = Split(ReturnName, ".")
    ReturnName = FileName(LBound(FileName))
Else
MsgBox "FileNameAndPath=空值，没有获得路径！"
End If
Exit Sub
ErrHandler:
MsgBox "FileNameOperation:GetFileName 出错"
End Sub
```

'OpenFolderOrFile 用 explorer.exe 打开文件

Rem FilePath 文件详细地址或者文件夹详细地址

```
Sub OpenFolderOrFile(FilePath)
On Error GoTo ErrHandler
Shell "explorer.exe " & Chr(34) & FilePath, 1
Exit Sub
ErrHandler:
MsgBox "FileNameOperation:OpenFolderOrFile 出错，可能是非 explorer.exe 打开类型。"
End Sub
```

'GetFileFolder 获得装载文件的文件夹

Rem FileNameAndPath 文件的名称&路径
Rem ReturnFileFolder 返回装它的文件夹

```
Sub GetFileFolder(FileNameAndPath, ReturnFileFolder)
On Error GoTo ErrHandler
Dim FileName() As String
Dim i As Long
If FileNameAndPath <> "" Then
    FileName = Split(FileNameAndPath, "\")
    For i = LBound(FileName) To (UBound(FileName) - 1)
    ReturnFileFolder = ReturnFileFolder + FileName(i) + "\"
    Next i
Else
MsgBox "FileNameAndPath=空值，没有获得路径！"
End If
Exit Sub
ErrHandler:
MsgBox "FileNameOperation:GetFileFolder 出错"
End Sub
```

'FindAllFilesUnderTheFolder 在相应文件夹下寻找文件，可以根据后缀名查找

Rem FindAllFilesUnderTheFolder=False 为没找到。
Rem 在相应文件夹下寻找文件，可以根据后缀名查找。
Rem 不能够找到文件夹下下级文件夹的内容。
Rem 来源：下面代码主要部分来源 **http://www.wb86.com/post/34.html**
Rem FileFolder 目标文件夹 例子 **C:/**
Rem ReturnNameAndPath 返回名称&路径
Rem 查询指定文件 **FileSuffix** 列 **:"txt",**忽略是全部

```
Public Function FindAllFilesUnderTheFolder(FileFolder As String, ReturnNameAndPath() As String,
_Optional FileSuffix As String = "") As Boolean
Dim Files As New Collection '声明一个集合
Dim n As Long, n1 As Long
Dim FileName
Dim i As Long
If Left(FileFolder, Len(FileFolder) - 1) = "\" Then
FileName = Dir(FileFolder & "*.*")
Else
FileName = Dir(FileFolder & "\*.*")  '返回目录下第一个文件，这里可改成别的文件夹名
```

```
End If
Files.Add FileName                    '将找到的第 1 个文件名添加到集合里
Do While FileName <> ""      '循环查找所有文件
    FileName = Dir
    Files.Add FileName                '将找到的文件名添加到集合里
Loop
For Each FileName In Files
    For i = 1 To Len(FileName)
        If Left(Right(FileName, i), 1) = "." Then Exit For
    Next i
    If InStr(FileName, ".") <> 0 Then
        If InStr(FileName, ".scf") <> 0 Or InStr(FileName, ".lnk") Or InStr(FileName, ".url") Then
            FileName = Left(FileName, Len(FileName) - i)
        End If
    End If
    If FileName <> "" Then
        n = n + 1
        ReDim Preserve ReturnNameAndPath(n)
        If Right(FileFolder, 1) = "\" Then
        ReturnNameAndPath(n) = FileFolder + FileName
        Else
        ReturnNameAndPath(n) = FileFolder + "\" + FileName
        End If
    End If
Next
n1 = 0
Dim ReturnNameAndPath1() As String
If FileSuffix <> "" Then
        If InStr(FileSuffix, ".") > 0 Then
        Else
        FileSuffix = "." + FileSuffix
        End If
        If n > 0 Then
        For i = LBound(ReturnNameAndPath) To UBound(ReturnNameAndPath)
            If InStr(ReturnNameAndPath(i), FileSuffix) > 0 Then
            n1 = n1 + 1
            ReDim Preserve ReturnNameAndPath1(n1)
            ReturnNameAndPath1(n1) = ReturnNameAndPath(i)
            End If
        Next i
        End If
        Erase ReturnNameAndPath
        ReturnNameAndPath = ReturnNameAndPath1
End If
```

```
If n = 0 Then
FindAllFilesUnderTheFolder = False
Else
FindAllFilesUnderTheFolder = True
        If n1 = 0 And FileSuffix <> "" Then
        FindAllFilesUnderTheFolder = False
        End If
End If
End Function
```

'TreeSearch(未通过)树形查找文件

Rem 测试未通过
Rem 来源：https://blog.csdn.net/mouday/article/details/61203177
Rem sPath= 文件夹名称最后不带"/"。
Rem sFileSpec = 后缀。例子："*.*" 。"*.exe" 。"*.vbp"

```
Public Function TreeSearch(ByVal sPath As String, ByVal sFileSpec As String) As Long
DoEvents
Static Files As Long
Dim sDir As String
Dim sSubDirs() As String
Dim Index As Long
Dim sFiles() As String
If Right(sPath, 1) <> "\" Then sPath = sPath & "\"
'获取文件名和数目
sDir = Dir(sPath & sFileSpec)
Do While Len(sDir)
Files = Files + 1
ReDim Preserve sFiles(1 To Files)
sFiles(Files) = sPath & sDir
'显示到列表
'List1.AddItem sFiles(Files)
'List1.ListIndex = List1.ListCount - 1
sDir = Dir
Loop
'获取文件夹名称
Index = 0
sDir = Dir(sPath, vbDirectory)
Do While Len(sDir) 'sDir <> ""
If sDir <> "." And sDir <> ".." Then
If GetAttr(sPath & sDir) And vbDirectory Then
Index = Index + 1
```

```
ReDim Preserve sSubDirs(1 To Index)
sSubDirs(Index) = sPath & sDir & "\"
End If
End If
sDir = Dir
Loop
'递归调用，获取子文件夹目录
For Index = 1 To Index
Call TreeSearch(sSubDirs(Index), sFileSpec)
Next Index
TreeSearch = Files
End Function
```

'MicrosoftScriptingRuntiomFindFolder 寻找返回文件位置

Rem 主要代码来源：https://zhidao.baidu.com/question/399676979.html
Rem 测试通过
Rem 使用 VB 的菜单：[工程] -- [引用]，在出现的窗口，勾选：Microsoft Scripting Runtiom
Rem 输入主文件夹，返回 Return_SubFileFolderNameAndPath，所有子文件夹的名称&路径
Rem FileFolder 目标文件夹 例子 C:/

```
Public Function MicrosoftScriptingRuntiomFindFolder(FileFolder As String,
ReturnSubFileFolderNameAndPath() As String) As Boolean
Dim fs, f, f1, s, sf
Dim n As Long
On Error GoTo ErrHandler
If Right(FileFolder, 1) <> "\" Then
FileFolder = FileFolder + "\"
End If
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(FileFolder)
Set sf = f.SubFolders

Erase ReturnSubFileFolderNameAndPath
For Each f1 In sf
n = n + 1
ReDim Preserve ReturnSubFileFolderNameAndPath(n)
ReturnSubFileFolderNameAndPath(n) = FileFolder + f1.Name
Next

If n = 0 Then
MicrosoftScriptingRuntiomFindFolder = False
Else
MicrosoftScriptingRuntiomFindFolder = True
```

```
End If
Exit Function
ErrHandler:
MsgBox "出错"
End Function
```

' GetAllFilesUnderFolders 获得该文件夹下所有文件

Rem 使用 VB 的菜单：[工程] -- [引用]，在出现的窗口，勾选：Microsoft Scripting Runtiom
Rem 获得所有文件的名称和路径，保存在 ReturnNameAndPathAdd 中
Rem MainFileFolder 要查询所有文件的父文件夹
Rem 查询指定文件 FileSuffix 例子 :"txt",忽略是全部查找
Rem GetAllFilesUnderFolders = True 获得文件地址

```
Public Function GetAllFilesUnderFolders(MainFileFolder As String, ReturnNameAndPathAdd() As String, _
Optional FileSuffix As String = "") As Boolean
Dim ReturnNameAndPath() As String
Dim RName() As String
Dim Name As Variant
Dim n As Long, i As Long
Dim NameAndPath
Dim Swich As Boolean, Swich1 As Boolean
Static Num As Long
'On Error GoTo ErrHandler
Rem 获得总文件夹下的所有文件名称的路径，后缀是可选的过滤条件。
Swich1 = FindAllFilesUnderTheFolder(MainFileFolder, ReturnNameAndPath(), FileSuffix)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 获得该文件夹下的所有文件名称
If Swich1 = True Then
For Each NameAndPath In ReturnNameAndPath
Num = Num + 1
ReDim Preserve ReturnNameAndPathAdd(Num)
ReturnNameAndPathAdd(Num) = NameAndPath
GetAllFilesUnderFolders = True
Next
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 获得总文件夹下的所有文件夹名。
Swich = MicrosoftScriptingRuntiomFindFolder(MainFileFolder, RName())
    If Swich = False Then
    Rem 文件夹下没有其他文件
    Exit Function
    Else
```

```
            For Each Name In RName()
            n = n + 1
            MainFileFolder = Name
            Rem 递归，反复寻找
            Call GetAllFilesUnderFolders(MainFileFolder, ReturnNameAndPathAdd(), FileSuffix)
            Rem 已经没有下层文件夹了运行下面
            Name = ""
            MainFileFolder = ""
            Next Name
            Num = 0 '重复点击开始比较不出现重复内容!
        End If
'ErrHandler:
End Function
```

```
Public Function CreateFolder(FolderNameAndPath) As Boolean
If Dir(FolderNameAndPath, vbDirectory) = "" Then
MkDir (FolderNameAndPath)
CreateFolder = True
Else
CreateFolder = False
MsgBox "文件夹已经存在."
End If
End Function
```

Rem 替换文件

Rem FolderNameAndPath=文件路径名称

```
Public Function FolderChange(ChoiceFile, FolderNameAndPath) As Variant
If Dir(FolderNameAndPath, vbDirectory) = "" Then
    MsgBox "没文件出错"
Else
    Dim SourceFile, DestinationFile
    SourceFile = ChoiceFile     ' 指定源文件路径和名。
    DestinationFile = FolderNameAndPath   ' 指定目标路径和文件名。
    If DestinationFile <> SourceFile Then
    FileCopy SourceFile, DestinationFile      ' 将源文件的内容复制到目的文件中
    End If
End If
End Function
```

Rem 文件夹大小

Rem FolderNameAndPath=文件路径名称

```
Public Function FolderSize(FolderNameAndPath) As Variant
Dim FSO As Object
If Dir(FolderNameAndPath, vbDirectory) = "" Then
    MsgBox "没文件出错"
Else
```

```
        Set FSO = CreateObject("Scripting.FileSystemObject")
        Set f = FSO.GetFile(FolderNameAndPath)
        FolderSize = f.Size
        Debug.Print FolderNameAndPath
End If
End Function
```

13. FileRead

详细说明：

函数或方法名称	作用
FileReadByLine	读取文件
SubTxt	FileReadByLine 子程序
SubExcel	FileReadByLine 子程序
FileReadByLineTexT	读取 Txt 文件
FileReadByLineExcel	读取 Excel 文件
FileReadByLineTexT2D	读取 2 维 Txt 文件
SubSplitTxtLine	FileReadByLineTexT2D 子程序
FileReadByLineRange	逐行读取 txt,xlsx,xls
FileReadByLineRangeNum	文件逐行读取 txt,xlsx,xls 限制读取条数
FileReadByLineRangeNumUnRead0	文件逐行读取 txt,xlsx,xls 限制读取条数，Txt 不读取 0,EXCEL 不读取 0
FileReadByLineExcelQuick	快速逐行读取 Excel 文件
FileReadByLineExcelRange	逐行读取 Excel 文件限制范围
FileReadByLineTexT1D2D	读取并返回 Text 内容一维数据、二维数据
FindLongestLine	SubSplitTxtLine 子程序
Array1DMax	FindLongestLine 子程序
WhiteInTxtLongTranspositionAddTitle	二维数据转置分行写入 TXT，以逗号分隔，增加标题文本

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*********************************************************************************
'名称 = FileRead.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 2 月 26 日
'简介= 文件数据读取
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=1.需要工程->引用->MicroSoft Excel 12.0 Object Library
'参考文献=[3]347-349[64]
'操作历史：
'修改程序结构                                                    2020-2-26
'增加 FileReadByLineTexT1D2D                                     2022-1-4
'增加 FindLongestLine                                            2022-1-4
'*********************************************************************************
```

'FileReadByLine 读取文件

Rem 读取文件
Rem FileName=用户选择的文件名
Rem ReturnDataE2D(j, i) =返回 EXCEL 内容 xlSheet.Cells(j, i)
Rem ReturnDataT() =返回 TEXT 内容

```
Public Function FileReadByLine(ByVal FilePath As String, _
ByRef ReturnDataE2D() As Variant, ByRef ReturnDataT() As Variant) As Boolean
Dim file As String, FileType As String, FileArray() As String
Dim ReturnLine As Long
file = FilePath '用户选择的文件名
FileArray = Split(file, ".")
    If IsArray(FileArray) Then
        FileType = FileArray(UBound(FileArray)) 'FileType 就是文件类型名
        Else
        MsgBox "选择的文件名错误,或者没有扩展名 Exit Function FileReadByLine！ " '
或者没有扩展名
        Exit Function
    End If
    SubTxt FileType, FilePath, ReturnDataT, ReturnLine, FileReadByLine
    SubExcel FileType, FilePath, ReturnDataE2D, FileReadByLine
End Function
```

Rem FileReadByLine 的子程序
Rem 读取 TXT 文件

```
Private Sub SubTxt(ByVal FileType As String, ByVal FilePath As String, _
ByRef ReturnDataT() As Variant, ByRef ReturnLine As Long, ByRef ReturnRead As Boolean)
Dim B1 As String, n As Long
    If FileType = "txt" Then
        n = 0
        Open FilePath For Input As #1
        Do While Not EOF(1)                                    '循环至文件尾.
        Line Input #1, B1                                      '读入一行数据并将其赋予某
变量.
        n = n + 1
        ReDim Preserve ReturnDataT(n)
        ReturnDataT(n) = B1
        DoEvents
        Loop
        ReturnLine = n '总排数
        Close #1
        ReturnRead = True
    End If
End Sub
```

Rem FileReadByLine 的子程序

Rem 读取 Excel 文件

```
Private Sub SubExcel(ByVal FileType As String, ByVal FilePath As String, _
ByRef ReturnDataE2D() As Variant, ByRef ReturnRead As Boolean)
Dim xhang As Long: Dim ylie As Long
Dim i As Integer, l As Integer, j As Integer
Dim xlapp As Object, xlBook As Object, xlSheet As Object
If FileType = "xlsx" Or FileType = "xls" Then
    Set xlapp = New Excel.Application
    xlapp.Visible = False
    Set xlBook = xlapp.Workbooks.Open(FilePath)      '打开 EXCEL 工作簿
    Set xlSheet = xlBook.Worksheets(1)                    '打开 EXCEL 工作表
    xhang = xlSheet.UsedRange.Rows.Count
    ylie = xlSheet.UsedRange.Columns.Count
    ReDim ReturnDataE2D(1 To xhang, 1 To ylie)
    For j = 1 To xhang
        For i = 1 To ylie
            ReturnDataE2D(j, i) = xlSheet.Cells(j, i)
        Next i
    Next j

    xlBook.Close (True)                    '关闭 EXCEL 工作簿
    xlapp.Quit                                  '关闭 EXCEL
    Set xlapp = Nothing                   '释放 EXCEL 对象
    'MsgBox "数据获得"
    ReturnRead = True
End If
End Sub
```

'FileReadByLineTexT 读取 Txt 文件

Rem FileReadByLineTexT 读取 Txt 文件

Rem FilePath=用户选择的文件名

Rem ReturnDataT()=返回 TEXT 内容

```
Public Function FileReadByLineTexT(ByVal FilePath As String, _
ByRef ReturnDataT() As Variant) As Boolean
Dim file As String, FileType As String, FileArray() As String
Dim ReturnLine As Long
file = FilePath '用户选择的文件名
FileArray = Split(file, ".")
        If IsArray(FileArray) Then
            FileType = FileArray(UBound(FileArray)) 'sType 就是文件类型名
            Else
```

```
                MsgBox "选择的文件名错误,或者没有扩展名 Exit Function FileReadByLineTexT!
" '或者没有扩展名
                Exit Function
            End If
SubTxt FileType, FilePath, ReturnDataT, ReturnLine, FileReadByLineTexT
End Function
```

'FileReadByLineExcel 读取 Excel 文件

Rem 读取 Excel
Rem FileName:用户选择的文件名
Rem ReturnDataE1D(K) 返回 EXCEL 内容 xlSheet.Cells(j, i) 一维数组
Rem ReturnDataE2D(j, i) 返回 EXCEL 内容 xlSheet.Cells(j, i) 二维数组

```
Public Function FileReadByLineExcel(ByVal FilePath As String, _
ByRef ReturnDataE1D() As Variant, ByRef ReturnDataE2D() As Variant) As Boolean
Dim file As String, FileType As String, FileArray() As String
file = FilePath '用户选择的文件名
FileArray = Split(file, ".")
            If IsArray(FileArray) Then
                FileType = FileArray(UBound(FileArray)) 'sType 就是文件类型名
                Else
                MsgBox "选择的文件名错误,或者没有扩展名 Exit Function FileReadByLineExcel!
" '或者没有扩展名
                Exit Function
            End If
 If FileType = "xlsx" Or FileType = "xls" Then
  'MsgBox "读取数据"
    Dim i As Long, l As Long, j As Long, k As Long
    k = 0
    Dim xlapp As Object, xlBook As Object, xlSheet As Object
    Set xlapp = New Excel.Application
    xlapp.Visible = False
    Set xlBook = xlapp.Workbooks.Open(FilePath)      '打开 EXCEL 工作簿
    Set xlSheet = xlBook.Worksheets(1)                    '打开 EXCEL 工作表
    Dim xhang As Long: Dim ylie As Long
    xhang = xlSheet.UsedRange.Rows.Count
    ylie = xlSheet.UsedRange.Columns.Count
    ReDim ReturnDataE2D(1 To xhang, 1 To ylie)
    Debug.Print "行数 Xhang=" & xhang & "列数 Ylie=" & ylie
    For j = 1 To xhang
        For i = 1 To ylie
            ReturnDataE2D(j, i) = xlSheet.Cells(j, i)
            If xlSheet.Cells(j, i) <> "" Then
```

```
                    k = k + 1
                    ReDim Preserve ReturnDataE1D(k)
                    ReturnDataE1D(k) = xlSheet.Cells(j, i)
                End If
            Next i
        Next j
        xlBook.Close (True)                         '关闭 EXCEL 工作簿
        xlapp.Quit                                          '关闭 EXCEL
        Set xlapp = Nothing                         '释放 EXCEL 对象
        'MsgBox "数据获得"
        FileReadByLineExcel = True
End If
End Function
```

'FileReadByLineTexT2D 读取 2 维 Txt 文件

Rem 读取 Txt

Rem ReturnDataT(n)返回 TEXT 内容

```
Public Function FileReadByLineTexT2D(ByVal FilePath As String, ByRef ReturnData2D() As Variant,
_ByRef ReturnData2DTransposition() As Variant) As Boolean    'text 改个 MULTIPLINE 就可以了
Dim file As String, FileType As String, FileArray() As String
Dim ReturnDataT() As Variant
Dim ReturnLine As Long
On Error GoTo ErrHandle
file = FilePath '用户选择的文件名
FileArray = Split(file, ".")
            If IsArray(FileArray) Then
                FileType = FileArray(UBound(FileArray)) 'sType 就是文件类型名
                Else
                MsgBox "选择的文件名错误,或者没有扩展名 Exit Function
FileReadByLineTexT2D！ " '或者没有扩展名
                Exit Function
            End If
SubTxt FileType, FilePath, ReturnDataT, ReturnLine, FileReadByLineTexT2D
SubSplitTxtLine ReturnDataT, ReturnData2D, ReturnLine, ReturnData2DTransposition
FileReadByLineTexT2D = True
Exit Function
ErrHandle:
MsgBox "FileRead007:FileReadByLineTexT2D 出错!"
FileReadByLineTexT2D = False
Resume Next
End Function
```

Rem FileReadByLineTexT2D 的子程序

Rem 把读取的文本行依靠"#"分成数据

```
Private Sub SubSplitTxtLine(ByRef ReturnDataT() As Variant, ByRef ReturnData2D() As Variant, _
ByVal n As Long, ByRef ReturnData2DTransposition() As Variant)
Dim a() As String
Dim p As Long, m As Long, i As Long, l As Long
a = Split(ReturnDataT(1), "#")
p = UBound(a) - LBound(a)
ReDim ReturnData2D(1 To p, 1 To n) 'X=p Y=n
n = 0
For m = LBound(ReturnDataT) To UBound(ReturnDataT)
a = Split(ReturnDataT(m), "#")
p = UBound(a) - LBound(a)
    For i = 0 To p - 1
        If a(i) = "" Then
        a(i) = 0
        End If
    ReturnData2D(i + 1, m) = CVar(a(i))
    'Debug.Print "ReturnData2D(" & i + 1 & "," & m & ")=" & ReturnData2D(i + 1, m)
    Next i
Next m
ReDim ReturnData2DTransposition(LBound(ReturnData2D, 2) To UBound(ReturnData2D, 2),
LBound(ReturnData2D, 1) To UBound(ReturnData2D, 1))
For i = LBound(ReturnData2D, 1) To UBound(ReturnData2D, 1)
    For l = LBound(ReturnData2D, 2) To UBound(ReturnData2D, 2)
    ReturnData2DTransposition(l, i) = ReturnData2D(i, l)
    Next l
Next i
End Sub
```

'FileReadByLineRange 逐行读取 txt,xlsx,xls

Rem mvarFileName:用户选择的文件名
Rem Return_DataE2D(j, i) 返回 EXCEL 内容 xlSheet.Cells(j, i) j=Line i=Row
Rem Return_DataT(n) 返回 TEXT 内容
Rem Line 读取的行数，TEXT 中为行数,EXCEL 中也为行数
Rem Row ROW 只在 EXCEL 中生效
Rem 当 ROW,LINE 为 0 的时候全部读取

```
Public Function FileReadByLineRange(ByVal mvarFilePath As String, ByVal Line As Long, _
ByVal Row As Long, ByRef Return_DataE2D() As Variant, ByRef Return_DataT() As Variant) As
String   'text 改个 MULTIPLINE 就可以了
Dim sFile$
Dim sType$, arrTmp
sFile = mvarFilePath '用户选择的文件名
```

```
arrTmp = Split(sFile, ".")
                If (IsArray(arrTmp)) Then
                  sType = arrTmp(UBound(arrTmp)) 'sType 就是文件类型名
                  Else
                  MsgBox "选择的文件名错误,或者没有扩展名 Exit Function FileReadByLineRange !
" '或者没有扩展名
                  Exit Function
               End If
      If (sType = "txt") Then
                Dim B1 As String, n As Long
                n = 0
                If (Row <> 0) Then
                MsgBox "TEXT 文档暂时不支持 ROW,程序继续运行"
                End If
                 Open mvarFilePath For Input As #1
                Do While Not EOF(1)
'循环至文件尾.
                 Line Input #1, B1
'读入一行数据并将其赋予某变量.
                 n = n + 1
                 If (n > Line And Line <> 0) Then
                 Exit Do
                 End If
                 ReDim Preserve Return_DataT(n)
                 Return_DataT(n) = B1
                 DoEvents
                 Loop
                  Dim i As Long
                    If (n < Line) Then
                    For i = n + 1 To Line
                    ReDim Preserve Return_DataT(i)
                    Return_DataT(i) = "Null"
                    Next i
                    End If
                 n = 0
                 Close #1
                 FileReadByLineRange = sType
      End If
If (sType = "xlsx" Or sType = "xls") Then
      'MsgBox "读取数据"
      Dim l As Integer, j As Integer
      Dim xlapp As Object, xlBook As Object, xlSheet As Object
      Set xlapp = New Excel.Application
      xlapp.Visible = False
```

```
    Set xlBook = xlapp.Workbooks.Open(mvarFilePath)      '打开 EXCEL 工作簿
    Set xlSheet = xlBook.Worksheets(1)                                    '打开 EXCEL 工作
表
    If (Line = 0 And Row = 0) Then '  当没有给出值时候，读取全部 16-10-7
    Line = xlSheet.UsedRange.Rows.Count
    Row = xlSheet.UsedRange.Columns.Count
    End If
    ReDim Return_DataE2D(1 To Line, 1 To Row)
    For j = 1 To Line
    For i = 1 To Row
    Return_DataE2D(j, i) = xlSheet.Cells(j, i)
    Next i
    Next j

    xlBook.Close (True)                          '关闭 EXCEL 工作簿
    xlapp.Quit                                          '关闭 EXCEL
    Set xlapp = Nothing                         '释放 EXCEL 对象
    'MsgBox "数据获得"
    FileReadByLineRange = sType
End If
End Function
```

'FileReadByLineRangeNum 文件逐行读取 txt,xlsx,xls 限制读取条数

Rem mvarFileName:用户选择的文件名
Rem Return_DataE1D(j, i) 返回 EXCEL 内容 xlSheet.Cells(j, i)   j=Line   i=Row
Rem Return_DataT(n) 返回 TEXT 内容
Rem BeginRow BeginROW 只在 EXCEL 中生效
Rem 当 BeginROW,BeginLINE 为 0 的时候从头开始
Rem BeginROW,BeginLINE 开始行，开始列，Num 读取的数目

```
Public Function FileReadByLineRangeNum(ByVal mvarFilePath As String, ByVal BeginLine As Long,
_ByVal BeginRow As Long, ByVal Num As Long, ByVal UsedLine As Long, ByVal UsedRow As Long,
_ByRef Return_DataE1D() As Variant, ByRef Return_DataT() As Variant) As String    'text 改个
MULTIPLINE 就可以了
Dim sFile$
Dim sType$, arrTmp
sFile = mvarFilePath '用户选择的文件名
arrTmp = Split(sFile, ".")
                If (IsArray(arrTmp)) Then
                   sType = arrTmp(UBound(arrTmp)) 'sType 就是文件类型名
                   Else
```

```
                MsgBox "选择的文件名错误,或者没有扩展名 Exit Function
FileReadByLineRangeNum！" '或者没有扩展名
                Exit Function
            End If
    If (sType = "txt") Then
                Dim B1 As String, n As Long, m As Long
                n = 0
                If (BeginRow <> 0) Then
                MsgBox "TEXT 文档暂时不支持 ROW,程序继续运行"
                End If

                Open mvarFilePath For Input As #1
                Do While Not EOF(1)
'循环至文件尾.
                Line Input #1, B1
'读入一行数据并将其赋予某变量.
                n = n + 1

                If n >= BeginLine Then
                m = m + 1
                ReDim Preserve Return_DataT(n)
                Return_DataT(m) = B1
                End If

                If (m > Num And Num <> 0) Then
                Exit Do
                End If

                DoEvents
                Loop

                 Dim i As Long
                  If (n < BeginLine + Num) Then
                  For i = n + 1 To BeginLine + Num
                  ReDim Preserve Return_DataT(i)
                  Return_DataT(i) = "Null"
                  Next i
                  End If
                n = 0
                Close #1
                FileReadByLineRangeNum = sType
    End If
If (sType = "xlsx" Or sType = "xls") Then
    'MsgBox "读取数据"
```

```
    Dim l As Integer, j As Integer, k As Integer
    Dim xlapp As Object, xlBook As Object, xlSheet As Object
    Set xlapp = New Excel.Application
    xlapp.Visible = False
    Set xlBook = xlapp.Workbooks.Open(mvarFilePath)      '打开 EXCEL 工作簿
    Set xlSheet = xlBook.Worksheets(1)                                    '打开 EXCEL 工作表
    j = BeginRow
    k = BeginLine
    n = 0
    ReDim Return_DataE1D(1 To Num)
    For i = 1 To Num
    If (j + n <= UsedRow) Then '2016-12-26 号，由 If (j + n <= UsedLine) Then 改来
    Return_DataE1D(i) = xlSheet.Cells(k, j + n)
    'Debug.Print Return_DataE1D(i) & "=" & xlSheet.Cells(k, j + n)
    n = n + 1
    Else
    k = k + 1
    j = 0
    n = 1
    Return_DataE1D(i) = xlSheet.Cells(k, j + n)
    'Debug.Print Return_DataE1D(i) & "=" & xlSheet.Cells(k, j + n)
    n = n + 1
        If (k > UsedLine) Then   '2016-12-26 号，由 If (k+1 > UsedLine) Then 改来
        MsgBox "超出数据排数，请检查。EXIT FUNCTION FileReadByLineRangeNum"
        Exit Function
        End If
    End If
    Next i
    xlBook.Close (True)                          '关闭 EXCEL 工作簿
    xlapp.Quit                                              '关闭 EXCEL
    Set xlapp = Nothing                         '释放 EXCEL 对象
    FileReadByLineRangeNum = sType
End If
End Function
```

'FileReadByLineRangeNumUnRead0 文件逐行读取 txt,xlsx,xls 限制读取条数，Txt 不读取 0,EXCEL 不读取 0

Rem 去除零只限 TXT，EXCEL 0 显示为 0

```
Public Function FileReadByLineRangeNumUnRead0(ByVal mvarFilePath As String, ByVal BeginLine As Long, _
ByVal BeginRow As Long, ByVal Num As Long, ByVal UsedLine As Long, ByVal UsedRow As Long, _
```

```
ByRef Return_DataE1D() As Variant, ByRef Return_DataT() As Variant) As String    'text 改个
MULTIPLINE 就可以了
Dim sFile$
Dim sType$, arrTmp
sFile = mvarFilePath '用户选择的文件名
arrTmp = Split(sFile, ".")
                If (IsArray(arrTmp)) Then
                  sType = arrTmp(UBound(arrTmp)) 'sType 就是文件类型名
                  Else
                  MsgBox "选择的文件名错误,或者没有扩展名 Exit Function
FileReadByLineRangeNumUnRead0！" '或者没有扩展名
                  Exit Function
              End If
      If (sType = "txt") Then
                Dim B1 As String, n As Long, m As Long
                n = 0
                If (BeginRow <> 0) Then
                MsgBox "TEXT 文档暂时不支持 ROW,程序继续运行"
                End If

                  Open mvarFilePath For Input As #1
                  Do While Not EOF(1)
'循环至文件尾.
                  Line Input #1, B1
'读入一行数据并将其赋予某变量.
                  n = n + 1

                  If n >= BeginLine Then

                    If (B1 <> 0) Then '  去除 0 项
                    m = m + 1
                    ReDim Preserve Return_DataT(n)
                    Return_DataT(m) = B1
                    End If
                  End If

                  If (m > Num And Num <> 0) Then
                  Exit Do
                  End If

                  DoEvents
                  Loop

                   Dim i As Long
```

```
                    If (n < BeginLine + Num - 1) Then
                    For i = n + 1 To BeginLine + Num
                    ReDim Preserve Return_DataT(i)
                    Return_DataT(i) = "Null"
                    Next i
                    End If
                n = 0
                Close #1
                FileReadByLineRangeNumUnRead0 = sType
    End If
If (sType = "xlsx" Or sType = "xls") Then
    'MsgBox "读取数据"
    Dim l As Integer, j As Integer, k As Integer
    Dim xlapp As Object, xlBook As Object, xlSheet As Object
    Set xlapp = New Excel.Application
    xlapp.Visible = False
    Set xlBook = xlapp.Workbooks.Open(mvarFilePath)      '打开 EXCEL 工作簿
    Set xlSheet = xlBook.Worksheets(1)                                    '打开 EXCEL 工作
表
    j = BeginRow
    k = BeginLine
    n = 0
    ReDim Return_DataE1D(1 To Num)
    For i = 1 To Num

    If (j + n <= UsedRow) Then
    If (0 <> xlSheet.Cells(k, j + n)) Then
    Return_DataE1D(i) = xlSheet.Cells(k, j + n)
    Else
    Return_DataE1D(i) = xlSheet.Cells(k, j + n)
    End If
    'Debug.Print Return_DataE1D(i) & "=" & xlSheet.Cells(k, j + n)
    n = n + 1
    Else
    k = k + 1
    j = 0
    n = 1
    Return_DataE1D(i) = xlSheet.Cells(k, j + n)
    'Debug.Print Return_DataE1D(i) & "=" & xlSheet.Cells(k, j + n)
    n = n + 1
        If (k > UsedLine) Then '2016-12-26 由(k +1> UsedLine) 改来
        MsgBox "超出数据排数，请检查。EXIT FUNCTION FileReadByLineRangeNumUnRead0"
        Exit Function
        End If
```

```
    End If
    Next i

    xlBook.Close (True)                          '关闭 EXCEL 工作簿
    xlapp.Quit                                   '关闭 EXCEL
    Set xlapp = Nothing                          '释放 EXCEL 对象
    FileReadByLineRangeNumUnRead0 = sType
End If

End Function
```

'FileReadByLineExcelQuick 快速逐行读取 Excel 文件

Rem 现在只返回二维数组
Rem mvarFileName:用户选择的文件名
Rem Return_DataE1D(K) 返回 EXCEL 内容 xlSheet.Cells(j, i) 一维数组
Rem Return_DataE2D(j, i) 返回 EXCEL 内容 xlSheet.Cells(j, i) 二维数组

```
Public Function FileReadByLineExcelQuick(ByVal mvarFilePath As String, ByRef Return_DataE1D()
As Variant, _
ByRef Return_DataE2D() As Variant) As Boolean
'Return_DataE1D 是去除空值的 1 维数组
'Return_DataE2D 是按照 EXCEL 表格内容排列的 2 维数组
'On Error GoTo A1
Dim sFile$
Dim sType$, arrTmp
sFile = mvarFilePath '用户选择的文件名
arrTmp = Split(sFile, ".")
                If IsArray(arrTmp) Then
                   sType = arrTmp(UBound(arrTmp)) 'sType 就是文件类型名
                   Else
                   MsgBox "选择的文件名错误,或者没有扩展名 Exit Function FileReadByLineExcel !
" '或者没有扩展名
                   Exit Function
               End If
 If sType = "xlsx" Or sType = "xls" Then
    'MsgBox "读取数据"
    Dim i As Long, l As Long, j As Long, k As Long
    k = 0
    Dim xlapp As Object, xlBook As Object, xlSheet As Object
    Set xlapp = Excel.Application ' 2018-5-4 由 Set xlapp =New Excel.Application 改成
    xlapp.Visible = False
    Set xlBook = xlapp.Workbooks.Open(mvarFilePath)      '打开 EXCEL 工作簿
    Set xlSheet = xlBook.Worksheets(1)                                    '打开 EXCEL 工作
```

```
表
    Dim xhang As Long: Dim ylie As Long
    xhang = xlSheet.UsedRange.Rows.Count
    ylie = xlSheet.UsedRange.Columns.Count
    ReDim Return_DataE2D(1 To xhang, 1 To ylie)
    Debug.Print mvarFilePath
    Debug.Print "XHang=" & xhang & " YLie=" & ylie
    Dim Data As Variant
    Data = xlapp.Range(Cells(1, 1), Cells(xhang, ylie))     '关键代码这个要在 EXCLE 中信任 VBA
宏
    For j = 1 To xhang
    For i = 1 To ylie
    Return_DataE2D(j, i) = Data(j, i)
            If xlSheet.Cells(j, i) <> "" Then
                k = k + 1
                ReDim Preserve Return_DataE1D(k)
            End If
    Next i
    Next j
FileReadByLineExcelQuick = True
    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Rem  结束进程的方法
    'xlBook.SaveAs mvarFilePath
    'If Dir(mvarFilePath) <> "" Then
    'Kill mvarFilePath
    'End If
     '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
a1:
    Set xlSheet = Nothing '2018-7-20
    xlBook.Close (True)                          '关闭 EXCEL 工作簿
    Set xlBook = Nothing '2018-7-20
    xlapp.Quit                                              '关闭 EXCEL
    Set xlapp = Nothing                          '释放 EXCEL 对象
    'MsgBox "数据获得"
End If
End Function
```

'FileReadByLineExcelRange 逐行读取 Excel 文件限制范围

Rem 返回 EXCEL 使用的行数和列数

```
Public Function FileReadByLineExcelRange(ByVal mvarFilePath As String, ByRef Row As Long,
ByRef Line As Long) As Boolean
Dim sFile$
```

```
Dim sType$, arrTmp
sFile = mvarFilePath '用户选择的文件名
arrTmp = Split(sFile, ".")
                If (IsArray(arrTmp)) Then
                   sType = arrTmp(UBound(arrTmp)) 'sType 就是文件类型名
                   Else
                   MsgBox "选择的文件名错误,或者没有扩展名 Exit Function
FileReadByLineRangeNumUnRead0！" '或者没有扩展名
                   Exit Function
               End If
      If (sType = "xlsx" Or sType = "xls") Then
      Dim l As Integer, j As Integer
      Dim xlapp As Object, xlBook As Object, xlSheet As Object
      Set xlapp = New Excel.Application
      xlapp.Visible = False
      Set xlBook = xlapp.Workbooks.Open(mvarFilePath)       '打开 EXCEL 工作簿
      Set xlSheet = xlBook.Worksheets(1)                                    '打开 EXCEL 工作
表
      Line = xlSheet.UsedRange.Rows.Count
      Row = xlSheet.UsedRange.Columns.Count
      xlBook.Close (True)                          '关闭 EXCEL 工作簿
      xlapp.Quit                                          '关闭 EXCEL
      Set xlapp = Nothing                          '释放 EXCEL 对象
      FileReadByLineExcelRange = True
      Else
      FileReadByLineExcelRange = False
      End If
End Function
```

'FileReadByLineTexT1D2D 读取并返回 Text 内容一维数据、二维数据

Rem 读取 Txt

Rem ReturnDataT(n)返回 TEXT 内容

```
Public Function FileReadByLineTexT1D2D(ByVal FilePath As String, ByRef ReturnData1D() As
Variant, _
ByRef ReturnData2D() As Variant, ByRef ReturnData2DTransposition() As Variant) As Boolean
'text 改个 MULTIPLINE 就可以了
Dim file As String, FileType As String, FileArray() As String
Dim ReturnDataT() As Variant
Dim ReturnLine As Long
'On Error GoTo ErrHandle
```

```
file = FilePath '用户选择的文件名
FileArray = Split(file, ".")
                If IsArray(FileArray) Then
                   FileType = FileArray(UBound(FileArray)) 'sType 就是文件类型名
                   Else
                   MsgBox "选择的文件名错误,或者没有扩展名 Exit Function
FileReadByLineTexT2D！" '或者没有扩展名
                   Exit Function
               End If
SubTxt FileType, FilePath, ReturnDataT, ReturnLine, FileReadByLineTexT1D2D
SubSplitTxtLine ReturnDataT, ReturnData1D, ReturnData2D, ReturnLine,
ReturnData2DTransposition
FileReadByLineTexT1D2D = True
Exit Function
ErrHandle:
MsgBox "FileRead007:FileReadByLineTexT2D  出错!"
FileReadByLineTexT1D2D = False
Resume Next
End Function
```

Rem SubSplitTxtLine 的子程序

Rem 找到最长的数据条

```
Private Function FindLongestLine(ByVal n As Long, ByRef ReturnDataT() As Variant) As Long
Dim Num() As Long, a() As String, p As Long, i As Long
For i = 1 To n
a = Split(ReturnDataT(i), "#")
p = UBound(a) - LBound(a)
ReDim Preserve Num(i)
Num(i) = p
Next i
FindLongestLine = Array1DMax(Num)
End Function
```

Rem 找到最大值

```
Public Function Array1DMax(ByRef Data() As Long) As Long
Dim i As Long
Dim cData() As Long
cData = Data
Rem   防止数组只有一个数字
If LBound(Data) = UBound(Data) Then
Array1DMax = Data(LBound(Data))
End If
Rem   找到数组的最大值赋给 Array1DMax
For i = LBound(cData()) To UBound(cData())
      If i < UBound(cData()) Then
            If cData(i) > cData(i + 1) Then
```

```
            Array1DMax = cData(i)
            cData(i + 1) = cData(i)
            Else
            Array1DMax = cData(i + 1)
            End If
      End If
Next i
End Function
```

14. FileWrite

详细说明：

函数或方法名称	作用
WhiteInExcel2007	把一维数据写入到 Excel 中
ArrayRangeD	数组的维数
ArrayRange	数组的维数
CreateNewExcelFile	创建新的 Excel 文件
WhiteInExcel20072D	二维数据写入到 Excel
WhiteInExcel20072DLng	二维数据写入到 Excel 数据类型 Long
WhiteInTxt	二维数据分行写入 TXT，以逗号分隔
WhiteInTxtLongTransposition	二维数据转置分行写入 TXT，以#号分隔
WhiteInTxtLong	二维数据分行写入 TXT，以#号分隔
Array1DRemoveOneLine	一维数组移除一个
Array1DSort	一维数据进行大小排列
SubSort	Array1DSort 子程序
ReadLine	读取 Txt 行内容
WhiteInExcel2007AddLine	添加内容在 Excel 最后一行后
WhiteInExcel2007AddLineByNum	添加内容在 Excel 特定行号后
WhiteInExcel2007AddLineD	添加内容在 Excel 最后一行数据类型 double
WhiteInExcel2007AddLineUnuse	如果编号存在则替换对应编号内容,没有编号则添加在最后一行
WhiteInExcel2007AddLineUnuseClearRows	如果编号存在则替换对应编号内容,没有编号则添加在最后一行。由于数据变长，首先删除此行再写入
WhiteInTxtAddLine	在 Txt 最后追加一排
SubReadTxt	WhiteInTxtAddLine 的子程序
WhiteInTxtAddLineUnuse	果编号存在则替换对应编号内容,没有编号则添加在最后一行
SubReadTxtFind	WhiteInTxtAddLineUnuse 的子程序
WhiteInTxtBySerial	文件 Txt 数据按照排头序号从小到大重写
GetDataLine	WhiteInTxtBySerial 子程序
GetDataSerial	WhiteInTxtBySerial 子程序
WhiteInTxtRemoveEmpty	写入 Txt 去除掉空项
WhiteInTxtTitle	修改 Txt 表头
WriteLine	按行编号写入一行内容替换原来行的内容

代码：

```
Option Explicit
Option Base 1
'*************************************************************************************
'名称 = FileWrite.Cls
'作者 = 张宏
'最后修改时间 = 2018 年 12 月 5 日
'简介= 文件数据写入
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=1.需要工程->引用->MicroSoft Excel 12.0 Object Library
'参考文献:
'操作历史:
'*************************************************************************************
```

'WhiteInExcel2007 把一维数据写入到 Excel 中

Rem WhiteInTxt 2018-12-4 加入
Rem WhiteInTxtLong 2018-12-5 加入
Rem excel 2003 工作表最大有 **2^16=65536** 行,**2^8=256** 列
Rem excel 2007 和 **excel 2010** 最大有 **2^20=1048576** 行,**2^14=16384** 列

```
Public Function WhiteInExcel2007(ByRef Data() As Variant, ByVal ExcelFileName As String, _
ByVal Worksheet As Integer) As Boolean    '
    Dim i As Long, l As Long
    Dim xlApp As Object, xlBook As Object, xlSheet As Object
    Set xlApp = New Excel.Application
    xlApp.Visible = False                                          '设置 EXCEL 不可见
    'Set xlBook = xlApp.Workbooks.Open(App.Path & "\data2007.xlsx") '打开 EXCEL 工作簿
    Set xlBook = xlApp.Workbooks.Open(ExcelFileName)
    Set xlSheet = xlBook.Worksheets(Worksheet)                     '打开 EXCEL
工作表
    Dim n As Long
    n = 0
    i = ArrayRange(Data)
    If i = 1 Then
    For l = LBound(Data) To UBound(Data)
    n = n + 1
    xlSheet.Cells(n, 1).Value = Data(l)
    Next
    End If
    n = 0
```

```
    If i <> 1 Then
    MsgBox "1 维以上还未编写"
    End If

    'For l = 1 To y3 - 3
    'For i = 1 To x3 - 3
    'xlSheet.Cells(l, i).Value = ColOut(0, i, l)
    'DoEvents
    'Next
    'Next
    Set xlSheet = Nothing '2018-7-20
    xlBook.Close (True)                    '关闭 EXCEL 工作簿
    Set xlBook = Nothing '2018-7-20
    xlApp.Quit                                    '关闭 EXCEL
    Set xlApp = Nothing                    '释放 EXCEL 对象

    MsgBox "完成写入"
End Function
```

' ArrayRangeD 数组的维数

```
Public Function ArrayRangeD(ByRef Data() As Double) As Integer
Dim i As Integer
Dim Ret As Integer
Dim ErrF As Boolean

ErrF = False
On Error GoTo ErrHandle
'判断代入的参数是否为数组
If Not IsArray(Data) Then
ArrayRangeD = -1
Exit Function
End If
'VB 中数组最大为 60
For i = 1 To 60
'用 UBound 函数判断某一维的上界，如果大数组的实际维数时产生超出范围错误，
' 此时我们通过 Resume Next  来捕捉错这个错误
Ret = UBound(Data, i)
If ErrF Then Exit For
Next i
'最后返回
ArrayRangeD = Ret
```

```
Exit Function
ErrHandle:
Ret = i - 1
ErrF = True
Resume Next

End Function
```

'ArrayRange 数组的维数

```
Public Function ArrayRange(ByRef Data() As Variant) As Integer
Dim i As Integer
Dim Ret As Integer
Dim ErrF As Boolean

ErrF = False
On Error GoTo ErrHandle
'判断代入的参数是否为数组
If Not IsArray(Data) Then
ArrayRange = -1
Exit Function
End If
'VB 中数组最大为 60
For i = 1 To 60
'用 UBound 函数判断某一维的上界，如果大数组的实际维数时产生超出范围错误，
' 此时我们通过 Resume Next 来捕捉错这个错误
Ret = UBound(Data, i)
If ErrF Then Exit For
Next i
'最后返回
ArrayRange = Ret

Exit Function
ErrHandle:
Ret = i - 1
ErrF = True
Resume Next

End Function
```

'CreateNewExcelFile 创建新的 Excel 文件

Rem 创建 Excel 文件

```
Public Function CreateNewExcelFile(ByVal ExcelFileNameAndPath As String, ByVal
NewWorksheetName As String)
    Dim xlApp As New Excel.Application
    Dim xlBook As New Excel.Workbook
    Dim xlSheet As New Excel.Worksheet
Set xlBook = xlApp.Workbooks.Add
If NewWorksheetName <> "" Then
Worksheets.Add(After:=Sheets(Sheets.Count)).Name = NewWorksheetName
End If
Set xlSheet = xlBook.Worksheets(1)
xlBook.SaveAs (ExcelFileNameAndPath) '操作后另存 目录自己安排
xlBook.Close
xlApp.Application.Quit
Set xlApp = Nothing
End Function
```

'WhiteInExcel20072D 二维数据写入到 Excel

Rem 去除最顶一排和最左一排

```
Public Function WhiteInExcel20072D(ByRef Data() As Variant, ByVal ExcelFileName As String, _
ByVal Worksheet As Integer) As Boolean    '
    Dim i As Long, l As Long
    Dim xlApp As Object, xlBook As Object, xlSheet As Object
    Set xlApp = New Excel.Application
    xlApp.Visible = False                                          '设置 EXCEL 不可见
    'Set xlBook = xlApp.Workbooks.Open(App.Path & "\data2007.xlsx") '打开 EXCEL 工作簿
    Set xlBook = xlApp.Workbooks.Open(ExcelFileName)
    Set xlSheet = xlBook.Worksheets(Worksheet)                     '打开 EXCEL
工作表
    Dim m As Long
    i = ArrayRange(Data)
    If i = 2 Then
        For m = LBound(Data, 2) To UBound(Data, 2)
            For l = LBound(Data, 1) To UBound(Data, 1)
            xlSheet.Cells(l + 1, m + 1).Value = Data(l, m)
            Next l
        Next m
    End If
    If i <> 2 Then
```

```
    MsgBox "2 维以外还未编写"
    End If
    Set xlSheet = Nothing '2018-7-20
    xlBook.Close (True)                         '关闭 EXCEL 工作簿
    Set xlBook = Nothing '2018-7-20
    xlApp.Quit                                          '关闭 EXCEL
    Set xlApp = Nothing                         '释放 EXCEL 对象

    MsgBox "完成写入"
End Function
```

'WhiteInExcel20072DLng 二维数据写入到 Excel 数据类型 Long

Rem 去除最顶一排和最左一排

```
Public Function WhiteInExcel20072DLng(ByRef Data() As Long, ByVal ExcelFileName As String, _
ByVal Worksheet As Integer) As Boolean    '
    Dim i As Long, l As Long
    Dim xlApp As Object, xlBook As Object, xlSheet As Object
    Set xlApp = New Excel.Application
    xlApp.Visible = False                                                   '设置 EXCEL 不可见
    'Set xlBook = xlApp.Workbooks.Open(App.Path & "\data2007.xlsx") '打开 EXCEL 工作簿
    Set xlBook = xlApp.Workbooks.Open(ExcelFileName)
    Set xlSheet = xlBook.Worksheets(Worksheet)                              '打开 EXCEL
工作表
    Dim m As Long
    'I = ArrayRange_Lng(Data)

    i = 2
    If i = 2 Then
        For m = LBound(Data, 2) To UBound(Data, 2)
            For l = LBound(Data, 1) To UBound(Data, 1)
            xlSheet.Cells(m + 1, l + 1).Value = Data(l, m)
            Next l
        Next m
    End If
    If i <> 2 Then
    MsgBox "2 维以外还未编写"
    End If
    Set xlSheet = Nothing '2018-7-20
     xlBook.Close (True)                        '关闭 EXCEL 工作簿
    Set xlBook = Nothing '2018-7-20
    xlApp.Quit                                          '关闭 EXCEL
    Set xlApp = Nothing                         '释放 EXCEL 对象
```

```
End Function
```

'WhiteInTxt 二维数据分行写入 TXT，以逗号分隔

Rem 把 2 维数据分行写入 TXT，以逗号分隔。

```
Public Function WhiteInTxt(ByRef Data2D() As Variant, ByVal TxtFileName As String) As Boolean
Dim i As Long, l As Long
Dim Str As Variant
Open TxtFileName For Output As #1
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
    Str = Str & Data2D(i, l) & "#"
    Next l
    Str = Str & vbCrLf
Next i
Str = Mid(Str, 1, Len(Str) - 2)
Print #1, Str
Close #1
End Function
```

'WhiteInTxtLongTransposition 二维数据转置分行写入 TXT，以#号分隔

Rem 把 2 维数据分行写入 TXT，以#号分隔。

```
Public Function WhiteInTxtLongTransposition(ByRef Data2DTransposition() As Long, ByVal
TxtFileName As String) As Boolean
Dim i As Long, l As Long
Dim Str As Variant
Dim Data2D() As Long
ReDim Data2D(LBound(Data2DTransposition, 2) To UBound(Data2DTransposition, 2),
LBound(Data2DTransposition, 1) To UBound(Data2DTransposition, 1))
Open TxtFileName For Output As #1

For i = LBound(Data2DTransposition, 1) To UBound(Data2DTransposition, 1)
    For l = LBound(Data2DTransposition, 2) To UBound(Data2DTransposition, 2)
    Data2D(l, i) = Data2DTransposition(i, l)
    Next l
Next i
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
    Str = Str & Data2D(i, l) & "#"
```

```
    Next l
    Str = Str & vbCrLf
Next i
Str = Mid(Str, 1, Len(Str) - 2)
Print #1, Str
Close #1
End Function
```

'WhiteInTxtLong 二维数据分行写入 TXT，以#号分隔

Rem 把 2 维数据分行写入 TXT，以#分隔。

```
Public Function WhiteInTxtLong(ByRef Data2D() As Long, ByVal TxtFileName As String) As Boolean
Dim i As Long, l As Long
Dim Str As Variant
Open TxtFileName For Output As #1
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
    Str = Str & Data2D(i, l) & "#"
    Next l
    Str = Str & vbCrLf
Next i
Str = Mid(Str, 1, Len(Str) - 2)
Print #1, Str
Close #1
End Function
```

'WhiteInTxtLongTranspositionAddTitle 二维数据转置分行写入 TXT，以逗号分隔，增加标题文本

Rem 把 2 维数据分行写入 TXT，以逗号分隔。

```
Public Function WhiteInTxtLongTranspositionAddTitle(ByRef Data2DTransposition() As Long,
ByVal TxtFileName As String) As Boolean
Dim i As Long, l As Long
Dim Str As Variant
Dim Data2D() As Long
ReDim Data2D(LBound(Data2DTransposition, 2) To UBound(Data2DTransposition, 2),
LBound(Data2DTransposition, 1) To UBound(Data2DTransposition, 1))
Open TxtFileName For Output As #1
For i = LBound(Data2DTransposition, 1) To UBound(Data2DTransposition, 1)
    For l = LBound(Data2DTransposition, 2) To UBound(Data2DTransposition, 2)
    Data2D(l, i) = Data2DTransposition(i, l)
```

```
        Next l
Next i

ReDim Data2DTitle(LBound(Data2D, 1) To UBound(Data2D, 1) + 1, LBound(Data2D, 2) To
UBound(Data2D, 2) + 1)
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
        For l = LBound(Data2D, 2) To UBound(Data2D, 2)
        Data2DTitle(i + 1, l + 1) = 0
        Next l
Next i
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
        For l = LBound(Data2D, 2) To UBound(Data2D, 2)
        Data2DTitle(i + 1, l + 1) = Data2D(i, l)
        Next l
Next i

For i = LBound(Data2DTitle, 1) To UBound(Data2DTitle, 1)
        For l = LBound(Data2DTitle, 2) To UBound(Data2DTitle, 2)
        Str = Str & Data2DTitle(i, l) & "#"
        Next l
        Str = Str & vbCrLf
Next i
Str = Mid(Str, 1, Len(Str) - 2)
Print #1, Str
Close #1
End Function
```

'Array1DRemoveOneLine 一维数组移除一个

Rem 从一维数组中剔除一条数据
Rem 例如：data(-1)=1,data(0)=2,data(1)=3,data(2)=4,data(3)=5 去除 1 则是 1,2,4,5。这里的 1 是指 data(1)中的 1
Rem Data()=原始数据
Rem RemoveWhich=数组标号，数据中剔除的数据
Rem ReturnData()=返回剔除后的数据，返回数组以 1 为底
Rem Array1DRemoveOneNum=返回剔除数据

```
Public Function Array1DRemoveOneLine(ByRef Data() As String, _
ByVal RemoveWhich As Long, ByRef ReturnData() As String) As String
Dim i As Long
Dim cData() As String
Dim n As Long
If (RemoveWhich < LBound(Data) Or RemoveWhich > UBound(Data)) Then
MsgBox "去除位置超出数组范围，程序跳出 Exit Function Array1DRemoveOneNum"
```

```
Exit Function
End If
If LBound(Data) = UBound(Data) Then
MsgBox "数组只有一个空间，程序跳出  Exit Function Array1DRemoveOneNum"
Exit Function
End If

cData = Data
n = 0
For i = LBound(Data) To UBound(Data)
    If i = RemoveWhich Then
    Array1DRemoveOneLine = Data(i)
    Else
    n = n + 1
    ReDim Preserve ReturnData(n)
    ReturnData(n) = cData(i)
    'Debug.Print "r(" & n & ")=" & "cdata(" & i & ")=" & cData(i)
    End If
Next i
End Function
```

'Array1DSort 一维数据进行大小排列

Rem 对数据进行大小排列
Rem Data()=原始数据
Rem ReturnSortLtoU()=返回数据从小到大排列
Rem ReturnSortUtoL()=返回数据从大到小排列

```
Public Function Array1DSort(ByRef Data() As Double, ByRef ReturnSortLtoU() As Double, _
ByRef ReturnSortUtoL() As Double) As Boolean
Dim i As Long, l As Long, T As Double
Dim cData()    As Double
cData = Data
For l = LBound(cData) To UBound(cData)
    For i = LBound(cData) To (UBound(cData) - l)
        If i = UBound(cData) Then
        i = LBound(cData)
        l = l + 1
            If l = UBound(cData) Then
            Exit For
            End If
        End If
        If cData(i) > cData(i + 1) Then
        T = cData(i)
```

```
            cData(i) = cData(i + 1)
            cData(i + 1) = T
            End If
      'Debug.Print "cData(" & i & ")= " & cData(i) & "    *****      " & l
      Next i
Next l
SubSort cData, ReturnSortLtoU, ReturnSortUtoL
End Function
```

Rem WhiteInTxtBySerial 子程序

Rem ReturnLine=总条数

Rem ReturnDataT()=返回获得的条数据

```
Private Sub GetDataLine(ByVal TxtFileName As String, ByRef ReturnLine As Long, ByRef
ReturnDataT() As String)
Dim n As Long, B1 As String
Open TxtFileName For Input As #1
                Do While Not EOF(1)                                    '循环至文件尾.
                Line Input #1, B1                                      '读入一行数据并将其赋予某
变量.
                n = n + 1
                ReDim Preserve ReturnDataT(n)
                ReturnDataT(n) = B1
                DoEvents
                Loop
                ReturnLine = n '总排数
Close #1
End Sub
```

Rem WhiteInTxtBySerial 子程序

Rem 获得数据的编号

```
Private Sub GetDataSerial(ByVal TxtFileName As String, ByVal n As Long, ByRef ReturnDataT() As
String, _
ByRef Num() As Double, ByRef RLtoU() As Double, ByRef Str1() As Long)
Dim i As Long, r As Long, O As Long
Dim Str As Variant
Dim p As Variant
Dim RUtoL() As Double
Dim q() As String
Open TxtFileName For Output As #1
      For Each p In ReturnDataT
      q = Split(p, "#")
      'Debug.Print q(0)
      'Debug.Print q(1)
      i = i + 1
      If IsNumeric(q(0)) = True Then
      r = r + 1
```

```
        ReDim Preserve Num(r)
        Num(r) = CDbl(q(0))
        Else
        O = O + 1
        ReDim Preserve Str1(O)
        Str1(O) = i
        End If
        Next
Array1DSort Num, RLtoU, RUtoL
End Sub
```

'ReadLine 读取 Txt 行内容

Rem 读取行内容

Rem TxtFileName=文件名全称

Rem Lines=行数

```
Public Function ReadLine(ByVal TxtFileName As String, ByRef Lines As Long) As String
Dim ReturnDataT() As Variant
Open TxtFileName For Input As #1
                Do While Not EOF(1)                                     '循环至文件尾.
                Line Input #1, B1                                       '读入一行数据并将其赋予某
变量.
                n = n + 1
                ReDim Preserve ReturnDataT(n)
                ReturnDataT(n) = B1
                DoEvents
                If Lines = n Then: Exit Do '读取到指定行数退出
                Loop
                ReturnLine = n '总排数
Close #1
ReadLine = ReturnDataT(Line)
End Function
```

Rem WhiteInTxtAddLine 的子程序

Rem 读取 TXT 文件

```
Private Sub SubReadTxt(ByVal FilePath As String, ByRef ReturnDataTxt As String, _
ByRef ReturnLine As Long)
Dim B1 As String, n As Long
                n = 0
                Open FilePath For Input As #1
                Do While Not EOF(1)                                     '循环至文件尾.
                Line Input #1, B1                                       '读入一行数据并将其赋予某
变量.
                n = n + 1
```

```
                ReturnDataTxt = ReturnDataTxt + B1 + vbCrLf
                DoEvents
                Loop
                ReturnLine = n '总排数
                Close #1
End Sub
```

Rem WhiteInTxtAddLineUnuse 的子程序

Rem 读取 TXT 文件

Rem Data1DS=替换这行内容

Rem FilePath=文件地址

Rem ReturnDataTxt=修改后的内容

Rem ReturnLine=总排数

```
Public Function SubReadTxtFind(ByVal Num As Long, ByVal Data1DS As String, ByVal FilePath As
String, _
ByRef ReturnDataTxt As String, ByRef ReturnLine As Long) As Boolean
Dim B1 As String, n As Long, a As Variant
                n = 0
                Debug.Print FilePath
                Open FilePath For Input As #1
                Do While Not EOF(1)                         '循环至文件尾.
                    Line Input #1, B1                       '读入一行数据并将其赋予某变量.
                    n = n + 1
                    If B1 = "" Then
                    Close #1 '退出 Function 前写入
                    Open FilePath For Output As #2
                    ReturnDataTxt = Mid(ReturnDataTxt, 1, Len(ReturnDataTxt) - 2)
                    Print #2, ReturnDataTxt
                    Close #2 '退出 Function 前写入
                    Debug.Print "跳出位置:" + FilePath
                    Debug.Print "跳出内容:" + ReturnDataTxt
                    Exit Function
                    End If
                    a = Split(B1, "#")
                    If a(0) = CStr(Num) Then '  找到目标编号
                    SubReadTxtFind = True
                    ReturnDataTxt = ReturnDataTxt + Data1DS + vbCrLf   '找到目标编号替换
                    Else
                    ReturnDataTxt = ReturnDataTxt + B1 + vbCrLf   '未找到目标编号不替换
                    End If
                    DoEvents
                Loop
                ReturnLine = n                                          '总排数
                'Debug.Print ReturnDataTxt
                Close #1
```

```
        Open FilePath For Output As #2
        ReturnDataTxt = Mid(ReturnDataTxt, 1, Len(ReturnDataTxt) - 2)
        Print #2, ReturnDataTxt
        Close #2
End Function
```

Rem Array1DSort 子程序
Rem 给出排序数组
Rem cData()=传入数据
Rem ReturnSortLtoU()=返回数据从小到大排列
Rem ReturnSortUtoL()=返回数据从大到小排列

```
Private Function SubSort(ByRef cData() As Double, ByRef ReturnSortLtoU() As Double, _
ByRef ReturnSortUtoL() As Double)
ReDim ReturnSortLtoU(LBound(cData) To UBound(cData))
ReDim ReturnSortUtoL(LBound(cData) To UBound(cData))
Dim i As Long, l As Long
For i = LBound(cData) To UBound(cData)
    'Debug.Print "cData(" & i & ")= " & cData(i) & "    PaiXu up"
    ReturnSortLtoU(i) = cData(i)
Next i
l = LBound(cData) - 1
For i = UBound(cData) To LBound(cData) Step -1
    'Debug.Print "cData(" & i & ")= " & cData(i) & "    PaiXu down"
    l = l + 1
    ReturnSortUtoL(l) = cData(i)
Next i
End Function
```

'WhiteInExcel2007AddLine 添加内容在 Excel 最后一行后

Rem 添加内容在最后一行后
Rem 写入 Excel
Rem Data()=输入数据
Rem ExcelFileName=文件名称地址
Rem Worksheet=要写入的工作表

```
Public Function WhiteInExcel2007AddLine(ByRef Data() As Variant, ByVal ExcelFileName As String,
_ByVal Worksheet As Integer) As Boolean    '
    Dim i As Long, l As Long
    Dim Xhang As Long
    Dim xlApp As Object, xlBook As Object, xlSheet As Object
    Set xlApp = New Excel.Application
    xlApp.Visible = False
'设置 EXCEL 不可见
    Set xlBook = xlApp.Workbooks.Open(ExcelFileName)
```

```
    Set xlSheet = xlBook.Worksheets(Worksheet)                    '打开 EXCEL 工作表

    Xhang = xlSheet.UsedRange.Rows.Count

    Dim n As Long
    For l = LBound(Data) To UBound(Data)
    n = n + 1
    xlSheet.Cells(Xhang + 1, n).Value = Data(l)
    Next l

    Set xlSheet = Nothing
    xlBook.Close (True)
'关闭 EXCEL 工作簿
    Set xlBook = Nothing
    xlApp.Quit
'关闭 EXCEL
    Set xlApp = Nothing
'释放 EXCEL 对象
        'MsgBox "完成写入"
End Function
```

'WhiteInExcel2007AddLineByNum 添加内容在 Excel 特定行号后

Rem 添加内容在特定行号后
Rem 写入 Excel
Rem Num = 从多少行写入
Rem Data()=输入数据
Rem ExcelFileName=文件名称地址
Rem Worksheet=要写入的工作表

```
Public Function WhiteInExcel2007AddLineByNum(ByVal Num As Long, ByRef Data() As Variant, _
ByVal ExcelFileName As String, ByVal Worksheet As Integer) As Boolean    '
    Dim i As Long, l As Long
    Dim Xhang As Long
    Dim xlApp As Object, xlBook As Object, xlSheet As Object
    Set xlApp = New Excel.Application
    xlApp.Visible = False
'设置 EXCEL 不可见
    Set xlBook = xlApp.Workbooks.Open(ExcelFileName)
    Set xlSheet = xlBook.Worksheets(Worksheet)                    '打开 EXCEL 工作表

    Dim n As Long
    For l = LBound(Data) To UBound(Data)
    n = n + 1
```

```
    xlSheet.Cells(Num + 1, n).Value = Data(l)
    Next l

    Set xlSheet = Nothing
    xlBook.Close (True)
'关闭 EXCEL 工作簿
    Set xlBook = Nothing
    xlApp.Quit
'关闭 EXCEL
    Set xlApp = Nothing
'释放 EXCEL 对象
        'MsgBox "完成写入"
End Function
```

'WhiteInExcel2007AddLineD 添加内容在 Excel 最后一行数据类型 double

Rem 添加内容在最后一行后
Rem 写入 Excel
Rem Data()=输入数据
Rem ExcelFileName=文件名称地址
Rem Worksheet=要写入的工作表

```
Public Function WhiteInExcel2007AddLineD(ByRef Data() As Double, ByVal ExcelFileName As
String, _
ByVal Worksheet As Integer) As Boolean    '
    Dim i As Long, l As Long
    Dim Xhang As Long
    Dim xlApp As Object, xlBook As Object, xlSheet As Object
    Set xlApp = New Excel.Application
    xlApp.Visible = False
'设置 EXCEL 不可见
    Set xlBook = xlApp.Workbooks.Open(ExcelFileName)
    Set xlSheet = xlBook.Worksheets(Worksheet)                    '打开 EXCEL 工作表

    Xhang = xlSheet.UsedRange.Rows.Count

    Dim n As Long
    For l = LBound(Data) To UBound(Data)
    n = n + 1
    xlSheet.Cells(Xhang + 1, n).Value = Data(l)
    Next l
```

```
    Set xlSheet = Nothing
    xlBook.Close (True)
'关闭 EXCEL 工作簿
    Set xlBook = Nothing
    xlApp.Quit
'关闭 EXCEL
    Set xlApp = Nothing
'释放 EXCEL 对象
        'MsgBox "完成写入"

End Function
```

'WhiteInExcel2007AddLineUnuse 如果编号存在则替换对应编号内容,没有编号则添加在最后一行

Rem 添加内容在最后一行后,如果存在则替换。
Rem Num=编号
Rem 写入 Excel
Rem Data()=输入数据
Rem ExcelFileName=文件名称地址
Rem Worksheet=要写入的工作表

```
Public Function WhiteInExcel2007AddLineUnuse(ByVal Num As Long, ByRef Data() As Variant, _
ByVal ExcelFileName As String, ByVal Worksheet As Integer) As Boolean    '
    Dim i As Long, l As Long
    Dim Xhang As Long
    Dim xlApp As Object, xlBook As Object, xlSheet As Object
    Set xlApp = New Excel.Application
    xlApp.Visible = False
'设置 EXCEL 不可见
    Set xlBook = xlApp.Workbooks.Open(ExcelFileName)
    Set xlSheet = xlBook.Worksheets(Worksheet)                    '打开 EXCEL 工作表

    Dim n As Long
    Dim Find As Boolean
    Xhang = xlSheet.UsedRange.Rows.Count

    For i = 1 To Xhang
    'Debug.Print xlSheet.Cells(i, 1).Value
        If Num = xlSheet.Cells(i, 1).Value Then
            Find = True
            For l = LBound(Data) To UBound(Data)
            n = n + 1
```

```
                xlSheet.Cells(i, n).Value = Data(l)
                'Debug.Print xlSheet.Cells(i, N).Value
                Next l
            End If
        Next i

                If Find = False Then
                    For l = LBound(Data) To UBound(Data)
                    n = n + 1
                    xlSheet.Cells(Xhang + 1, n).Value = Data(l)
                    'Debug.Print Data(l)
                    Next l
                End If

        Set xlSheet = Nothing
        xlBook.Close (True)
'关闭 EXCEL 工作簿
        Set xlBook = Nothing
        xlApp.Quit
'关闭 EXCEL
        Set xlApp = Nothing
'释放 EXCEL 对象
        'MsgBox "完成写入"

End Function
```

'WhiteInExcel2007AddLineUnuseClearRows 如果编号存在则替换对应编号内容,没有编号则添加在最后一行。由于数据变长，首先删除此行再写入

Rem 添加内容在最后一行后,如果存在则替换，删除行后写入（删除原因是变长）
Rem Num=写入行号
Rem 写入 Excel
Rem Data()=输入数据
Rem ExcelFileName=文件名称地址
Rem Worksheet=要写入的工作表

```
Public Function WhiteInExcel2007AddLineUnuseClearRows(ByVal Num As Long, ByRef Data() As
Variant, _
ByVal ExcelFileName As String, ByVal Worksheet As Integer) As Boolean    '
    Dim i As Long, l As Long
    Dim Xhang As Long
```

```
    Dim xlApp As Object, xlBook As Object, xlSheet As Object
    Set xlApp = New Excel.Application
    xlApp.Visible = False
'设置 EXCEL 不可见
    Set xlBook = xlApp.Workbooks.Open(ExcelFileName)
    Set xlSheet = xlBook.Worksheets(Worksheet)                    '打开 EXCEL 工作表

    Dim n As Long
    Dim Find As Boolean
    Xhang = xlSheet.UsedRange.Rows.Count

    For i = 1 To Xhang
    'Debug.Print xlSheet.Cells(i, 1).Value
        If Num = xlSheet.Cells(i, 1).Value Then
            Find = True
            xlSheet.Rows(i).Delete '由于变长，首先删除此行再写入
            For l = LBound(Data) To UBound(Data)
            n = n + 1
            xlSheet.Cells(i, n).Value = Data(l)
            'Debug.Print xlSheet.Cells(i, N).Value
            Next l

        End If
    Next i

            If Find = False Then
                For l = LBound(Data) To UBound(Data)
                n = n + 1
                xlSheet.Cells(Xhang + 1, n).Value = Data(l)
                'Debug.Print Data(l)
                Next l
            End If

    Set xlSheet = Nothing
    xlBook.Close (True)
'关闭 EXCEL 工作簿
    Set xlBook = Nothing
    xlApp.Quit
'关闭 EXCEL
    Set xlApp = Nothing
'释放 EXCEL 对象
    'MsgBox "完成写入"

End Function
```

'WhiteInTxtAddLine 在 Txt 最后追加一排

Rem 在最后追加一排
Rem 把 **2** 维数据分行写入 **TXT**，**#**以分隔。
Rem Data2D()=输入数据
Rem TxtFileName=文件名称地址

```
Public Function WhiteInTxtAddLine(ByRef Data1D() As Variant, ByVal TxtFileName As String) As
Boolean
Dim i As Long, l As Long
Dim Str As Variant
Dim OldData As String
Dim Line As Long
SubReadTxt TxtFileName, OldData, Line
Open TxtFileName For Output As #1
Str = OldData
For i = LBound(Data1D) To UBound(Data1D)
    Str = Str & Data1D(i) & "#"
Next i
Str = Str & vbCrLf
Str = Mid(Str, 1, Len(Str) - 2)
Print #1, Str
'Debug.Print Str
Close #1
End Function
```

'WhiteInTxtAddLineUnuse 果编号存在则替换对应编号内容,没有编号则添加在最后一行

Rem 添加内容在最后一行后,如果存在则替换
Rem 把 **2** 维数据分行写入 **TXT**，**#**以分隔。
Rem Data2D()=输入数据
Rem TxtFileName=文件名称地址

```
Public Function WhiteInTxtAddLineUnuse(ByVal Num As Long, ByRef Data1D() As Variant, _
ByVal TxtFileName As String) As Boolean
Dim i As Long, l As Long
Dim Str As Variant
Dim OldData As String
Dim Line As Long
Dim Data1DS As String, Str1 As Variant, Find As Boolean
For Each Str1 In Data1D
Data1DS = Data1DS + CStr(Str1) + "#"
```

```
Next
Find = SubReadTxtFind(Num, Data1DS, TxtFileName, OldData, Line)
If Find = False Then '没有替换
    Open TxtFileName For Output As #1
    Str = OldData + vbCrLf
    'Debug.Print Str
    For i = LBound(Data1D) To UBound(Data1D)
        Str = Str & Data1D(i) & "#"
    Next i
    Str = Str & vbCrLf
    Str = Mid(Str, 1, Len(Str) - 2)
    Print #1, Str
    Close #1
End If
End Function
```

'WhiteInTxtBySerial 文件 Txt 数据按照排头序号从小到大重写

Rem 【条件要求没有重复的序号】
Rem 数据按照排头序号从小到大重写
Rem 把 2 维数据分行写入 TXT，#以分隔。
Rem TxtFileName=文件名称地址

```
Public Function WhiteInTxtBySerial(ByVal TxtFileName As String) As Boolean
Dim ReturnLine As Long, ReturnDataT() As String
Dim Num() As Double, Str1() As Long, RLtoU() As Double
Dim i As Long, l As Long, Str As String, p As Variant, m As Long, NewStr As String
On Error Resume Next
GetDataLine TxtFileName, ReturnLine, ReturnDataT
If UBound(ReturnDataT) = 1 Then: Exit Function '只有一排时候直接退出
GetDataSerial TxtFileName, ReturnLine, ReturnDataT, Num, RLtoU, Str1
For i = LBound(Str1) To UBound(Str1)
Str = Str + ReturnDataT(Str1(i)) + vbCrLf
Array1DRemoveOneLine ReturnDataT, Str1(i), ReturnDataT
Next i
'Debug.Print TxtFileName
For i = LBound(RLtoU) To UBound(RLtoU)
    For l = LBound(Num) To UBound(Num)
        If RLtoU(i) = Num(l) Then
        'p = Split(ReturnDataT(l), "#")
        Str = Str + ReturnDataT(l) + vbCrLf
        Exit For '加速程序
        End If
    Next l
```

```
Next i
Str = Mid(Str, 1, Len(Str) - 2)
Print #1, Str
Close #1
End Function
```

'WhiteInTxtRemoveEmpty 写入 Txt 去除掉空项

Rem 写入 Txt 去除掉空项
Rem Data2D()=输入数据
Rem TxtFileName=文件名称地址

```
Public Function WhiteInTxtRemoveEmpty(ByRef Data2D() As Variant, ByVal TxtFileName As String)
As Boolean
Dim i As Long, l As Long
Dim Str As Variant
Open TxtFileName For Output As #1
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    If Data2D(i, 1) = "" Then: Exit For ' 开头文件空则消除
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
        If Data2D(i, l) <> "" Then '去除空项
        Str = Str & Data2D(i, l) & "#"
        End If
    Next l
    Str = Str & vbCrLf
    'Debug.Print Str
Next i
Str = Mid(Str, 1, Len(Str) - 2)
Print #1, Str
Close #1
End Function
```

'WhiteInTxtTitle 修改 Txt 表头

Rem 更改 2Dtxt 数据的标题
Rem Str=输入字符串
Rem TxtFileName=文件名称地址

```
Public Function WhiteInTxtTitle(ByVal Str As String, ByVal TxtFileName As String) As Boolean
Dim i As Long, l As Long
WriteLine TxtFileName, 1, Str '修改该行内容为 Str
End Function
```

'WriteLine 按行编号写入一行内容替换原来行的内容

Rem 按行编号写入一行内容替换原来行的内容
Rem TxtFileName=文件名全称
Rem Line=行数
Rem ChangeString=改变行内容

```
Public Function WriteLine(ByVal TxtFileName As String, ByRef Line As Long, _
ByVal ChangeString As String) As String
Dim n As Long, p As Variant, Str As String, B1 As String, ReturnDataT() As String
Open TxtFileName For Input As #1
'Debug.Print TxtFileName
                Do While Not EOF(1)                                    '循环至文件尾.
                Line Input #1, B1                                      '读入一行数据并将其赋予某
变量.
                n = n + 1
                ReDim Preserve ReturnDataT(n)
                ReturnDataT(n) = B1
                DoEvents
                Loop
                'ReturnLine = n '总排数
Close #1
ReturnDataT(Line) = ChangeString
Open TxtFileName For Output As #1
                For Each p In ReturnDataT
                Str = Str + p + vbCrLf
                Next
Str = Mid(Str, 1, Len(Str) - 2)
Print #1, Str
Close #1
End Function
```

15. FindRoad

详细说明：

函数或方法名称	作用
FindRoad	寻路程序

代码：

```
Option Explicit
Option Base 1
'**************************************************************************
'名称 = FindRoad.Cls
'作者 = 张宏
'最后修改时间 = 2018 年 5 月 12 日
'简介 = 2D 寻路程序
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：。
'**************************************************************************
```

'FindRoad 寻路程序

```
Rem BUG65:任何位置到(1,1)都会出错
Rem StartX,StarY 寻路开始的坐标
Rem NumX,NumY 网格范围
Rem TargetX,TargetY 目标的位置坐标
Rem PositionFreeOrNot()障碍物坐标，PositionFreeOrNot=false 有阻挡
Rem Return_Road()返回一条路径
Public Function FindRoad(ByVal StartX As Long, ByVal StartY As Long, _
ByVal NumX As Long, ByVal NumY As Long, _
ByVal TargetX As Long, ByVal TargetY As Long, ByRef PositionFreeOrNot() As Boolean, _
ByRef Return_Road() As Long)

Dim i As Long, l As Long, m As Long, n As Long, O As Long, P As Long, Q As Long
Dim nn As Long, mm As Long
Dim RowMove As Long, LineMove As Long
Dim xx As Long, yy As Long
Dim UsedPoint() As Boolean
ReDim UsedPoint(1 To NumX, 1 To NumY) ' 防止用过的坐标再次用
```

```
Dim SavePriontX() As Long, SavePriontY() As Long
ReDim Preserve SavePriontX(1)
ReDim Preserve SavePriontY(1)
Dim Save() As Long
ReDim Save(1 To NumX, 1 To NumY)
Dim Node() As Long
Dim MaxStep As Long
Dim SaveRoad() As Long
ReDim SaveRoad(1 To NumX, 1 To NumY, 1 To NumX, 1 To NumY)
Dim oo As Long
'~~~~~~~~~~~~~~~~~~~~
Rem 起始赋值
n = 1
SavePriontX(1) = StartX
SavePriontY(1) = StartY
xx = 1
yy = 1
'UsedPoint(1, 1) = True BUG:65 2018-7-22 这个为什么有，会造成 BUG
mm = 0 '用于记录
Save(StartX, StartY) = 0 '起点
nn = 0
'~~~~~~~~~~~~~~~~~~~~
Do
m = m + 1
For RowMove = -1 To 1
For LineMove = -1 To 1
        If (RowMove = 0 And LineMove <> 0) Or (LineMove = 0 And RowMove <> 0) Then '确定是上下左右移动，无斜移动，无不移动。
            If (SavePriontX(m) + RowMove > 0 And SavePriontY(m) + LineMove > 0 And SavePriontX(m) + RowMove <= NumX And SavePriontY(m) + LineMove <= NumY) Then  '防止 xx=0 or yy=0 XX,YY 超出边界
                xx = SavePriontX(m) + RowMove 'SavePriontX(m)=初始起点，后面起点
                yy = SavePriontY(m) + LineMove
                'Debug.Print "图片坐标 X=" & xx; "Y=" & yy
                    If UsedPoint(xx, yy) = False Then      '记录使用过的新起点,防止新起点 2 次使用
                        If PositionFreeOrNot(xx, yy) = True Then '这个是建筑物，敌人的时候为 False
                            n = n + 1
                            ReDim Preserve SavePriontX(n)
                            ReDim Preserve SavePriontY(n)
                            SavePriontX(n) = xx '  新的原点
                            SavePriontY(n) = yy
                            UsedPoint(xx, yy) = True '记录使用过的新起点，防止新起点 2
```

```
次使
                                    'Debug.Print "UsedPoint(" & 1 & "," & 1 & ")=" & UsedPoint(1,
1)
                                    'Debug.Print "UsedPoint(" & xx & "," & yy & ")=" & UsedPoint(xx,
yy)
                                    'Debug.Print "m=" & m & "n=" & n
                                    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                                    'Part1:2018-5-11
                                    Rem 获得何时增加 MM 使得 MM 是步数。
                                     If Save(SavePriontX(m), SavePriontY(m)) = mm Then '父节点
                                          Save(xx, yy) = mm + 1 '该父节点下子节点的 MM 值加一
                                          'Debug.Print "Save(" & xx & ", " & yy & ")="; Save(xx, yy)
                                          nn = nn + 1 ' 记录同一 mm 下 save 的个数
                                          ReDim Preserve Node(mm + 1)
                                          Node(mm + 1) = nn '记录下子节点的个数
                                          '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                                          Rem Part1:2018-5-12 用于记录到达目标的路径
                                          SaveRoad(SavePriontX(m), SavePriontY(m), xx, yy) = mm +
1
                                          'Debug.Print "SaveRoad(" & SavePriontX(m) & "," &
SavePriontY(m) & ", " & xx & ", " & yy & ")=" & mm + 1
                                       End If
                                         '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                        If xx = TargetX And yy = TargetY Then
                              'MsgBox "Find "
                              MaxStep = mm
                              Exit Do
                        End If
                        '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                                 End If
                     End If
                End If
          End If
Next LineMove
Next RowMove
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'Part2:2018-5-11
Rem 获得何时增加 MM 使得 MM 是步数。
If m <> 1 Then '第一次不算在内
oo = oo + 1
If Node(mm) = oo Then  '当 MM 值增加的次数 OO 等于子该父节点的子节点数目的时候，移
动到下一个子节点
mm = mm + 1 '移动到下一个子节点，既 MM+1
oo = 0 '归 0
```

```
nn = 0 '归 0
End If
Else '第一次
mm = mm + 1
oo = 0
nn = 0
End If
Loop While m < n
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem Part2:2018-5-12 把路径存储在 Road 中
'Dim Return_Road() As Long
ReDim Return_Road(1 To NumX, 1 To NumY)
Dim aa As Long, bb As Long
For i = MaxStep + 1 To 1 Step -1
    For l = 1 To NumX
        For m = 1 To NumY
                    If i <> MaxStep + 1 Then
                            If SaveRoad(l, m, aa, bb) = i Then
                                aa = l
                                bb = m
                                Return_Road(l, m) = i - 1
                                'Debug.Print "Return_Road(" & l & "," & m & ")=" & Return_Road(l,
m)
                            End If
                    Else
                            If SaveRoad(l, m, TargetX, TargetY) = i Then
                                aa = l
                                bb = m
                                Return_Road(l, m) = i - 1
                                'Debug.Print "Return_Road(" & l & "," & m & ")=" & Return_Road(l,
m)
                            End If
                    End If
        Next m
    Next l
Next i
Return_Road(TargetX, TargetY) = MaxStep + 1
'Debug.Print "Return_Road(" & TargetX & "," & TargetY & ")=" & Return_Road(TargetX, TargetY)
'Call MoveThrough(Return_Road)


End Function
```

16. Gdip

详细说明：

函数或方法名称	作用
GdipShowPic	显示 PNG 图片
GdipInitialize	GdipShowPic 子程序
GdipShowPic	GdipShowPic 子程序

代码：

```
Option Explicit
Option Base 1
'*************************************************************************
'名称 = Gdip .Cls
'作者 = 张宏
'最后修改时间 = 2018 年 11 月 24 日
'简介 = 显示 PNG 透明位图
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：可以将 PNG 文件显示在指定的控件的 X,Y 位置上。
'*************************************************************************
Private Declare Function GdiplusStartup Lib "gdiplus" (token As Long, inputbuf As
GdiplusStartupInput, Optional ByVal outputbuf As Long = 0) As GpStatus
Private Declare Sub GdiplusShutdown Lib "gdiplus" (ByVal token As Long)
Private Declare Function GdipCreateFromHDC Lib "gdiplus" (ByVal hwnd As Long, graphics As
Long) As GpStatus
Private Declare Function GdipDeleteGraphics Lib "gdiplus" (ByVal graphics As Long) As GpStatus
Private Declare Function GdipDrawImageRect Lib "gdiplus" (ByVal graphics As Long, ByVal image
As Long, ByVal X As Single, ByVal Y As Single, ByVal Width As Single, ByVal Height As Single) As
GpStatus
'Private Declare Function GdipLoadImageFromFile Lib "gdiplus" (ByVal FileName As String, image
As Long) As GpStatus
Private Declare Function GdipLoadImageFromFile Lib "gdiplus" (ByVal FileName As Long, hImage
As Long) As Long
Private Declare Function GdipGetImageWidth Lib "gdiplus" (ByVal image As Long, Width As Long)
As GpStatus
Private Declare Function GdipGetImageHeight Lib "gdiplus" (ByVal image As Long, Height As Long)
As GpStatus
Private Declare Function GdipDisposeImage Lib "gdiplus" (ByVal image As Long) As GpStatus
Private Type GdiplusStartupInput
    GdiplusVersion As Long
```

```
    DebugEventCallback As Long
    SuppressBackgroundThread As Long
    SuppressExternalCodecs As Long
End Type
Private Enum GpStatus
    Ok = 0
    GenericError = 1
    InvalidParameter = 2
    OutOfMemory = 3
    ObjectBusy = 4
    InsufficientBuffer = 5
    NotImplemented = 6
    Win32Error = 7
    WrongState = 8
    Aborted = 9
    FileNotFound = 10
    ValueOverflow = 11
    AccessDenied = 12
    UnknownImageFormat = 13
    FontFamilyNotFound = 14
    FontStyleNotFound = 15
    NotTrueTypeFont = 16
    UnsupportedGdiplusVersion = 17
    GdiplusNotInitialized = 18
    PropertyNotFound = 19
    PropertyNotSupported = 20
End Enum
Dim m_token As Long
```

'GdipShowPic 显示 PNG 图片

Rem 显示 PNG 图片
Rem hDC=图片句柄
Rem FileName=文件
Rem X,Y=图片坐标

```
Function GdipShowPic(ByVal hDC As Long, ByVal FileName As String, ByVal X As Long, ByVal Y As Long)
    Rem 初始化
    Call GdipInitialize
    Dim pImg As Long
    Dim pGraphics As Long
    Dim W As Long, H As Long
    Call GdipCreateFromHDC(hDC, pGraphics) '图片要画在什么地方
```

```
    'Call GdipLoadImageFromFile(StrConv(c_pngPath, vbUnicode), pImg)'这个文件位置不支持中文文件
    Call GdipLoadImageFromFile(StrPtr(FileName), pImg) '这个文件位置支持中文文件
    Call GdipGetImageWidth(pImg, W) '获得图片的宽度
    Call GdipGetImageHeight(pImg, H) '获得图片的长度
    Call GdipDrawImageRect(pGraphics, pImg, X, Y, W, H) '0,0 显示的位置坐标。W,H 显示为什么大小
    Call GdipDisposeImage(pImg)
    Call GdipDeleteGraphics(pGraphics)
    '~~~~~~~~~~~~~~~~~~~~~~~~
    Rem 关闭
    Call GdipShutDown
    '~~~~~~~~~~~~~~~~~~~~~~~~
End Function
```

Rem GdipShowPic 子程序

Rem 初始化

```
Sub GdipInitialize()
    Dim StartupInput As GdiplusStartupInput
    StartupInput.GdiplusVersion = 1
    If GdiplusStartup(m_token, StartupInput, ByVal 0) Then
        MsgBox "Error initializing GDI+"
        Exit Sub
    End If
End Sub
```

Rem GdipShowPic 子程序

Rem 关闭

```
Sub GdipShutDown()
Call GdiplusShutdown(m_token)
End Sub
```

17. GetExtName

详细说明：

函数或方法名称	作用
GetExtName	获得扩展名
Search	在路径下搜索获得文件地址显示在 Form1.List1

代码：

```
Option Explicit
Option Base 1
```

''

'函数 GetExtName
'功能:得到文件后缀名(扩展名)+查询文件
'输入:文件名
'输出:文件后缀名(扩展名)

''

'GetExtName 获得扩展名

Rem GetExtName=扩展名
Rem strFileName=路径

```
Public Function GetExtName(strFileName As String) As String
Dim strTmp As String
Dim strByte As String
Dim i As Long
For i = Len(strFileName) To 1 Step -1
strByte = Mid(strFileName, i, 1)
If strByte <> "." Then
strTmp = strByte + strTmp
Else
Exit For
End If
Next i
GetExtName = strTmp
End Function
```

'Search 在路径下搜索获得文件地址显示在 Form1.List1

```
Public Function Search(ByVal strPath As String, Optional strSearch As String = "") As Boolean
Dim strFileDir() As String
Dim strFile As String
Dim i As Long
Dim lDirCount As Long
On Error GoTo MyErr
If Right(strPath, 1) <> "\" Then strPath = strPath + "\"
strFile = Dir(strPath, vbDirectory Or vbHidden Or vbNormal Or vbReadOnly)
While strFile <> "" '搜索当前目录
DoEvents
If (GetAttr(strPath + strFile) And vbDirectory) = vbDirectory Then '如果找到的是目录
If strFile <> "." And strFile <> ".." Then '排除掉父目录(..)和当前目录(.)
lDirCount = lDirCount + 1 '将目录数增 1
ReDim Preserve strFileDir(lDirCount) As String
strFileDir(lDirCount - 1) = strFile '用动态数组保存当前目录名
End If
Else
If strSearch = "" Then
Form1.List1.AddItem strPath + strFile
ElseIf LCase(GetExtName(strPath + strFile)) = LCase(GetExtName(strSearch)) Then
'满足搜索条件，则处理该文件
Form1.List1.AddItem strPath + strFile '将文件全名保存至列表框 List1 中
End If
End If
strFile = Dir
Wend
For i = 0 To lDirCount - 1
Form1.Label3.Caption = strPath + strFileDir(i)
Call Search(strPath + strFileDir(i), strSearch) '递归搜索子目录
Next
ReDim strFileDir(0) '将动态数组清空
Search = True '搜索成功
Exit Function
MyErr:
Search = False '搜索失败
End Function
```

18. GetWindowPic

详细说明：

函数或方法名称	作用
GetGameWindowPic	获得界面图片

代码：

```
Option Explicit
Option Base 1
'**************************************************************************
'名称 = GetWindowPic .Cls
'作者 = 张宏
'最后修改时间 = 2018 年 12 月 14 日
'简介 = 获得指定窗体的截图。
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'**************************************************************************
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long '获取句柄
Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal X As Long, ByVal Y As
Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long,
ByVal ySrc As Long, ByVal dwRop As Long) As Long '获取图片数据
```

'GetGameWindowPic 获得界面图片

```
Rem SavePicture Clipboard.GetData, "c:\test.bmp"
Rem 获得一个屏幕截图
Rem hDC=窗体句柄
Rem Pic=窗体
Rem SavePathAndName=保存文件名称
Rem TitleHight=标题高度（具体什么作用有待验证）
Public Function GetGameWindowPic(hDC As Long, ByRef Pic As Object, _
SavePathAndName As String, Optional TitleHight As Long = 0) As Boolean
'FrmMain.PictureBack.BorderStyle = 2 '窗体风格,否则截取不全
Pic.AutoRedraw = True '开启自动重绘
SetForegroundWindow hDC
Sleep 500
BitBlt Pic.hDC, 0, 0, Screen.Width, Screen.Height, GetDC(0), 0, TitleHight, vbSrcCopy '获取屏幕数
```

```
据
SavePicture Pic.Image, SavePathAndName '保存为图片,覆盖模式
End Function
```

19. Judgment

详细说明：

函数或方法名称	作用
JudgmentSpecialCharacterInString	在字符串中判断指定字符是否存在
JudgmentInteger	正整数判定
JudgmentBlank	空白判定
JudgmentTextLine	行数判定
JudgmentTextAllNumbers	判断文档是否全数字
JudgmentChineseCharactersNumber	判断判断文档汉字字数
JudgmentCapitalCharactersNumber	判断判断大写字母数
JudgmentSmallLetterNumber	判断文档小写字母数
JudgmentCharactersNumber	判断文档字母数
JudgmentAsc	判断大小写和数字
JudgmentOdd	判断奇数偶数
JudgmentTextMax	判断最大值
JudgmentTextMaxAbs	判断绝对值最大值
JudgmentTextNum	判断数字

代码：

```
Option Explicit
Option Base 1
'*********************************************************************************
'名称 = S4_Judgment.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 4 月 17 日
'简介 = 判断字符数字程序。
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'判断
'增加 JudgmentTextNum                                    2016-12-28
'增加 JudgmentCharactersNum                              2019-4-17
'增加 JudgmentSmallLetterNum                             2019-4-17
'增加 JudgmentCapitalCharactersNum                       2019-4-17
```

'增加 JudgmentChineseCharactersNum 2019-4-17
'修改去掉连接符，修改批注，修改函数和过程名称 2020-2-7
'修改 JudgmentTextAllNumbers 2020-2-7
'增加 JudgmentSpecialCharacterInString 2020-2-7
'修改 JudgmentSmallLetterNum 2020-4-17
'修改 JJudgmentCapitalCharactersNum 2020-4-17
'修改 JudgmentCharactersNumber 2020-4-17
'*

'JudgmentSpecialCharacterInString 在字符串中判断指定字符是否存在

Rem 判断指定字符是否存在

```
Public Function JudgmentSpecialCharacterInString(ByVal Text As String, ByVal SpecialCharacter As String) As Boolean
Dim i As Long
JudgmentSpecialCharacterInString = False
For i = 1 To Len(Text)
Debug.Print Mid(Text, i, 1)
If Mid(Text, i, 1) = SpecialCharacter Then
JudgmentSpecialCharacterInString = True
End If
Next i
Rem 空格特殊情况
If Len(Text) = 0 Then
    If SpecialCharacter = "" Then
    JudgmentSpecialCharacterInString = True
    End If
End If
End Function
```

'JudgmentInteger 正整数判定

Rem 正整数判定
Rem Num 输入字符
Rem ReturnReport 返回过程
Rem True 表示是正整数

```
Public Function JudgmentInteger(ByVal Num As Variant, ByRef ReturnReport As String) As Boolean
If (IsNumeric(Num) = True) Then
  If CDbl(Num) > 0 And Int(Num) = CDbl(Num) Then
```

```
    JudgmentInteger = True
    ReturnReport = ReturnReport + "正整数" + CStr(Num)
    Else
    ReturnReport = ReturnReport + "不是正整数"
    JudgmentInteger = False
    End If
Else
    ReturnReport = ReturnReport + "不是数字"
    JudgmentInteger = False
End If
End Function
```

'JudgmentBlank 空白判定

Rem 判断存在空格

```
Public Function JudgmentBlank(ByVal Text As Variant, ByRef ReturnReport As String) As Boolean
If (InStr(1, Text, " ") > 0) Then
JudgmentBlank = False
ReturnReport = ReturnReport + "存在空格位置=" + CStr(InStr(1, Text, " "))
Else
JudgmentBlank = True
ReturnReport = ReturnReport + "无空格"
End If
End Function
```

'JudgmentTextLine 行数判定

Rem 判断行数

```
Public Function JudgmentTextLine(ByVal Text As Variant, ByRef ReturnReport As String) As Long
JudgmentTextLine = CLng(UBound(Split(Text, vbCrLf))) + 1
End Function
```

'JudgmentTextAllNumbers 判断文档是否全数字

Rem 判断文档是否全数字

```
Public Function JudgmentTextAllNumbers(ByVal Text As Variant, ByRef ReturnReport As String) As
Boolean
Dim Data() As String
Dim i As Long, n As Long
Data = Split(Text, vbCrLf)
JudgmentTextNum = True
```

```
For i = LBound(Data) To UBound(Data)
If IsNumeric(Data(i)) = False Then
JudgmentTextNum = False
ReturnReport = ReturnReport + i + "行非数字" + vbCrLf
n = n + 1
Else
ReturnReport = ReturnReport + i + "行数字" + vbCrLf
End If
Next i
If n = 0 Then
JudgmentTextAllNumbers = True
Else
JudgmentTextAllNumbers = False
End If
End Function
```

'JudgmentChineseCharactersNumber 判断判断文档汉字字数

Rem 判断文档汉字字数

```
Public Function JudgmentChineseCharactersNumber(ByVal Text As Variant, ByRef ReturnReport
As String) As Long
Dim Characters As Variant
For i = 1 To Len(Text)
Characters = Mid(Text, i, 1)
If (AscW(Characters) > -40870 And AscW(Characters) < -19967) Or (AscW(Characters) < 40870
And AscW(Characters) > 19967) Then
Sum = Sum + 1
End If
Next i
JudgmentChineseCharactersNumber = Sum
End Function
```

'JudgmentCapitalCharactersNumber 判断判断大写字母数

Rem 判断文档大写字母数

```
Public Function JudgmentCapitalCharactersNumber(ByVal Text As Variant, ByRef ReturnReport As
String) As Long
Dim Characters As Variant
For i = 1 To Len(Text)
Characters = Asc(Mid(Text, i, 1))
If Characters >= 65 And Characters <= 90 Then
Sum = Sum + 1
```

```
End If
Next i
JudgmentCapitalCharactersNumber = Sum
End Function
```

'JudgmentSmallLetterNumber 判断文档小写字母数

Rem 判断文档小写字母数

```
Public Function JudgmentSmallLetterNumber(ByVal Text As Variant, ByRef ReturnReport As String)
As Long
Dim Characters As Variant
For i = 1 To Len(Text)
Characters = Asc(Mid(Text, i, 1))
If Characters >= 97 And Characters <= 122 Then
Sum = Sum + 1
End If
Next i
JudgmentSmallLetterNumber = Sum
End Function
```

'JudgmentCharactersNumber 判断文档字母数

Rem 判断文档字母数

```
Public Function JudgmentCharactersNumber(ByVal Text As Variant, ByRef ReturnReport As String)
As Long
Dim Characters As Variant
For i = 1 To Len(Text)
Characters = Asc(Mid(Text, i, 1))
If (Characters >= 97 And Characters <= 122) Or (Characters >= 65 And Characters <= 90) Then
Sum = Sum + 1
End If
Next i
JudgmentCharactersNumNumber = Sum
End Function
```

'JudgmentAsc 判断大小写和数字

Rem 1=数字 **2**=小写字母 **3**=大写字母

```
Public Function JudgmentAsc(ByVal Text As Variant, ByRef Result As Long, ByRef Return_Report
As String, ByRef AscCode As Long)
        Result = 0
```

```
    If Asc(Left(Text, 1)) >= 65 And Asc(Left(Text, 1)) <= 90 Then
      Return_Report = Return_Report + "是大写字母"
      Result = 3
    ElseIf Asc(Left(Text, 1)) >= 48 And Asc(Left(Text, 1)) <= 57 Then
      Return_Report = Return_Report + "是数字"
      Result = 1
    ElseIf Asc(Left(Text, 1)) >= 97 And Asc(Left(Text, 1)) <= 122 Then
      Return_Report = Return_Report + "是小写字母"
      Result = 2
    End If
    AscCode = Asc(Left(Text, 1))
End Function
```

'JudgmentOdd 判断奇数偶数

Rem 判断奇数和偶数

```
Public Function JudgmentOdd(ByVal Data As Integer, ByRef Return_Report As String) As Boolean
If Data Mod 2 = 0 Then
    'Print 偶数
    JudgmentOdd = False
Else
    'Print 奇数
    JudgmentOdd = True
End If
End Function
```

'JudgmentTextMax 判断最大值

Rem 判断最大值

```
Public Function JudgmentTextMax(ByVal Text As Variant, ByRef Return_Report As String) As
Double
Dim obj As New Array1D001
Dim TextMax() As String, TextMaxDBL() As Double
TextMax = Split(Text, vbCrLf)
obj.Array1D_CDataTypeStoD TextMax, TextMaxDBL
JudgmentTextMax = obj.Array1D_Max(TextMaxDBL())
End Function
```

'JudgmentTextMaxAbs 判断绝对值最大值

Rem 判断绝对值最大值

```
Public Function JudgmentTextMaxAbs(ByVal Text As Variant, ByRef Return_Report As String) As Double
Dim obj As New Array1D001
Dim TextMax() As String, TextMaxDBL() As Double
TextMax = Split(Text, vbCrLf)
obj.Array1D_CDataTypeStoD TextMax, TextMaxDBL
'TextMaxDBL = Abs(TextMaxDBL)
Dim i As Long
For i = LBound(TextMaxDBL) To UBound(TextMaxDBL)
TextMaxDBL(i) = Abs(TextMaxDBL(i))
Next i
JudgmentTextMaxAbs = obj.Array1D_Max(TextMaxDBL())
End Function
```

'JudgmentTextNum 判断数字

Rem 判断数字

```
Public Function JudgmentTextNum(ByVal Text As Variant, ByRef Return_Report As String) As Boolean
Dim Data() As String
Dim i As Long
Data = Split(Text, vbCrLf)
JudgmentTextNum = True
For i = LBound(Data) To UBound(Data)
If IsNumeric(Data(i)) = False Then
JudgmentTextNum = False
Return_Report = i & "False"
Else
Return_Report = i & "True"
End If
Next i
End Function
```

20. Matrix

详细说明：

函数或方法名称	作用
TransposedMatrix	矩阵转置
MatrixAddition	矩阵加法
MatrixMultiplication	矩阵乘法
MatrixMultiplicationCoefficient	矩阵乘以系数
Factorial	阶乘
ComplementMinor	余子式
AdjointMatrix	伴随矩阵
ArrayRange	AdjointMatrix 子程序
InverseMatrix	逆矩阵
Array1DCDataTypeDtoS	把一维数组的数据类型由 Double 转换成 String
Array1DDifferentElementNum	得到数据中无重复元素的数据
InversionS	逆序数字符串型
MatrixT	矩阵初等变换
MatrixT_Main	MatrixT 子程序
MatrixT_LadderType	MatrixT 子程序
MatrixT_SwapLines	MatrixT_LadderType 子程序
MatrixT_ShowChange	MatrixT_LadderType 子程序
MatrixT_MaxLineNot0	MatrixT_LadderType 子程序
MatrixT_LineIs0	MatrixT_LadderType 子程序
MatrixT_SimplestLadderType	MatrixT_Main 子程序
MatrixT_LineChangeTo1	MatrixT_SimplestLadderType 子程序
MatrixT_RowNot0	MatrixT_SimplestLadderType 子程序
MatrixT_Simplify	MatrixT_RowNot0 子程序
MatrixT_FirstRow0	MatrixT_Main 子程序
MatrixT_CreateFirstRow0	MatrixT_Main 子程序
FullPermutation	全排列（9 位数字）
Swap	交换两个数

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*************************************************************************
'名称 = Matrix.Cls
'作者 = 张宏
```

'最后修改时间 = 2021 年 6 月 6 日
'简介 = 矩阵
'声明 = 程序中的英文只做代号，不是标准翻译
'参考文献=判断数组维度参考文献[65]
'使用说明=1.工程需要添加 Array1D190505 类模块
'　　2.工程需要添加 MatrixDetFullPermutationD 类模块
'操作历史：
'***

```
Dim obj As New S4_Array1D190505
Dim Obj1 As New S4_MatrixDetFullPermutationD
Dim Permutation() As String, CountNum As Long, PermutationMark() As String
```

Rem PasteFixT1 子程序
Rem X^T 矩阵转置
Rem DataX2D()=传入矩阵
Rem ReturnDataXT()=转置结果

```
Public Function TransposedMatrix(ByRef DataX2D() As Double, ByRef ReturnDataXT() As Double)
Dim i As Long, l As Long
Dim Num1 As Long, Num2 As Long
Num1 = UBound(DataX2D, 1) - LBound(DataX2D, 1) + 1
Num2 = UBound(DataX2D, 2) - LBound(DataX2D, 2) + 1
ReDim ReturnDataXT(1 To Num2, 1 To Num1)
For i = LBound(DataX2D, 1) To UBound(DataX2D, 1)
    For l = LBound(DataX2D, 2) To UBound(DataX2D, 2)
    ReturnDataXT(l, i) = DataX2D(i, l)
    'Debug.Print ReturnDataXT(l, i) & " l= " & l & " i= " & i
    Next l
Next i
End Function
```

'MatrixAddition 矩阵加法

Rem 矩阵+法
Rem DataA2D()=传入矩阵 A
Rem DataB2D()=传入矩阵 B
Rem ReturnDataSum()=计算结果

```
Public Function MatrixAddition(ByRef DataA2D() As Double, ByRef DataB2D() As Double, _
ByRef ReturnDataSum() As Double)
Dim i As Long, l As Long
Dim Num1 As Long, Num2 As Long
Num1 = UBound(DataA2D, 1) - LBound(DataA2D, 1) + 1
Num2 = UBound(DataA2D, 2) - LBound(DataA2D, 2) + 1
If (UBound(DataB2D, 1) <> UBound(DataA2D, 1) Or LBound(DataB2D, 1) <> LBound(DataA2D, 1)
Or UBound(DataB2D, 2) <> UBound(DataA2D, 2) Or LBound(DataB2D, 2) <> LBound(DataA2D, 2))
```

```
Then
MsgBox "数据源有问题，两矩阵数元素要相等，数组要同底。"
End If
ReDim ReturnDataSum(1 To Num1, 1 To Num2)
For i = LBound(DataA2D, 1) To UBound(DataA2D, 1)
    For l = LBound(DataA2D, 2) To UBound(DataA2D, 2)
    ReturnDataSum(i, l) = DataA2D(i, l) + DataB2D(i, l)
    'Debug.Print ReturnDataSum(i, l) & " i= " & i & " l= " & l
    Next l
Next i
End Function
```

'MatrixMultiplication 矩阵乘法

Rem 矩阵*矩阵不能够互换前后。A*B<>B*A
Rem MatrixMultiplication 只能计算 2 维数组数据,一维的不能计算需要改进 16-9-7
Rem DataA2D()=传入矩阵 A
Rem DataB2D()=传入矩阵 B
Rem ReturnDataProduct()=最终值

```
Public Function MatrixMultiplication(ByRef DataA2D() As Double, ByRef DataB2D() As Double, _
ByRef ReturnDataProduct() As Double)
Dim i As Long, l As Long, m As Long
Dim Num1 As Long, Num2 As Long, Num3 As Long, Num4 As Long
Num1 = UBound(DataA2D, 1) - LBound(DataA2D, 1) + 1
Num2 = UBound(DataA2D, 2) - LBound(DataA2D, 2) + 1
Num3 = UBound(DataB2D, 1) - LBound(DataB2D, 1) + 1
Num4 = UBound(DataB2D, 2) - LBound(DataB2D, 2) + 1
If (Num2 = Num3) Then '矩阵乘法必要
ReDim ReturnDataProduct(1 To Num1, 1 To Num4)
    For i = LBound(DataA2D, 1) To UBound(DataA2D, 1)
        For m = LBound(DataB2D, 2) To UBound(DataB2D, 2)
            For l = LBound(DataA2D, 2) To UBound(DataA2D, 2)
            ReturnDataProduct(i, m) = ReturnDataProduct(i, m) + DataA2D(i, l) * DataB2D(l, m)
            'Debug.Print ReturnDataProduct(i, m) & " i= " & i & " m= " & m
            Next l
        Next m
    Next i
Else
MsgBox "数据源有问题，两矩阵必须 A 列数等于 B 行数。"
End If
End Function
```

'MatrixMultiplicationCoefficient 矩阵乘以系数

Rem 矩阵乘以系数
Rem λ=为系数
Rem ReturnDataProduct()=返回矩阵乘以系数值

```
Public Function MatrixMultiplicationCoefficient(ByRef Data2D() As Double, _
ByVal λ As Double, ByRef ReturnDataProduct() As Double)
Dim i As Long, l As Long
Dim Num1 As Long, Num2 As Long
Num1 = UBound(Data2D, 1) - LBound(Data2D, 1) + 1
Num2 = UBound(Data2D, 2) - LBound(Data2D, 2) + 1
ReDim ReturnDataProduct(1 To Num1, 1 To Num2)
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
    ReturnDataProduct(i, l) = Data2D(i, l) * λ
    'Debug.Print ReturnDataProduct(i, l) & " i= " & i & " l= " & l
    Next l
Next i
End Function
```

'Factorial 阶乘

Rem 阶乘

```
Public Function Factorial(ByVal n As Long) As Double
Dim i As Long
Dim Part1 As Double, Part2 As Double
Factorial = 1
Part1 = 1
Part2 = 1
If (n > 171) Then
'MsgBox "无法计算 171 以上数字阶乘"
    If (n Mod 2 = 1) Then
      For i = 1 To (n / 2 + 0.5)
      Part1 = Part1 * i
      Next
      For i = (n / 2 + 0.5 + 1) To n
      Part2 = Part2 * i
      Next
      Factorial = Part1 * Part2
    End If

    If (n Mod 2 = 0) Then
```

```
        For i = 1 To (n / 2)
        Part1 = Part1 * i
        Next
        For i = (n / 2 + 1) To n
          Part2 = Part2 * i
        Next
        Factorial = Part1 * Part2
        End If
Else
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~`
For i = 1 To n
Factorial = Factorial * i
Next
End If
End Function
```

'ComplementMinor 余子式

Rem 余子式
Rem Data2D()=传入矩阵
Rem Line =行
Rem Row =列
Rem ReturnAlgebraicComplement=返回代数余子式的值
Rem ReturnM2D()=是余子式的行列形式，不是最终数值

```
Public Function ComplementMinor(ByRef Data2D() As Double, ByVal Line As Long, ByVal Row As
Long, _ByRef ReturnAlgebraicComplement As Double, ByRef ReturnM2D() As Double) As Double
Dim i As Long, l As Long, n As Long, m As Long
Dim Num1 As Long, Num2 As Long
n = 0
m = 0
Num1 = UBound(Data2D, 1) - LBound(Data2D, 1) + 1
Num2 = UBound(Data2D, 2) - LBound(Data2D, 2) + 1
ReDim ReturnM2D(1 To (Num1 - 1), 1 To (Num2 - 1))
    For i = LBound(Data2D, 1) To UBound(Data2D, 1)
          If Line <> i Then
          n = n + 1
          End If
          For l = LBound(Data2D, 2) To UBound(Data2D, 2)
                If (i = Line Or l = Row) Then
                Else
                m = m + 1
                ReturnM2D(n, m) = Data2D(i, l)
                'Debug.Print ReturnM2D(n, m) & " n= " & n & " m= " & m
```

```
            End If
        Next l
m = 0
Next i
n = 0
ComplementMinor = Obj1.Det(ReturnM2D())
ReturnAlgebraicComplement = (-1) ^ (Line + Row) * ComplementMinor
End Function
```

'AdjointMatrix 伴随矩阵

Rem 伴随矩阵 A*
Rem 传入矩阵=Data2D()
Rem 传出矩阵=ReturnData2D()

```
Public Function AdjointMatrix(ByRef Data2D() As Double, ByRef ReturnData2D() As Double)
Dim i As Long, l As Long
Dim Num1 As Long, Num2 As Long
Dim CM As Double, AC As Double
Dim r() As Double
    If (ArrayRange(Data2D) = 2) Then
    Num1 = UBound(Data2D, 1) - LBound(Data2D, 1) + 1
    Num2 = UBound(Data2D, 2) - LBound(Data2D, 2) + 1
    ReDim ReturnData2D(1 To Num1, 1 To Num2)
        For i = LBound(Data2D, 1) To UBound(Data2D, 1)
            For l = LBound(Data2D, 2) To UBound(Data2D, 2)
            CM = ComplementMinor(Data2D, i, l, AC, r())
            ReturnData2D(l, i) = AC
            'Debug.Print ReturnData2D(i, l) & " l= " & l & " i= " & i
            Next l
        Next i
    Else
    MsgBox "数据不是 2 维数据  Exit Function AdjointMatrix"
    Exit Function
    End If
End Function
```

'InverseMatrix 逆矩阵

Rem 逆阵 A^(-1)
Rem Data2D()=传入矩阵
Rem ReturnData2D()=传出矩阵

```
Public Function InverseMatrix(ByRef Data2D() As Double, ByRef ReturnData2D() As Double)
```

```
Dim i As Long, l As Long
Dim r() As Double
Dim Part1 As Double
If Obj1.Det(Data2D) = 0 Then
MsgBox "出错：此矩阵不可逆,是奇异矩阵  Exit Function InverseMatrix"
Exit Function
End If
Part1 = 1 / Obj1.Det(Data2D)
AdjointMatrix Data2D, r
MatrixMultiplicationCoefficient r, Part1, ReturnData2D
End Function
```

Rem AdjointMatrix 子程序

Rem 判断数组维度[65]

```
Public Function ArrayRange(ByRef Data() As Double) As Integer
Dim i As Integer
Dim Ret As Integer
Dim ErrF As Boolean

ErrF = False
On Error GoTo ErrHandle
'判断代入的参数是否为数组
If Not IsArray(Data) Then
ArrayRange = -1
Exit Function
End If
'VB 中数组最大为 60
For i = 1 To 60
'用 UBound 函数判断某一维的上界，如果大数组的实际维数时产生超出范围错误，
'  此时我们通过 Resume Next  来捕捉错这个错误
Ret = UBound(Data, i)
If ErrF Then Exit For
Next i
'最后返回
ArrayRange = Ret

Exit Function
ErrHandle:
Ret = i - 1
ErrF = True
Resume Next
End Function
```

'Array1DCDataTypeDtoS 把一维数组的数据类型由 Double 转换成 String

Rem 把一维数组的数据类型由 Double 转换成 String
Rem DblData()=输入的数据
Rem ReturnStrData()=传出的数据

```
Private Function Array1DCDataTypeDtoS(ByRef DblData() As Double, _
ByRef ReturnStrData() As String) As Boolean
Dim i As Long
ReDim ReturnStrData(LBound(DblData) To UBound(DblData))
For i = LBound(DblData) To UBound(DblData)
ReturnStrData(i) = CStr(DblData(i))
Next i
Array1DCDataTypeDtoS = True
End Function
```

'Array1DDifferentElementNum 得到数据中无重复元素的数据

Rem 得到数据中无重复元素的数据
Rem 例如：110 就有 1 为重复元素，120 就没有重复元素，133 就有 3 为重复元素
Rem Data()=原始数据
Rem ReturnDifferent()=数据中没有重复元素的
Rem ReturnSame()=数据中有重复元素的
Rem Array1DDifferentElementNum=返回无重复元素的数据个数

```
Public Function Array1DDifferentElementNum(ByRef Data() As Double, ByRef ReturnDifferent() As
Double, _ByRef ReturnSame() As Double) As Integer
Dim i As Long, l As Long
Dim Num() As String, DblNum() As Double, rd() As Double, r() As Long
Dim n As Long, n1 As Long, DNum As Long

For i = LBound(Data) To UBound(Data)
        For l = 1 To Len(CStr(Data(i)))
            ReDim Preserve Num(l) As String
            ReDim Preserve DblNum(l) As Double
            Num(l) = Mid(Data(i), l, 1)
            DblNum(l) = CDbl(Num(l))
        Next l
        DNum = obj.Array1DDifferentNum(DblNum(), rd(), r())
        '**************************************
        If DNum = Len(CStr(Data(i))) Then
            n = n + 1
```

```
                ReDim Preserve ReturnDifferent(n) As Double
                ReturnDifferent(n) = Data(i)
                Array1DDifferentElementNum = n
            Else
                n1 = n1 + 1
                ReDim Preserve ReturnSame(n1) As Double
                ReturnSame(n1) = Data(i)
            End If
            '**************************************
Next i
End Function
```

'InversionS 逆序数字符串型

Rem 逆序数字符串型
Rem Permutation = 排列的字符串型
Rem Digit =全排列阶数

```
Public Function InversionS(ByVal Permutation As String, ByVal Digit As Long) As Long
Dim Num() As String, i As Long, l As Long
Num = Split(Permutation, ",")
For i = 1 To Digit
    If (i + 1) <= Digit Then
        For l = 1 To Digit
            If (l + i) <= Digit Then
            'Debug.Print "Num(" & i-1 & ")=" & Num(i) & "    Num(" & L + i-1 & ")=" & Num(L +
i)
                If (Num(i - 1) > Num(l + i - 1)) Then ' 字符串也可以比较大小
                InversionS = InversionS + 1
                End If
            End If
        Next l
    End If
Next i
End Function
```

'MatrixT 矩阵初等变换

Rem 矩阵初等变换（矩阵初等变换 Matrix Elementary Transformation）

```
Public Function MatrixT(ByRef Data2D() As Double) As Long
Dim i As Long
For i = 1 To 2 '2 次运算才能化简完成
MatrixT_Main Data2D
```

```
Next i
End Function
```

Rem MatrixT 子程序

```
Public Function MatrixT_Main(ByRef Data2D() As Double) As Long
Rem  把含有 0 的项放在下面
Dim i As Long, l As Long
If MatrixT_LadderType(Data2D) = True Or MatrixT_FirstRow0(Data2D) = True Then
MatrixT_ShowChange Data2D
MatrixT_SimplestLadderType Data2D
MatrixT_ShowChange Data2D
Else
MatrixT_CreateFirstRow0 Data2D
MatrixT_ShowChange Data2D
MatrixT_LadderType Data2D
MatrixT_ShowChange Data2D
MatrixT_SimplestLadderType Data2D
MatrixT_ShowChange Data2D
End If
End Function
```

Rem MatrixT 子程序

Rem 行阶梯型

Rem 把含有 0 的项放在下面

```
Public Function MatrixT_LadderType(ByRef Data2D() As Double) As Boolean
Dim i As Long, l As Long
Dim Line As Long
Dim Swich As Boolean
For l = LBound(Data2D, 2) To UBound(Data2D, 2) Step 1
        For i = LBound(Data2D, 1) To UBound(Data2D, 1)
            If Data2D(i, l) = 0 And i + 1 <= UBound(Data2D, 1) Then '当前位置是 0，且不是最后排
                If MatrixT_LineIs0(Data2D, i, l) = True Then '当位置，同排前面位置都是 0
                        Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
                        MatrixT_MaxLineNot0 Data2D, Line, l
                    If i < Line Then '防止和下面的 0 项交换
                        Swich = MatrixT_SwapLines(Data2D, i, Line)
                        MatrixT_ShowChange Data2D
                        Debug.Print ""
                    End If
                End If
            End If
      Next i
Next l
MatrixT_ShowChange Data2D
End Function
```

Rem MatrixT_LadderType 子程序

Rem 调换行

```
Public Function MatrixT_SwapLines(ByRef Data2D() As Double, ByVal ALine As Long, _
ByVal BLine As Long) As Boolean
Dim i As Long, l As Long
Dim OldData2D() As Double
OldData2D = Data2D
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
        If i = ALine Then
        Data2D(i, l) = OldData2D(BLine, l)
        MatrixT_SwapLines = True
        End If
        If i = BLine Then
        Data2D(i, l) = OldData2D(ALine, l)
        MatrixT_SwapLines = True
        End If
        'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
    Next l
Next i
End Function
```

Rem MatrixT_LadderType 子程序

Rem 显示转变结果在 TxtResult 中

```
Public Sub MatrixT_ShowChange(ByRef Data2D() As Double)
Dim i As Long, l As Long
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
    'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
    FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + CStr(Data2D(i, l)) + vbTab
    Next l
    FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + vbCrLf
Next i
FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + vbCrLf
End Sub
```

Rem MatrixT_LadderType 子程序

Rem 获得最大的不为 0 的排的编号

```
Public Sub MatrixT_MaxLineNot0(ByRef Data2D() As Double, _
ByRef Line As Long, ByVal Row As Long)
Dim i As Long
For i = UBound(Data2D, 1) To LBound(Data2D, 1) Step -1
        If Data2D(i, Row) <> 0 Then
        Line = i
        Exit For '必须有
        End If
```

```
            'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
Next i
End Sub
```

Rem MatrixT_LadderType 子程序

Rem 所在排当前位置前面是否为 0

Rem 为 0 的话就可以进行交换

```
Public Function MatrixT_LineIs0(ByRef Data2D() As Double, _
ByVal Line As Long, ByVal Row As Long) As Boolean
Dim l As Long
MatrixT_LineIs0 = True
If Row > LBound(Data2D, 2) Then '防止是第一行
    For l = LBound(Data2D, 2) To Row Step 1
            If Data2D(Line, l) <> 0 Then
                MatrixT_LineIs0 = False
                Exit For '必须有
            End If
            'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
    Next l
End If
End Function
```

Rem MatrixT_Main 子程序

Rem 最简行阶梯

```
Public Sub MatrixT_SimplestLadderType(ByRef Data2D() As Double)
Dim i As Long, l As Long
For i = UBound(Data2D, 1) To LBound(Data2D, 1) Step -1
    MatrixT_LadderType Data2D '2020-6-11 更改位置
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
        If Data2D(i, l) <> 0 Then
        MatrixT_RowNot0 Data2D, i, l
        Exit For
        End If
    Next l
Next i
MatrixT_ShowChange Data2D
MatrixT_LadderType Data2D '增加一个，化简完成后还需要调整行
MatrixT_ShowChange Data2D
MatrixT_LineChangeTo1 Data2D
End Sub
```

Rem MatrixT_SimplestLadderType 子程序

Rem 化为 1

```
Public Sub MatrixT_LineChangeTo1(ByRef Data2D() As Double)
Dim i As Long, l As Long, m As Long
Dim Coefficient() As Double
Dim Swith() As Boolean
```

```
Rem  获取每排除以的系数
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
      For l = LBound(Data2D, 2) To UBound(Data2D, 2)
                  If Data2D(i, l) <> 0 And Data2D(i, l) <> 1 Then '不是 0 或者 1
                        'If i + 1 <= UBound(Data2D, 1) Then '不是最后一排
                            ' If Data2D(i + 1, l) = 0 Then '下面一个数是 0
                              ReDim Preserve Coefficient(i)
                              ReDim Preserve Swith(i)
                              Coefficient(i) = Data2D(i, l)
                              Swith(i) = True
                          '     End If
                        'ElseIf i = UBound(Data2D, 1) Then '最后一排的情况
                          '     ReDim Preserve Coefficient(i)
                          '     ReDim Preserve Swith(i)
                        '       Coefficient(i) = Data2D(i, l)
                       '        Swith(i) = True
                          'End If

                        Exit For
                ElseIf Data2D(i, l) = 1 Then
                Exit For
                End If
      Next l
Next i
Rem  化为 1
On Error Resume Next
          For i = LBound(Data2D, 1) To UBound(Data2D, 1)
                  For l = LBound(Data2D, 2) To UBound(Data2D, 2)
                          If Swith(i) = True Then
                          Data2D(i, l) = Data2D(i, l) / Coefficient(i)
                          End If
                  Next l
          Next i
End Sub
```

Rem MatrixT_SimplestLadderType 子程序

Rem 获得位置正上面不为 0 的单元的编号

```
Public Sub MatrixT_RowNot0(ByRef Data2D() As Double, ByVal Line As Long, _
ByVal Row As Long) ' ByRef Coefficient As Double, ByRef CalculateLine As Long)
Dim i As Long
Dim Coefficient As Double
For i = Line - 1 To LBound(Data2D, 1) Step -1 '上面所有排依次化简
          If Data2D(i, Row) <> 0 Then '最后一排不为 0  的数
          MatrixT_Simplify Data2D, i, Line, Row
          End If
```

```
Next i
End Sub
```

Rem MatrixT_RowNot0 子程序

Rem 把上面一排化简

```
Public Sub MatrixT_Simplify(ByRef Data2D() As Double, _
ByVal CalculateLine As Long, ByVal Line As Long, ByVal Row As Long)
Dim l As Long
Dim OldData2D() As Double
OldData2D = Data2D
For l = Row To UBound(Data2D, 2) '把上面一行进行变化，第一个值为 0
    If Data2D(Line, Row) <> 0 Then
    Data2D(CalculateLine, l) = Data2D(CalculateLine, l) + Data2D(Line, l) * _
    OldData2D(CalculateLine, Row) / Data2D(Line, Row) * (-1)
    End If
Next l
End Sub
```

Rem MatrixT_Main 子程序

Rem 看第一列是否为 0

```
Public Function MatrixT_FirstRow0(ByRef Data2D() As Double) As Boolean
Dim i As Long
For i = LBound(Data2D) To UBound(Data2D)
            If Data2D(i, LBound(Data2D, 2)) = 0 Then
                MatrixT_FirstRow0 = True '第一列存在 0
                Exit For
            End If
            'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
Next i
End Function
```

Rem MatrixT_Main 子程序

```
Public Sub MatrixT_CreateFirstRow0(ByRef Data2D() As Double)
MatrixT_RowNot0 Data2D, UBound(Data2D, 1), LBound(Data2D, 2)
End Sub
```

'FullPermutation 全排列（9 位数字）

Rem 全排列

Rem FullPermutation=全排列总共个数

Rem Retrun_Element()返回全排列各个方法

Rem Digit 全排列阶数

Rem 自己的全排列要比别人方法简单但是算起来要多算很多

```
Public Function FullPermutation(ByVal Digit As Long, ByRef Retrun_Element() As Double) As Long
Dim i As Long, l As Long, m As Long
Dim Part() As String
```

```
'Dim StandardNumber As Long ' 9位数字没有问题
Dim Part1 As Double: Dim Switch As Long: Dim nn As Long
If (Digit < 8 And Digit > 0) Then ' <9 时间原因，应为 7 要 33 秒，8 推测要 10 分钟但是内存溢出，9 要 33 小时
For i = 1 To Digit
ReDim Preserve Part(1 To Digit) As String
Part(i) = CStr(i)
Next i

'Debug.Print Part
'StandardNumber = CDbl(Part)
'Debug.Print StandardNumber

ReDim Retrun_Element(1 To Factorial(Digit)) As Double
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~列出所有元素
If Digit = 1 Then
Retrun_Element(1) = 1
End If
If Digit = 2 Then
Retrun_Element(1) = 12
Retrun_Element(2) = 21
End If
If Digit = 3 Then
Retrun_Element(1) = 123
Retrun_Element(2) = 132
Retrun_Element(3) = 213
Retrun_Element(4) = 231
Retrun_Element(5) = 312
Retrun_Element(6) = 321
End If
Dim i1 As Long, i2 As Long, i3 As Long, i4 As Long
Dim n As Long
n = 0
Dim Element() As String
Dim ElementD() As Double
If Digit = 4 Then '  特定的 4 按自己想出来的一种方法给出
For i1 = 1 To Digit
For i2 = 1 To Digit
For i3 = 1 To Digit
For i4 = 1 To Digit
n = n + 1
ReDim Preserve Element(n) As String
ReDim Preserve ElementD(n) As Double
'If (i1 <> i2 Or i2 <> i3 Or i3 <> i4 Or i1 <> i3 Or i1 <> i4 Or i2 <> i4) Then '
```

```
Element(n) = CStr(i1) + CStr(i2) + CStr(i3) + CStr(i4)
ElementD(n) = CDbl(Element(n))
'End If
Next i4
Next i3
Next i2
Next i1
Dim obj As New Array1D001
Dim s() As Double
FullPermutation = obj.Array1D_DifferentElementNum(ElementD(), Retrun_Element(), s())
End If
n = 0
'*****************************************************************************
************************
If Digit > 4 Then
Dim Max As String
Dim Mix As String
Dim n1 As Long
For i = 1 To Digit
Max = Max + CStr(Digit)
Mix = Mix + CStr(i)
Next
'Debug.Print Max
Do
n = n + 1
nn = nn + 1
ReDim Preserve Element(nn) As String
ReDim Preserve ElementD(nn) As Double
Element(nn) = CDbl(Mix) + n - 1

'*****************************************************
For l = 1 To (Digit - n1) '加速程序 16-7-3 ， 原来是 i     *
'*****************************************************
'If n1 = 3 Then
'MsgBox n1
'End If

For i = 1 To Digit
If Mid(Element(nn), i, 1) > Digit Then

'*******************************
n1 = i + 1 '加速程序 16-7-3            *
'*******************************
```

```
Part1 = CDbl(Element(nn)) + (10 - Digit) * 10 ^ (Digit - i)
Mix = CStr(Part1)
Element(nn) = Mix
Switch = 1

'*********************************
Exit For '加速程序 16-7-3                *
'*********************************

End If
Next i

'*********************************
If (n1 = 0) Then '加速程序 16-7-3*
Exit For                                                '*
End If                                                    '*
'*********************************

Next l
ElementD(nn) = CDbl(Element(nn))
If Switch = 1 Then
Switch = 0
n = 1
n1 = 0
End If
Loop While Element(nn) < Max
FullPermutation = obj.Array1D_DifferentElementNum(ElementD(), Retrun_Element(), s())
End If
'*****************************************************************************
****************
Else
MsgBox "错误：超出范围最多 7 位数字，或负值与 0"
End If
End Function
```

'Swap 交换两个数

Rem 别人的全排列中的交换两个数程序

```
Public Function Swap(ByRef Num1 As String, ByRef Num2 As String) '交换两个数（别人的全排列）
    tmp = Num1
    Num1 = Num2
    Num2 = tmp
End Function
```

21. MatrixDetFullPermutationD

详细说明：

函数或方法名称	作用
Det	N 阶行列式的运算
FullPermutationDictionaryStringForDet	全排列字符串字典法
CreateFirstString	FullPermutationDictionaryStringForDet 子程序
DictionaryMethodPart1	FullPermutationDictionaryStringForDet 子程序
DictionaryMethodPart2	FullPermutationDictionaryStringForDet 子程序
DictionaryMethodPart3	FullPermutationDictionaryStringForDet 子程序
DictionaryMethodPart4	FullPermutationDictionaryStringForDet 子程序
RecreateString	DictionaryMethodPart4 子程序
DetCalculatePartForSingle	FullPermutationDictionaryStringForDet 子程序
Array1DCDataTypeStoD	DictionaryMethodPart1 子程序
InversionS	DetCalculatePartForSingle 子程序

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*********************************************************************************
'名称 = MatrixDetFullPermutationD.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 7 日
'简介 = 全排列字典法
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'*********************************************************************************
```

'Det　N 阶行列式的运算

Rem N 阶行列式的运算
Rem Data2D()=传入矩阵
Rem Det=行列式值

```
Public Function Det(ByRef Data2D() As Double) As Double
Dim i As Long, l As Long, Num1 As Long, Num2 As Long
Dim n As Long, ReturnDetValue As Double
```

```
Dim Answer As Variant
Num1 = UBound(Data2D, 1) - LBound(Data2D, 1) + 1
Num2 = UBound(Data2D, 2) - LBound(Data2D, 2) + 1
If Num1 <> Num2 Then
MsgBox "不是 N 阶行列式,或是常数"
End If
FullPermutationDictionaryStringForDet Data2D, Num1, ReturnDetValue   '全排列采用字典法生成
Det = ReturnDetValue
    If Det = 0 Then
        If A13GboolCloseWrongMsg = False Then
        MsgBox "出错：此矩阵不可逆,是奇异矩阵", vbCritical
        S3GboolCritical = True '2020-6-20 奇异矩阵直接退出计算（用极小值代替 0 计算结果是否计算不可知）
        Answer = MsgBox("Det = 1E-150 继续计算，是否关闭奇异矩阵错误提示", vbYesNo)
            If Answer = vbYes Then
            A13GboolCloseWrongMsg = True
            End If
        End If
    Det = 1E-150
    End If
End Function
```

'FullPermutationDictionaryStringForDet 全排列字符串字典法

Rem 全排列字符串
Rem Data2D()=传入矩阵
Rem Digit=全排列阶数
Rem ReturnDetValue=返回行列式值

```
Public Function FullPermutationDictionaryStringForDet(ByRef Data2D() As Double, ByVal Digit As Long, _ByRef ReturnDetValue As Double)
Dim i As Long, l As Long, j As Long
Dim UseString As String, AntitoneString As String, DNumber() As Double
Dim n As Double
If Digit = 1 Then
ReturnDetValue = Data2D(LBound(Data2D), LBound(Data2D))
Exit Function
End If
CreateFirstString Digit, UseString, AntitoneString
DetCalculatePartForSingle Data2D, Digit, UseString, ReturnDetValue
Debug.Print 1 & "全排列:" & UseString
n = 1
    Do
```

```
            DictionaryMethodPart1 i, UseString, DNumber
            DictionaryMethodPart2 i, j, DNumber
            DictionaryMethodPart3 i, j, Digit, DNumber
            DictionaryMethodPart4 i, Digit, DNumber, UseString
            DetCalculatePartForSingle Data2D, Digit, UseString, ReturnDetValue
            n = n + 1
            'Debug.Print n & "全排列:" & UseString
            DoEvents
    Loop While UseString <> AntitoneString '当数字个数小于要求时继续循环
End Function
```

Rem FullPermutationDictionaryStringForDet 子程序

Rem 创建第一个排列

Rem Digit=全排列阶数

Rem ReturnFirstString=返回第一个排列

Rem ReturnAntitoneString=返回第一个排列的逆排列

```
Private Function CreateFirstString(ByVal Digit As Long, ByRef ReturnFirstString As String, _
ByRef ReturnAntitoneString As String)
Dim i As Long, n As Long, m As Long
m = Digit
For i = 1 To Digit
n = n + 1
ReturnFirstString = ReturnFirstString + CStr(n) + ","
ReturnAntitoneString = ReturnAntitoneString + CStr(m) + ","
m = m - 1
Next i
ReturnFirstString = Left(ReturnFirstString, Len(ReturnFirstString) - 1)
ReturnAntitoneString = Left(ReturnAntitoneString, Len(ReturnAntitoneString) - 1)
End Function
```

Rem FullPermutationDictionaryStringForDet 子程序

Rem 字典法步骤 1

Rem Returni=返回 i 值

Rem Numbers=一个排列

Rem ReturnDNumber()=返回排列的数字值

```
Private Function DictionaryMethodPart1(ByRef Returni As Long, ByVal Numbers As String, _
ByRef ReturnDNumber() As Double)
Dim j As Long
Dim Number() As String
Number = Split(Numbers, ",") '拆分种子字符串为数字
Array1DCDataTypeStoD Number, ReturnDNumber '转换数字为双精度型
For j = LBound(ReturnDNumber) To UBound(ReturnDNumber)
    If j + 1 <= UBound(ReturnDNumber) Then
        If ReturnDNumber(j) < ReturnDNumber(j + 1) Then
            Returni = j + 1
        End If
```

```
    End If
Next j
End Function
```

Rem FullPermutationDictionaryStringForDet 子程序

Rem 字典法步骤 2

Rem i=i 值

Rem Returnj=返回 j 值

Rem DNumber()=排列的数字值

```
Private Function DictionaryMethodPart2(ByVal i As Long, ByRef Returnj As Long, _
ByRef DNumber() As Double)
Dim k As Long
For k = LBound(DNumber) To UBound(DNumber)
        If DNumber(i - 1) < DNumber(k) Then
            Returnj = k
        End If
Next k
End Function
```

Rem FullPermutationDictionaryStringForDet 子程序

Rem 字典法步骤 3

Rem i=i 值

Rem j=j 值

Rem Digit=全排列阶数

Rem 返回 ReturnDNumber()=返回排列的数字值

```
Private Function DictionaryMethodPart3(ByVal i As Long, ByRef j As Long, ByVal Digit As Long, _
ByRef ReturnDNumber() As Double)
Dim Data As Double
Data = ReturnDNumber(i - 1)
ReturnDNumber(i - 1) = ReturnDNumber(j)
ReturnDNumber(j) = Data
End Function
```

Rem DictionaryMethodPart4 子程序

Rem 重新组建字符串

Rem DNumber()=排列数字数组

Rem Digit=全排列阶数

```
Private Function RecreateString(ByRef DNumber() As Double, ByVal Digit As Long) As String
Dim i As Long, n As Long, UseString As String
For i = 1 To Digit
    UseString = UseString + CStr(DNumber(n)) + ","
    n = n + 1
Next i
RecreateString = Left(UseString, Len(UseString) - 1)
End Function
```

Rem FullPermutationDictionaryStringForDet 子程序

Rem 字典法步骤 4

Rem i=i 值

Rem Digit=全排列阶数

Rem DNumber()=排列的数字值

Rem ReturnNumber=返回重组字符串

```
Private Function DictionaryMethodPart4(ByVal i As Long, ByVal Digit As Long, _
ByRef DNumber() As Double, ByRef ReturnNumber As String)
Dim D1Number() As Double
Dim m As Long, n As Long
D1Number = DNumber
For m = i To Digit - 1
        n = n + 1
        D1Number(Digit - n) = DNumber(m)
Next m
ReturnNumber = RecreateString(D1Number, Digit)
End Function
```

Rem FullPermutationDictionaryStringForDet 子程序

Rem 全排列计算（用单一数据）

Rem Data2D()=传入矩阵

Rem Digit=全排列阶数

Rem Permutation=排列字符串

Rem ReturnDetValue=行列式值

```
Private Sub DetCalculatePartForSingle(ByRef Data2D() As Double, ByVal Digit As Long, _
ByVal Permutation As String, ByRef ReturnDetValue As Double)
Dim i As Long, l As Long
Dim n As Long, Part1 As Double
Dim OneSequence() As String
    Part1 = 1
    OneSequence = Split(Permutation, ",")
    For n = 1 To Digit
    Part1 = Part1 * Data2D(n, CLng(OneSequence(n - 1)))
    Next n
    l = InversionS(Permutation, Digit)
    ReturnDetValue = ReturnDetValue + (-1) ^ l * Part1
End Sub
```

Rem DictionaryMethodPart1 子程序

Rem 把数组从字符串型转化成字符型

```
Private Function Array1DCDataTypeStoD(ByRef StrData() As String, ByRef ReturnDblData() As
Double) As Boolean
Dim i As Long
ReDim ReturnDblData(LBound(StrData) To UBound(StrData))
For i = LBound(StrData) To UBound(StrData)
ReturnDblData(i) = CDbl(StrData(i))
Next i
Array1DCDataTypeStoD = True
```

```
End Function
```

Rem DetCalculatePartForSingle 子程序

Rem 逆序数字符串型

Rem Permutation = 排列的字符串型

Rem Digit =全排列阶数

```
Public Function InversionS(ByVal Permutation As String, ByVal Digit As Long) As Long
Dim Num() As String, i As Long, l As Long
Num = Split(Permutation, ",")
For i = 1 To Digit
    If (i + 1) <= Digit Then
        For l = 1 To Digit
            If (l + i) <= Digit Then
            'Debug.Print "Num(" & i-1 & ")=" & Num(i) & "    Num(" & L + i-1 & ")=" & Num(L +
i)
                If (Num(i - 1) > Num(l + i - 1)) Then '  字符串也可以比较大小
                InversionS = InversionS + 1
                End If
            End If
        Next l
    End If
Next i
End Function
```

22. MatrixDetFullPermutationOwn

详细说明：

函数或方法名称	作用
SubInteractiveAll	全排列
SubInteractiveAllPart1	SubInteractiveAll 子程序
SubInteractiveAllPart2	SubInteractiveAll 子程序
SubInteractiveAllPart3	SubInteractiveAll 子程序

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'**************************************************************************
'名称 = MatrixDetFullPermutationOwn.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 20 日
'简介 = 1:非 10 位的全排列例如：A(1 to 4)B(1 to 3)C(1 to 5)的全排列
'          2:从正交试验 ODLSD,ODSSR 类模块中提取出来
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'**************************************************************************
```

'SubInteractiveAll 全排列

```
Rem 任意个字母组合
Rem LetterCombination()=字母组合,A 如果在第一列,C 如果在第三列
'那么 AC 为 LetterCombination(1)=1，LetterCombination(2)=3。
'A 如果在第一列,B 如果在第二列,C 如果在第三列
'那么 ABC 为 LetterCombination(1)=1，LetterCombination(2)=2，LetterCombination(3)=3。
Rem ReturnNumStringInteractive()=返回记录交互作用字母组合后的数字
Rem TotalInteractiveMeanNum=交互作用平均值的总数量
Rem MaxLevel()=因素最大水平数,这里是 A 后面的数字的最大值如(1 to 4)的 4,B 后面的数字的最大值(1 to 3)的 3
Rem ReturnStringInteractive()=返回获得字母及其序号组合全排列如：A1B1C1 A1B1C2 A1B2C1
Public Sub SubInteractiveAll(LetterCombination, ReturnNumStringInteractive, _
TotalInteractiveMeanNum, MaxLevel, ReturnStringInteractive)
Dim ChangeNum() As Long
```

```
ReDim ReturnNumStringInteractive(1 To UBound(LetterCombination), _
1 To TotalInteractiveMeanNum)
SubInteractiveAllPart1 LetterCombination, ReturnNumStringInteractive, _
TotalInteractiveMeanNum, MaxLevel, ReturnStringInteractive, ChangeNum
SubInteractiveAllPart2 LetterCombination, ReturnNumStringInteractive, _
TotalInteractiveMeanNum, MaxLevel, ReturnStringInteractive, ChangeNum
SubInteractiveAllPart3 LetterCombination, ReturnNumStringInteractive, _
TotalInteractiveMeanNum, MaxLevel, ReturnStringInteractive
End Sub
```

Rem SubInteractiveAll 的子程序

Rem 获得全部字母组合前的单个字母改变序号参数

```
Private Sub SubInteractiveAllPart1(LetterCombination, ReturnNumStringInteractive, _
TotalInteractiveMeanNum, MaxLevel, ReturnStringInteractive, ChangeNum)
Dim i As Long, l As Long
ReDim ChangeNum(1 To UBound(LetterCombination))
For l = 1 To UBound(LetterCombination)
    ChangeNum(l) = 1
    For i = UBound(LetterCombination) To l + 1 Step -1
        Rem 获得全部字母组合前的单个字母改变序号参数
        ChangeNum(l) = ChangeNum(l) * MaxLevel(LetterCombination(i))
    Next i
    Debug.Print "ChangeNum(" & l & ")=" & ChangeNum(l)
Next l
End Sub
```

Rem SubInteractiveAll 的子程序

Rem 获得全部字母组合前的单个字母改变序号参数

```
Private Sub SubInteractiveAllPart2(LetterCombination, ReturnNumStringInteractive, _
TotalInteractiveMeanNum, MaxLevel, ReturnStringInteractive, ChangeNum)
Dim i As Long, l As Long, m As Long, CountAll As Long, Count As Long, Add As Long
CountAll = TotalInteractiveMeanNum '交互平均数总数
Add = 1
For m = 1 To UBound(LetterCombination)
  For i = 1 To CountAll
                Count = Count + 1
                ReturnNumStringInteractive(m, i) = CStr(Add)
                Debug.Print "ReturnNumStringInteractive(" & m; "," & i & ")=" & _
                ReturnNumStringInteractive(m, i)
                If Count = ChangeNum(m) Then '达到改变范围值增加 1
                Add = Add + 1
                Count = 0
                If Add > MaxLevel(LetterCombination(m)) Then '达到序号上限归 1
                Add = 1
                End If
                End If
```

```
    Next i
    Add = 1 '重置 Add
Next m
End Sub
```

Rem SubInteractiveAll 的子程序

Rem 获得字母和序号最终组合

```
Private Sub SubInteractiveAllPart3(LetterCombination, ReturnNumStringInteractive, _
TotalInteractiveMeanNum, MaxLevel, ReturnStringInteractive)
Dim Count As Long, l As Long, i As Long
Count = 1
For l = 1 To TotalInteractiveMeanNum
      For i = 1 To UBound(LetterCombination)
      ReDim Preserve ReturnStringInteractive(l)
      ReturnStringInteractive(l) = ReturnStringInteractive(l) + _
      HeadLetter(LetterCombination(i)) + CStr(ReturnNumStringInteractive(i, l))
      Next i
      Count = Count + 1
      Debug.Print "ReturnStringInteractive(" & l & ")=" & ReturnStringInteractive(l)
Next l
End Sub
```

23. PicOperation

详细说明：

函数或方法名称	作用
GetColorFormOneBoxToAnotherBox	获得 PicGet 控件的各个像素显示到 Return_PicReslt 控件上
GetOnePointColorFormObject	从句柄窗体的 X,Y 位置获得颜色
GetColorBlockFlip	图片翻转
ChangeColorToGray	图片变成灰的
GetColorFormOneBoxToData	获得 PicGet 控件的各个像素
GetColorFormOneBoxToDataRGB	获得控件的各个像素 RGB 值
GetColorFormOneBoxToDataRGB2D	获得控件的各个像素二维 RGB 值

代码：

```
Option Explicit
Option Base 1
'**************************************************************************
'名称 = PicOperation.Cls
'作者 = 张宏
'最后修改时间 =2021 年 12 月 31 日
'简介 = 操作图片像素
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=需要把图片框设置为像素显示，边框样式没有，平面化
'操作历史：
'**************************************************************************
Private Declare Function GetPixel Lib "gdi32" (ByVal hDC As Long, ByVal X As Long, ByVal Y As Long) As Long
```

'GetColorFormOneBoxToAnotherBox 获得 PicGet 控件的各个像素显示到 Return_PicReslt 控件上

```
Rem 获得 PicGet 控件的各个像素显示到 Return_PicReslt 控件上
Rem Form 和 Image 没有测试
Public Function GetColorFormOneBoxToAnotherBox(ByRef PicGet As Object, ByRef Return_PicReslt As Object) As Boolean
Dim i As Long, l As Long
```

```
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
ReDim Pic_Pixel(0 To PicGet.Width - 1, 0 To PicGet.Height - 1) '边框设置
Return_PicReslt.Width = PicGet.Width
Return_PicReslt.Height = PicGet.Height
For i = 0 To PicGet.Width - 1
    For l = 0 To PicGet.Height - 1
        Pic_Pixel(i, l) = GetPixel(FrmMain.PicGet.hDC, i, l)
        Red = (Pic_Pixel(i, l) And &HFF)
        Green = (Pic_Pixel(i, l) And &HFF00&) / 256
        Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
        Return_PicReslt.PSet (i, l), RGB(Red, Green, Blue)
    Next l
Next i
End Function
```

'GetOnePointColorFormObject 从句柄窗体的 X,Y 位置获得颜色

Rem 从句柄窗体的 X,Y 位置获得颜色，Return_Color 返回颜色值

```
Public Function GetOnePointColorFormObject(hDC, X, Y, Return_Color) As Boolean
Dim ChoiceColor As Long
Dim Red As Variant, Green As Variant, Blue As Variant
ChoiceColor = GetPixel(hDC, X, Y)
Red = (ChoiceColor And &HFF)
Green = (ChoiceColor And &HFF00&) / 256
Blue = (ChoiceColor And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
Return_Color = RGB(Red, Green, Blue)
End Function
```

'GetColorBlockFlip 图片翻转

Rem PicBox 图片翻转为 Return_PicBox

```
Public Function GetColorBlockFlip(ByRef PicBox As Object, ByRef Return_PicBox As Object)
Dim i As Long, l As Long
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
ReDim Pic_Pixel(0 To PicBox.Width - 1, 0 To PicBox.Height - 1) 'PictureBox 边框设置
For i = 0 To PicBox.Width - 1 Step 1
    For l = 0 To PicBox.Height - 1 Step 1
        Pic_Pixel(i, l) = GetPixel(PicBox.hDC, PicBox.Width - 1 - i, l)
        Red = (Pic_Pixel(i, l) And &HFF)
        Green = (Pic_Pixel(i, l) And &HFF00&) / 256
```

```
            Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
            Return_PicBox.PSet (i, l), RGB(Red, Green, Blue)
        Next l
Next i
End Function
```

'ChangeColorToGray 图片变成灰的

Rem 图像变成灰的

```
Public Function ChangeColorToGray(ByRef PicBox As Object, ByRef Return_PicBox As Object)
Dim i As Long, l As Long
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
Dim Gray As Variant, GrayColor As Variant
ReDim Pic_Pixel(0 To PicBox.ScaleWidth - 1, 0 To PicBox.ScaleHeight - 1) 'PictureBox 边框设置
'Debug.Print PicBox.ScaleWidth
'Debug.Print PicBox.ScaleHeight
For l = 0 To PicBox.ScaleHeight - 1
        For i = 0 To PicBox.ScaleWidth - 1
            Pic_Pixel(i, l) = GetPixel(PicBox.hDC, i, l)
            Red = (Pic_Pixel(i, l) And &HFF)
            Green = (Pic_Pixel(i, l) And &HFF00&) / 256
            Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
            Gray = (9798 * Red + 19235 * Green + 3735 * Blue) / 32768          '调整灰度
            GrayColor = RGB(Gray, Gray, Gray)
            Return_PicBox.PSet (i, l), GrayColor
        Next i
Next l
End Function
```

'GetColorFormOneBoxToData 获得 PicGet 控件的各个像素

Rem 获得 PicGet 控件的各个像素储存到 Return_data

Rem Form 和 Image 没有测试

```
Public Function GetColorFormOneBoxToData(ByRef PicGet As Object, ByRef Return_data() As
Long) As Boolean
Dim i As Long, l As Long
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
ReDim Return_data(1 To PicGet.ScaleWidth, 1 To PicGet.ScaleHeight)   '边框设置
ReDim Pic_Pixel(0 To PicGet.ScaleWidth - 1, 0 To PicGet.ScaleHeight - 1) '边框设置
Debug.Print PicGet.ScaleWidth
```

```
Debug.Print PicGet.ScaleHeight
For i = 0 To PicGet.ScaleWidth - 1
    For l = 0 To PicGet.ScaleHeight - 1
        Pic_Pixel(i, l) = GetPixel(PicGet.hDC, i, l)
        Red = (Pic_Pixel(i, l) And &HFF)
        Green = (Pic_Pixel(i, l) And &HFF00&) / 256
        Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
        Return_data(i + 1, l + 1) = RGB(Red, Green, Blue)
    Next l
Next i
End Function
```

'GetColorFormOneBoxToDataRGB 获得控件的各个像素 RGB 值

Rem 获得 PicGet 控件的各个像素

Rem Form 和 Image 没有测试

```
Public Function GetColorFormOneBoxToDataRGB(ByRef PicGet As Object, _
ByRef Return_Red() As Long, ByRef Return_Green() As Long, ByRef Return_Blue() As Long) As
Boolean
Dim i As Long, l As Long
Dim Red As Long, Green As Long, Blue As Long
Dim Pic_Pixel() As Long
Dim n As Long
ReDim Return_data(1 To PicGet.ScaleWidth, 1 To PicGet.ScaleHeight)    '边框设置
ReDim Pic_Pixel(0 To PicGet.ScaleWidth - 1, 0 To PicGet.ScaleHeight - 1) '边框设置
For l = 0 To PicGet.ScaleHeight - 1
For i = 0 To PicGet.ScaleWidth - 1
        n = n + 1
        ReDim Preserve Return_Red(n)
        ReDim Preserve Return_Green(n)
        ReDim Preserve Return_Blue(n)
        Pic_Pixel(i, l) = GetPixel(PicGet.hDC, i, l)
        Red = (Pic_Pixel(i, l) And &HFF)
        Return_Red(n) = Red
        Green = (Pic_Pixel(i, l) And &HFF00&) / 256
        Return_Green(n) = Green
        Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
        Return_Blue(n) = Blue
        'Return_Data(i + 1, l + 1) = RGB(Red, Green, Blue)
    Next i
Next l
End Function
```

'GetColorFormOneBoxToDataRGB2D 获得控件的各个像素二维 RGB 值

```
Public Function GetColorFormOneBoxToDataRGB2D(ByRef PicGet As Object, _
ByRef Return_Red2D() As Long, ByRef Return_Green2D() As Long, ByRef Return_Blue2D() As Long)
As Boolean
Dim i As Long, l As Long
Dim Red As Long, Green As Long, Blue As Long
Dim Pic_Pixel() As Long
Dim n As Long
ReDim Return_data(1 To PicGet.ScaleWidth, 1 To PicGet.ScaleHeight)    '边框设置
ReDim Pic_Pixel(0 To PicGet.ScaleWidth - 1, 0 To PicGet.ScaleHeight - 1) '边框设置
ReDim Return_Red2D(1 To PicGet.ScaleWidth, 1 To PicGet.ScaleHeight)
ReDim Return_Green2D(1 To PicGet.ScaleWidth, 1 To PicGet.ScaleHeight)
ReDim Return_Blue2D(1 To PicGet.ScaleWidth, 1 To PicGet.ScaleHeight)
For l = 0 To PicGet.ScaleHeight - 1
For i = 0 To PicGet.ScaleWidth - 1
            Pic_Pixel(i, l) = GetPixel(PicGet.hDC, i, l)
            Red = (Pic_Pixel(i, l) And &HFF)
            Return_Red2D(i + 1, l + 1) = Red
            Green = (Pic_Pixel(i, l) And &HFF00&) / 256
            Return_Green2D(i + 1, l + 1) = Green
            Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
            Return_Blue2D(i + 1, l + 1) = Blue
            'Return_Data(i + 1, l + 1) = RGB(Red, Green, Blue)
      Next i
Next l
End Function
```

24. Rnd

详细说明：

函数或方法名称	作用
RndNumber	产生随机数字

代码：

```
'*************************************************************************
'名称 = Rnd.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 20 日
'简介 = 产生随机数
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'*************************************************************************
```

'RndNumber 产生随机数字

Rem UboundNum =100　　随机数上限值
Rem Min = 1　　随机数下限值
Rem Number = 99 产生号码数量(数量值应小于等于随机数上限值-随机数下限值)否则会产生死循环

```
Public Function RndNumber(ByVal LboundNum As Long, ByVal UboundNum As Long, _
ByVal Number As Long, ByRef Return_data() As Long)
  Dim i As Long, l As Long, j As Long
  ReDim Return_data(1 To Number)

  If Number > UboundNum - LboundNum Then
  MsgBox "数量超出"
  Exit Function
  End If

  For l = 1 To Number
  Randomize
     For i = 1 To Number
          Return_data(i) = Int((UboundNum - LboundNum + 1) * Rnd + LboundNum)
          For j = 1 To i
               If i <> j And Return_data(i) = Return_data(j) Then i = i - 1
```

```
        Next j
    Next i
 Next l
End Function
```

25. FSMOpration

详细说明：

函数或方法名称	作用
GetTxtMSFInputContent	获得对应位置 Text 文本保存到 MSFlexGrid
PreventMSFWorkDataChange	ClearForNext 子程序
MSFWorkFix	冻结 MSFlexGrid 中 Row 单元格
ChangeColorRow	固定单元格 Row 变色
ChangeColorRow	固定单元格 Col 变色
ShowCol1Information	第 1 列信息显示
MSFWorkShowAll	显示 MSFlexGrid 空格内的全部内容
MSFWorkPaste	粘贴动作
MSFWorkPaste1	粘贴动作用 String 保存粘贴板内容
MSFWorkPaste2	粘贴动作改为忽略冻结行,对比间比设计专用
PasteFix	MSFWorkPaste2 子程序
PasteFixT1	MSFWorkPaste2 子程序
TransposedMatrix	PasteFixT1 子程序
MSFWorkClear	MSFlexGrid 清除内容
MSFWorkCopy	复制内容
GetMaxNum	MSFWorkCopy 子程序
MSFWorkDelete	MSFlexGrid 删除动作
AddRowOrCol	MSFWorkPaste1 子程序
AddHead	MSFWorkPaste2 子程序
PositionFix	判断当前位置是否冻结（多程序子程序）
PositionInBound	判断当前位置是否超越规定边界（多程序子程序）
AutoColWidth	MSFWorkShowAll 子程序
ResetCell	重置 MSFlexGrid 单元格
ClearClick	清除 MSFlexGrid 内容
AddSheet	增加 MSFlexGrid 表单
JS	可以把文本里面的文本型公式计算出来
MSFWorkGetAugmentedMatrixA	从 MSFlexGrid 中获得增广矩阵
MSFWorkGetMatrixA	从 MSFlexGrid 中获得矩阵
MSFWorkGetMatrixB	从 MSFlexGrid 中获得矩阵

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'**************************************************************************
'名称 = S3_FSMOpration.Cls
'作者 = 张宏
'最后修改时间 =2020 年 7 月 7 日
'简介 = MSFlexGrid 控件使用类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'说明 = 1.S3GlngChoiceSheet 是表单
'       2.MSFlexGrid 控件需要添加
'操作历史：
'修改 PositionFix 2020-8-15
'**************************************************************************
Dim S1Obj2 As New S1_PrivateControlBoxOpration
Dim S4Obj1 As New S4_ControlBoxHaveORNot
```

'GetTxtMSFInputContent 获得对应位置 Text 文本保存到 MSFlexGrid

Rem MSFWork_LeaveCell 触发内容从 TxtMSFInput 保存到 MSFWork

```
Public Sub GetTxtMSFInputContent()
FrmWork.Work(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Row,
FrmWork.MSFWork(S3GlngChoiceSheet).Col) = FrmWork.TxtMSFInput.Text
FrmWork.TxtMSFInput.Visible = False
FrmWork.TxtMSFInput.Text = ""
End Sub
```

Rem ClearForNext 子程序

Rem 防止 MSFWork_leavecell 中让数据等于 TxtMSFInput=""从而没有数据

```
Public Sub PreventMSFWorkDataChange()
FrmWork.MSFWork(S3GlngChoiceSheet).Row = 0
FrmWork.MSFWork(S3GlngChoiceSheet).Col = 0
FrmWork.TxtMSFInput.Left = -1000
FrmWork.TxtMSFInput.Text = ""
End Sub
```

'MSFWorkFix 冻结 MSFlexGrid 中 Row 单元格

Rem MSFWork 冻结 MSF Row 单元格

```
Public Sub MSFWorkFix(ByVal Row As Long, ByVal Col As Long, ByVal ClearFix As Boolean)
Dim i As Long, l As Long
If ClearFix = True Then
'Erase S3GBoolMSFFix2D
    For i = LBound(S3GBoolMSFFix2D, 1) To UBound(S3GBoolMSFFix2D, 1)
        For l = LBound(S3GBoolMSFFix2D, 2) To UBound(S3GBoolMSFFix2D, 2)
        S3GBoolMSFFix2D(i, l, S3GlngChoiceSheet) = False
        Next l
    Next i
End If
S3GBoolMSFFix2D(Row, Col, S3GlngChoiceSheet) = True
ChangeColorRow Row, Col
End Sub
```

'ChangeColorRow 固定单元格 Row 变色

Rem 固定单元格 Row 变色

```
Private Sub ChangeColorRow(ByVal Row As Long, ByVal Col As Long)
FrmWork.MSFWork(S3GlngChoiceSheet).Row = Row
FrmWork.MSFWork(S3GlngChoiceSheet).Col = Col
FrmWork.MSFWork(S3GlngChoiceSheet).CellBackColor = &H808080
End Sub
```

'ChangeColorRow 固定单元格 Col 变色

Rem 固定单元格 Col 变色

```
Private Sub ChangeColorCol(ByVal Col As Long)
Dim i As Long
For i = 1 To FrmWork.MSFWork(S3GlngChoiceSheet).Cols - 1
FrmWork.MSFWork(S3GlngChoiceSheet).Col = Col
FrmWork.MSFWork(S3GlngChoiceSheet).Col = i
FrmWork.MSFWork(S3GlngChoiceSheet).CellBackColor = &H808080
Next i
End Sub
```

'ShowCol1Information 第 1 列信息显示

Rem 第 1 列信息显示

```
Public Sub ShowCol1Information(ByVal Row As Long)
FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(Row, 0) = "输入项"
End Sub
```

'MSFWorkShowAll 显示 MSFlexGrid 空格内的全部内容

Rem 显示 MSFWork 空格内的全部内容

```
Public Sub MSFWorkShowAll()
AutoColWidth FrmWork, FrmWork.MSFWork(S3GlngChoiceSheet)
End Sub
```

'MSFWorkPaste 粘贴动作

Rem MSFWork 粘贴动作

```
Public Sub MSFWorkPaste(ByVal t As Boolean)
Dim InformationRow() As String
Dim InformationCol() As String
Dim Mult As Boolean
Dim i As Long, l As Long
FrmWork.TxtMSFInput = Clipboard.GetText()
S3GBoolUserAction = False
For i = 1 To Len(FrmWork.TxtMSFInput)
    If Mid(FrmWork.TxtMSFInput, i, 1) = vbCr Or Mid(FrmWork.TxtMSFInput, i, 1) = vbTab Then
        Mult = True '判断是否粘贴多行多列
        Exit For
    End If
Next i
If Mult = True Then
    InformationRow = Split(FrmWork.TxtMSFInput, vbCrLf)
    For i = LBound(InformationRow) To UBound(InformationRow)
        InformationCol = Split(InformationRow(i), vbTab)
            For l = LBound(InformationCol) To UBound(InformationCol)
                Rem 无冻结的内容和在显示范围内的才能粘贴
                If PositionFix(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = False And _
                PositionInBound(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = True Then
                    If t = False Then
                    AddRowOrCol i, l

FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w + i, FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) _
                    = InformationCol(l)
                    'AddHead
                    Else ' 转置粘贴
                    AddRowOrCol l, i
```

```
FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w + l, FrmWork.MSFWork(S3GlngChoiceSheet).Col + i) _
                                = InformationCol(l)
                                'AddHead
                                End If
                            End If
                        Next l
        Next i
        Rem 无冻结的内容和在显示范围内的才能粘贴
        If PositionFix(LBound(InformationRow), LBound(InformationCol)) = False Then
            If PositionInBound(LBound(InformationRow), LBound(InformationCol)) = True Then
            Rem 防止复制点的内容为全部复制内容
            FrmWork.TxtMSFInput =
FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(LBound(InformationRow),
LBound(InformationCol))
            End If
        End If
Else
FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w, FrmWork.MSFWork(S3GlngChoiceSheet).Col) _
= FrmWork.TxtMSFInput
End If
FrmWork.TxtMSFInput.Left = -10000
AddHead
End Sub
```

'MSFWorkPaste1 粘贴动作用 String 保存粘贴板内容

Rem MSFWork 粘贴动作 1 改为 String

```
Public Sub MSFWorkPaste1(ByVal t As Boolean)
Dim InformationRow() As String
Dim InformationCol() As String
Dim Mult As Boolean
Dim i As Long, l As Long
Dim Str As String
On Error Resume Next
Str = Clipboard.GetText()
S3GBoolUserAction = False
For i = 1 To Len(Str)
    If Mid(Str, i, 1) = vbCr Or Mid(Str, i, 1) = vbTab Then
        Mult = True '判断是否粘贴多行多列
        Exit For
```

```
    End If
Next i
If Mult = True Then
    InformationRow = Split(Str, vbCrLf)
    For i = LBound(InformationRow) To UBound(InformationRow)
        InformationCol = Split(InformationRow(i), vbTab)
            For l = LBound(InformationCol) To UBound(InformationCol)
                Rem 无冻结的内容和在显示范围内的才能粘贴
                    If t = False Then
                        If PositionFix(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = False And _
                        PositionInBound(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = True Then
                        AddRowOrCol i, l

FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w + i, FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) _
                        = InformationCol(l)
                        'AddHead
                        End If
                    Else ' 转置粘贴
                        If PositionFix(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = False And _
                        PositionInBound(FrmWork.MSFWork(S3GlngChoiceSheet).Row + l,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + i) = True Then
                        AddRowOrCol l, i

FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w + l, FrmWork.MSFWork(S3GlngChoiceSheet).Col + i) _
                        = InformationCol(l)
                        'AddHead
                        End If
                    End If
            Next l
    Next i
    Rem 无冻结的内容和在显示范围内的才能粘贴
    If PositionFix(LBound(InformationRow), LBound(InformationCol)) = False Then
        If PositionInBound(LBound(InformationRow), LBound(InformationCol)) = True Then
        Rem 防止复制点的内容为全部复制内容
        Str = FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(LBound(InformationRow),
LBound(InformationCol))
        End If
    End If
Else
```

```
FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w, FrmWork.MSFWork(S3GlngChoiceSheet).Col) _
= Str
End If
FrmWork.TxtMSFInput.Left = -10000
AddHead
End Sub
```

'MSFWorkPaste2 粘贴动作改为忽略冻结行,对比间比设计专用

Rem MSFWork 粘贴动作 2 改为忽略冻结行,对比间比设计专用

```
Public Sub MSFWorkPaste2(ByVal t As Boolean)
Dim InformationRow() As String
Dim InformationCol() As String
Dim Mult As Boolean
Dim i As Long, l As Long
Dim m As Long, n As Long
Dim PData2D() As String
Dim Str As String
If t = True Then
ReDim PData2D(1 To S3GlngMSFUboundCol(S3GlngChoiceSheet), 1 To
S3GlngMSFUboundRow(S3GlngChoiceSheet) / 2)
Else
ReDim PData2D(1 To S3GlngMSFUboundRow(S3GlngChoiceSheet) / 2, 1 To
S3GlngMSFUboundCol(S3GlngChoiceSheet))
End If
Str = Clipboard.GetText()
S3GBoolUserAction = False
For i = 1 To Len(Str)
    If Mid(Str, i, 1) = vbCr Or Mid(Str, i, 1) = vbTab Then
        Mult = True '判断是否粘贴多行多列
        Exit For
    End If
Next i
If Mult = True Then
    InformationRow = Split(Str, vbCrLf)
    For i = LBound(InformationRow) To UBound(InformationRow)
        InformationCol = Split(InformationRow(i), vbTab)
            For l = LBound(InformationCol) To UBound(InformationCol)
                If i + 1 <= UBound(PData2D, 1) And l + 1 <= UBound(PData2D, 2) Then '复制
内容超出粘贴范围的情况
                PData2D(i + 1, l + 1) = InformationCol(l)
                End If
```

```
                Next l
        Next i
        If t = False Then
        PasteFix PData2D
        Else
        PasteFixT1 PData2D
        End If
        Rem 无冻结的内容和在显示范围内的才能粘贴
        If PositionFix(LBound(InformationRow), LBound(InformationCol)) = False Then
            If PositionInBound(LBound(InformationRow), LBound(InformationCol)) = True Then
            Rem 防止复制点的内容为全部复制内容
            Str = FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(LBound(InformationRow),
LBound(InformationCol))
            End If
        End If
Else
FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w, FrmWork.MSFWork(S3GlngChoiceSheet).Col) _
= Str
End If
FrmWork.TxtMSFInput.Left = -10000
AddHead
End Sub
```

Rem MSFWorkPaste2 子程序

Rem 粘贴忽略冻结

```
Private Sub PasteFix(ByRef PData2D() As String)
Dim n As Long, m As Long, i As Long, l As Long
                    n = 1
                    For i = 0 To S3GlngMSFUboundRow(S3GlngChoiceSheet) - 1
                        For l = 0 To S3GlngMSFUboundCol(S3GlngChoiceSheet) - 1
                            If PositionFix(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = False And _
                            PositionInBound(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = True Then
                                m = m + 1

FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w + i, FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) _
                                = PData2D(n, m)
                                If m = S3GlngMSFUboundCol(S3GlngChoiceSheet) Then
                                n = n + 1
                                m = 0
                                End If
                            End If
```

```
                    Next l
                Next i
End Sub
```

Rem 失败的转职粘贴

Rem 粘贴忽略冻结转置(这个会把所有空格按照转置的结果顺序填满)

```
Private Sub PasteFixT(ByRef PData2D() As String)
Dim n As Long, m As Long, i As Long, l As Long
Dim PData2DT() As String
TransposedMatrix PData2D, PData2DT
                n = 1
                For i = 0 To S3GlngMSFUboundRow(S3GlngChoiceSheet) - 1
                    For l = 0 To S3GlngMSFUboundCol(S3GlngChoiceSheet) - 1
                        If PositionFix(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = False And _
                        PositionInBound(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = True Then
                                m = m + 1

FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w + i, FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) _
                                = PData2DT(n, m)
                                If m = UBound(PData2DT, 2) Then
                                    If n + 1 <= UBound(PData2DT, 1) Then
                                    n = n + 1
                                    m = 0
                                    End If
                                End If
                        End If
                    Next l
                Next i
End Sub
```

Rem MSFWorkPaste2 子程序

Rem 粘贴忽略冻结转置

```
Private Sub PasteFixT1(ByRef PData2D() As String)
Dim n As Long, m As Long, i As Long, l As Long
Dim PData2DT() As String
TransposedMatrix PData2D, PData2DT
                n = 1
                For i = 0 To S3GlngMSFUboundRow(S3GlngChoiceSheet) - 1
                    For l = 0 To S3GlngMSFUboundCol(S3GlngChoiceSheet) - 1
                        If PositionFix(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = False And _
                        PositionInBound(FrmWork.MSFWork(S3GlngChoiceSheet).Row + i,
FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) = True Then
```

```
                                        If FrmWork.MSFWork(S3GlngChoiceSheet).Col + l <=
UBound(PData2DT, 2) Then
                                        m = m + 1

FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(FrmWork.MSFWork(S3GlngChoiceSheet).Ro
w + i, FrmWork.MSFWork(S3GlngChoiceSheet).Col + l) _
                                        = PData2DT(n, m)
                                            If m = UBound(PData2DT, 2) Then
                                                If n + 1 <= UBound(PData2DT, 1) Then
                                                n = n + 1
                                                m = 0
                                                End If
                                            End If
                                        End If
                                    End If
                                Next l
                            Next i
End Sub
```

'TransposedMatrix　X^T 矩阵转置

Rem X^T 矩阵转置
Rem DataX2D()=传入矩阵
Rem ReturnDataXT()=转置结果

```
Public Function TransposedMatrix(ByRef DataX2D() As String, ByRef ReturnDataXT() As String)
Dim i As Long, l As Long
Dim Num1 As Long, Num2 As Long
Num1 = UBound(DataX2D, 1) - LBound(DataX2D, 1) + 1
Num2 = UBound(DataX2D, 2) - LBound(DataX2D, 2) + 1
ReDim ReturnDataXT(1 To Num2, 1 To Num1)
For i = LBound(DataX2D, 1) To UBound(DataX2D, 1)
    For l = LBound(DataX2D, 2) To UBound(DataX2D, 2)
    ReturnDataXT(l, i) = DataX2D(i, l)
    'Debug.Print ReturnDataXT(l, i) & " l= " & l & " i= " & i
    Next l
Next i
End Function
```

'MSFWorkClear 清除内容

Rem MSFWork 清除动作

```
Public Sub MSFWorkClear()
```

```
Dim i As Long, l As Long
Dim a1 As Long, a2 As Long, B1 As Long, B2 As Long
GetMaxNum a1, a2, B1, B2
For i = a1 To a2
    For l = B1 To B2
        FrmWork.MSFWork(0).TextMatrix(i, l) = ""
    Next l
Next i
End Sub
```

'MSFWorkCopy 复制内容

Rem MSFWork 复制动作

```
Public Sub MSFWorkCopy()
Dim i As Long, l As Long
Dim CopyString As String
Dim a1 As Long, a2 As Long, B1 As Long, B2 As Long
GetMaxNum a1, a2, B1, B2
Clipboard.Clear
For i = a1 To a2
    For l = B1 To B2
        If l < B2 Then
        CopyString = CopyString + FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(i, l) +
vbTab
        Else
        CopyString = CopyString + FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(i, l)
        End If
    Next l
CopyString = CopyString + vbCrLf
Next i
Clipboard.SetText CopyString
End Sub
```

Rem MSFWorkCopy 子程序

Rem 选定区域

```
Public Sub GetMaxNum(ByRef a1 As Long, ByRef a2 As Long, _
ByRef B1 As Long, ByRef B2 As Long)
Dim a3 As Long, B3 As Long
a2 = FrmWork.MSFWork(S3GlngChoiceSheet).RowSel
a1 = FrmWork.MSFWork(S3GlngChoiceSheet).Row
B2 = FrmWork.MSFWork(S3GlngChoiceSheet).ColSel
B1 = FrmWork.MSFWork(S3GlngChoiceSheet).Col
If a2 < a1 Then
a3 = a2
```

```
a2 = a1
a1 = a3
End If
If B2 < B1 Then
B3 = B2
B2 = B1
B1 = B3
End If
End Sub
```

'MSFWorkDelete MSFlexGrid 删除动作

Rem MSFWork 删除动作

```
Public Sub MSFWorkDelete()
Dim i As Long, l As Long, t As Boolean
Dim CopyString As String
Dim a1 As Long, a2 As Long, B1 As Long, B2 As Long
GetMaxNum a1, a2, B1, B2
FrmWork.TxtMSFInput.Text = ""
Rem  选择部分删除用粘贴空内容代替
Clipboard.Clear
For i = a1 To a2
    For l = B1 To B2
        If l < B2 Then
        CopyString = CopyString + "" + vbTab
        Else
        CopyString = CopyString + ""
        End If
    Next l
CopyString = CopyString + vbCrLf
Next i
Clipboard.SetText CopyString
Call MSFWorkPaste1(t)
End Sub
```

Rem MSFWorkPaste1 子程序

Rem 复制粘贴超出范围重新扩大范围

```
Private Sub AddRowOrCol(ByRef AddRow As Long, ByRef AddCol As Long)
If FrmWork.MSFWork(S3GlngChoiceSheet).Row + AddRow >=
FrmWork.MSFWork(S3GlngChoiceSheet).Rows Then
    FrmWork.MSFWork(S3GlngChoiceSheet).Rows =
FrmWork.MSFWork(S3GlngChoiceSheet).Row + AddRow + 1
End If
If FrmWork.MSFWork(S3GlngChoiceSheet).Col + AddCol >=
```

```
FrmWork.MSFWork(S3GlngChoiceSheet).Cols Then
    FrmWork.MSFWork(S3GlngChoiceSheet).Cols = FrmWork.MSFWork(S3GlngChoiceSheet).Col
+ AddCol + 1
End If
End Sub
```

Rem MSFWorkPaste2 子程序

Rem 增加表头于空白处

```
Public Sub AddHead()
Dim i As Long
For i = 0 To FrmWork.MSFWork(S3GlngChoiceSheet).Rows - 1
If FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(i, 0) = "" Then
FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(i, 0) = i
End If
Next i
For i = 0 To FrmWork.MSFWork(S3GlngChoiceSheet).Cols - 1
If FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(0, i) = "" Then
FrmWork.MSFWork(S3GlngChoiceSheet).TextMatrix(0, i) = i
End If
Next i
End Sub
```

'PositionFix 判断当前位置是否冻结

Rem 判断当前位置是否冻结

Rem X,Y 表示坐标

```
Public Function PositionFix(ByVal X As Long, ByVal Y As Long) As Boolean
Dim i As Long
On Error GoTo ErrHandle
If S3GBoolMSFFix2D(X, Y, S3GlngChoiceSheet) = True Then
PositionFix = True
End If
Exit Function
ErrHandle:
If Err.Number = 9 Then
Exit Function '2020-8-15 由 ResumeNext 改成 ExitFunction
End If
End Function
```

'PositionInBound 判断当前位置是否超越规定边界

Rem 判断当前位置是否超越规定边界

Rem X,Y 表示坐标

```
Public Function PositionInBound(ByVal X As Long, ByVal Y As Long) As Boolean
Dim i As Long
If S3GlngMSFUboundRow(S3GlngChoiceSheet) >= X And
S3GlngMSFUboundCol(S3GlngChoiceSheet) >= Y Then
PositionInBound = True
End If
End Function
```

Rem 非公用程序

Rem 初始化系统部分参数

```
Public Sub InitializeS3()
Dim i As Long, l As Long
    For i = LBound(S3GBoolMSFFix2D, 1) To UBound(S3GBoolMSFFix2D, 1)
        For l = LBound(S3GBoolMSFFix2D, 2) To UBound(S3GBoolMSFFix2D, 2)
        S3GBoolMSFFix2D(i, l, S3GlngChoiceSheet) = False
        Next l
    Next i
S3CDAnalyzeKey(S3GlngChoiceSheet) = False
S3ACDAnalyzeKey(S3GlngChoiceSheet) = False
S3CRDAnalyzeKey(S3GlngChoiceSheet) = False
S3OFCRDAnalyzeKey(S3GlngChoiceSheet) = False
S3RBDAnalyzeKey(S3GlngChoiceSheet) = False
S3SFRBDAnalyzeKey(S3GlngChoiceSheet) = False
S3LDFAnalyzeKey(S3GlngChoiceSheet) = False
S3SFLSTDAnalyzeKey(S3GlngChoiceSheet) = False
S3SPEAnalyzeKey(S3GlngChoiceSheet) = False
S3TFSPDAnalyzeKey(S3GlngChoiceSheet) = False
S3MLSTDAnalyzeKey(S3GlngChoiceSheet) = False
S3TFCGCRDAnalyzeKey(S3GlngChoiceSheet) = False
S3TFSGCRDAnalyzeKey(S3GlngChoiceSheet) = False
S3TFRBDAnalyzeKey(S3GlngChoiceSheet) = False
S3ODAnalyzeKey(S3GlngChoiceSheet) = False
S3LCLRAnalyzeKey(S3GlngChoiceSheet) = False
S3MLRCRAnalyzeKey(S3GlngChoiceSheet) = False
S3ANOCOVAnalyzeKey(S3GlngChoiceSheet) = False
S3TFGOFAnalyzeKey(S3GlngChoiceSheet) = False
S3TFIAnalyzeKey(S3GlngChoiceSheet) = False
S3CSTFIAnalyzeKey(S3GlngChoiceSheet) = False
S3HTOVAnalyzeKey(S3GlngChoiceSheet) = False
S3Sample1Key(S3GlngChoiceSheet) = False          '平均数抽样样本容量
S3Sample2Key(S3GlngChoiceSheet) = False          '百分数抽样样本容量
S3Sample3Key(S3GlngChoiceSheet) = False          '处理数 K=2 配对设计试验重复数
S3Sample4Key(S3GlngChoiceSheet) = False          '处理数 K=2 非配对设计试验重复数
S3Sample5Key(S3GlngChoiceSheet) = False          '处理数 K>=3 的试验重复数
S3Sample6Key(S3GlngChoiceSheet) = False          '两个百分数样本容量的确定
```

```
ReDim S3GBoolMSFFix2D(0 To P2MaxFixRow, 0 To P2MaxFixCol, 0 To P2MaxSheet) '冻结参数Row 大小设置
S3GlngMSFUboundRow(S3GlngChoiceSheet) = P2MaxRow                    '边界设置
S3GlngMSFUboundCol(S3GlngChoiceSheet) = P2MaxCol                    '边界设置
S3GboolCritical = False
S3GBoolUserAction = True
InitializeS3Bas '重置非公用模块数据
InitializeAllFrm
End Sub
```

Rem 非公用程序

Rem 初始化系统部分参数

```
Public Sub InitializeS3Begin()
Dim i As Long
ReDim S3CDAnalyzeKey(0 To P2MaxSheet)
ReDim S3ACDAnalyzeKey(0 To P2MaxSheet)
ReDim S3CRDAnalyzeKey(0 To P2MaxSheet)
ReDim S3OFCRDAnalyzeKey(0 To P2MaxSheet)
ReDim S3RBDAnalyzeKey(0 To P2MaxSheet)
ReDim S3SFRBDAnalyzeKey(0 To P2MaxSheet)
ReDim S3LDFAnalyzeKey(0 To P2MaxSheet)
ReDim S3SFLSTDAnalyzeKey(0 To P2MaxSheet)
ReDim S3SPEAnalyzeKey(0 To P2MaxSheet)
ReDim S3TFSPDAnalyzeKey(0 To P2MaxSheet)
ReDim S3MLSTDAnalyzeKey(0 To P2MaxSheet)
ReDim S3TFCGCRDAnalyzeKey(0 To P2MaxSheet)
ReDim S3TFSGCRDAnalyzeKey(0 To P2MaxSheet)
ReDim S3TFRBDAnalyzeKey(0 To P2MaxSheet)
ReDim S3ODAnalyzeKey(0 To P2MaxSheet)
ReDim S3LCLRAnalyzeKey(0 To P2MaxSheet)
ReDim S3MLRCRAnalyzeKey(0 To P2MaxSheet)
ReDim S3ANOCOVAnalyzeKey(0 To P2MaxSheet)
ReDim S3TFGOFAnalyzeKey(0 To P2MaxSheet)
ReDim S3TFIAnalyzeKey(0 To P2MaxSheet)
ReDim S3CSTFIAnalyzeKey(0 To P2MaxSheet)
ReDim S3HTOVAnalyzeKey(0 To P2MaxSheet)
ReDim S3Sample1Key(0 To P2MaxSheet)          '平均数抽样样本容量
ReDim S3Sample2Key(0 To P2MaxSheet)          '百分数抽样样本容量
ReDim S3Sample3Key(0 To P2MaxSheet)          '处理数 K=2 配对设计试验重复数
ReDim S3Sample4Key(0 To P2MaxSheet)          '处理数 K=2 非配对设计试验重复数
ReDim S3Sample5Key(0 To P2MaxSheet)          '处理数 K>=3 的试验重复数
ReDim S3Sample6Key(0 To P2MaxSheet)          '两个百分数样本容量的确定
Erase S3GBoolMSFFix2D
ReDim S3GBoolMSFFix2D(0 To P2MaxFixRow, 0 To P2MaxFixCol, 0 To P2MaxSheet) '冻结参数Row 大小设置
```

```
ReDim S3GlngMSFUboundRow(0 To P2MaxSheet)                    '边界设置
ReDim S3GlngMSFUboundCol(0 To P2MaxSheet)                    '边界设置
For i = 0 To P2MaxSheet
S3GlngMSFUboundRow(i) = P2MaxRow                                    '边界设置
S3GlngMSFUboundCol(i) = P2MaxCol
Next i
S3GboolCritical = False
S3GBoolUserAction = True
InitializeS3Bas '重置非公用模块数据
InitializeAllFrm
End Sub
```

Rem 非公用程序

Rem 重置非公用模块数据

```
Public Sub InitializeS3Bas()
A6GstrPlot = 0
A6GlngWhichSameA = 0
A6GlngWhichSameB = 0
Erase D1GstrTestNameAll
Erase D2GdblData()
Erase D2GstrTestNameC()
Erase D2GstrMstrTestNameC1
D2GlngJianGe = 0
A11GlngChoiceFactorNum = 0
A11GlngChoiceInteractiveFactorNum = 0
Erase A11GstrHeadLetter
A11GBoolRandomizedBlockDesign = False
A13GlngGroupNum = 0
Erase A13GlngKickNumber
Erase A13GlngRemain
A13GboolCloseWrongMsg = False
A14GboolOneFactor = False
A14GboolCompletelyRandomizedExperiment = False
InitializeS3PBas '重置公用模块数据
End Sub
```

Rem 非公用程序

Rem 重置公用模块数据

```
Private Sub InitializeS3PBas()
P1GlngDecimalPlaces = 0
P1GlngDispose = 0
P1GlngSubPlotDispose = 0
P1GlngRepetitive = 0
P1GboolReverseType = True
Erase P1GvarData2D()
Erase P1GdblData2D()
```

```
Erase P1GstrTestName()
Erase P1GstrTxtInput()
Erase P1GlngDataXSerial()
MissingPlotAnalysis = 0
MissingPlotNumber = 0
Erase MissingPlotPositionX()
Erase MissingPlotPositionY()

End Sub
```

Rem 非公用程序

Rem 卸载窗体

```
Public Sub InitializeAllFrm()
Unload FrmD1
Unload FrmD2
Unload FrmD3
Unload FrmD4
Unload FrmD5
Unload FrmD6
Unload FrmD7
Unload FrmD8
Unload FrmD9
Unload FrmD10
End Sub
```

Rem 非公用程序

Rem 清除重置当前 MSFlexGrid

```
Public Sub ClearForNext(ByRef obj As Object, ByRef Trigger1 As Boolean)
Dim Answer As String
Dim S3Obj1 As New S3_FSMOpration
Dim S1Obj2 As New S1_PrivateControlBoxOpration
Answer = MsgBox("操作会清除工作表上所有数据,是否继续？", vbYesNo + vbInformation)
If Answer = vbYes Then
    S3Obj1.PreventMSFWorkDataChange
    FrmWork.MSFWork(S3GlngChoiceSheet).Clear
    S1Obj2.PrivateSetBoxAttribute
    S3Obj1.InitializeS3
    S1Obj2.FrmOpration1 obj
Else
    Trigger1 = True
End If
End Sub
```

Rem MSFWorkShowAll 子程序

https://blog.csdn.net/weixin_30613433/article/details/97360596?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.nonecase

Rem 2014-08-08 16:59:00 weixin_30613433 :2020-7-21

```
Public Sub AutoColWidth(Form As Object, Grid As Object)
    '统一窗体和控件的文字大小
    Dim FontSize As Integer
    FontSize = Form.FontSize
    Form.FontSize = Grid.Font.Size
    Dim RowNum As Long, ColNum As Long, ColWidth As Double
    With Grid
        '遍历每一列
        For ColNum = 0 To .Cols - 1
            ColWidth = 0
            '遍历每一行,找到最长文本
            For RowNum = 0 To .Rows - 1
                If Form.TextWidth(.TextMatrix(RowNum, ColNum)) > ColWidth Then
                    ColWidth = Form.TextWidth(.TextMatrix(RowNum, ColNum))
                End If
            Next
            '在最长文本长度的基础上加 90 缇（6 像素）
            .ColWidth(ColNum) = ColWidth + 150
        Next
    End With
    Form.FontSize = FontSize
End Sub
```

'ResetCell 重置 MSFlexGrid 单元格

Rem 重置单元格高度宽度

```
Public Sub ResetCell()
Dim i As Long, l As Long
For i = 0 To FrmWork.MSFWork(S3GlngChoiceSheet).Cols - 1
    FrmWork.MSFWork(S3GlngChoiceSheet).ColWidth(i) = 1000
Next i
For l = 0 To FrmWork.MSFWork(S3GlngChoiceSheet).Rows - 1
    FrmWork.MSFWork(S3GlngChoiceSheet).RowHeight(l) = 200
Next l
End Sub
```

'ClearClick 清除 MSFlexGrid 内容

Rem 清除内容

```
Public Sub ClearClick(ByRef Cancel As Integer)
Dim Answer As Variant
```

```
Answer = MsgBox("关闭会清除现有内容，是否继续？", vbYesNo)
If Answer = vbYes Then
PreventMSFWorkDataChange
FrmWork.MSFWork(S3GlngChoiceSheet).Clear
S1Obj2.PrivateSetBoxAttribute
InitializeS3
Else
Cancel = -1
End If
End Sub
```

'AddSheet 增加 MSFlexGrid 表单

Rem 增加表单

```
Public Sub AddSheet(ByVal n As Long)
Load FrmWork.CmdMult(n)
Set FrmWork.CmdMult(n).Container = FrmWork.PicMult
FrmWork.CmdMult(n).Caption = "工作表" & n + 1
FrmWork.CmdMult(n).Left = FrmWork.CmdAdd.Left
FrmWork.CmdAdd.Left = FrmWork.CmdAdd.Left + FrmWork.CmdMult(n).Width
FrmWork.CmdMult(n).Visible = True

Load FrmWork.MSFWork(n)
Set FrmWork.MSFWork(n).Container = FrmWork.PicWork
FrmWork.MSFWork(n).Left = 0
FrmWork.MSFWork(n).Top = 0
FrmWork.MSFWork(n).Visible = True

S3GlngChoiceSheet = n
FrmWork.TxtMSFInput.ZOrder 0
AddHead
FrmWork.MSFWork(n).Width = FrmWork.MSFWork(0).Width
FrmWork.MSFWork(n).Height = FrmWork.MSFWork(0).Height
If FrmWork.CmdAdd.Left >= 19 * FrmWork.CmdMult(0).Width + FrmWork.CmdMult(0).Left Then
FrmWork.CmdAdd.Enabled = False
Else
FrmWork.CmdAdd.Enabled = True
End If
WorkSheetGetColor n
End Sub
```

Rem 非共用程序

```
Public Sub WorkSheetGetColor(ByVal Choice As Long)
Dim i As Long
```

```
For i = 0 To S3GlngMaxSheet
    If S4Obj1.fChkControls(FrmWork, "CmdMult", i) = True Then
    FrmWork.CmdMult(i).BackColor = &H8000000B
    End If
Next i
FrmWork.CmdMult(Choice).BackColor = vbGreen
End Sub
```

Rem 非共用程序

```
Public Sub WorkSheetGetColor1()
Dim i As Long
For i = S3GlngChoiceSheet To 0 Step -1   '最大表单数
    If S4Obj1.fChkControls(FrmWork, "CmdMult", i) = True Then
    S3GlngChoiceSheet = i
    Exit For
    End If
Next i
For i = 0 To S3GlngMaxSheet
    If S4Obj1.fChkControls(FrmWork, "CmdMult", i) = True Then
    FrmWork.CmdMult(i).BackColor = &H8000000B
    End If
Next i
FrmWork.CmdMult(S3GlngChoiceSheet).BackColor = vbGreen
End Sub
```

'JS 可以把文本里面的文本型公式计算出来

Rem https://zhidao.baidu.com/question/1445906878644459780.html

Rem qq15495835 2013-10-08/2020-4-3

Rem 可以把文本里面的文本型公式计算出来。

```
Public Function JS(ByVal Expressions As String, Optional Wrong As Boolean) As String
  Dim i As Long
    Dim Mssc As Object
    Set Mssc = CreateObject("MSScriptControl.ScriptControl")
    Mssc.language = "vbscript"
    On Error GoTo EvalErr
    JS = Mssc.Eval(Expressions)
    JS = CDec(JS)
    Exit Function
EvalErr:
    'Report = Report + "出错计算式=" + Expressions
    'Debug.Print Expressions
    Wrong = True
    Exit Function
```

```
End Function
```

'MSFWorkGetAugmentedMatrixA 从 MSFlexGrid 中获得增广矩阵

Rem MSFWork 获得增广矩阵

```
Public Sub MSFWorkGetAugmentedMatrixA(ByRef MatrixA() As Double, ByRef Value() As Double)
Dim i As Long, l As Long, n As Long, m As Long
Dim a1 As Long, a2 As Long, B1 As Long, B2 As Long
Dim CopyString As String
GetMaxNum a1, a2, B1, B2
ReDim MatrixA(1 To Abs(FrmWork.MSFWork(0).RowSel - FrmWork.MSFWork(0).Row) + 1, 1 To _
Abs(FrmWork.MSFWork(0).ColSel - FrmWork.MSFWork(0).Col))
ReDim Value(1 To a2 - a1 + 1)
For i = a1 To a2
    n = n + 1
    For l = B1 To B2 - 1
        m = m + 1
        MatrixA(n, m) = CDbl(FrmWork.MSFWork(0).TextMatrix(i, l))
        Debug.Print "MatrixA(" & n & "," & m & ")=" & MatrixA(n, m)
        If l < B2 Then '2020-6-7 改了
        CopyString = CopyString + FrmWork.MSFWork(0).TextMatrix(i, l) + vbTab
        Else
        CopyString = CopyString + FrmWork.MSFWork(0).TextMatrix(i, l)
        End If
    Next l
    m = 0
CopyString = CopyString + vbCrLf
Next i
FrmWork.TxtA.Text = CopyString
n = 0
For i = a1 To a2
l = B2
n = n + 1
Value(n) = CDbl(FrmWork.MSFWork(0).TextMatrix(i, l))
Next i
End Sub
```

'MSFWorkGetMatrixA 从 MSFlexGrid 中获得矩阵

Rem MSFWork 获得矩阵

```
Public Sub MSFWorkGetMatrixA(ByRef MatrixA() As Double)
Dim i As Long, l As Long, n As Long, m As Long
```

```
Dim a1 As Long, a2 As Long, B1 As Long, B2 As Long
Dim CopyString As String
GetMaxNum a1, a2, B1, B2
ReDim MatrixA(1 To Abs(FrmWork.MSFWork(0).RowSel - FrmWork.MSFWork(0).Row) + 1, 1 To _
Abs(FrmWork.MSFWork(0).ColSel - FrmWork.MSFWork(0).Col) + 1)
For i = a1 To a2
    n = n + 1
    For l = B1 To B2
        m = m + 1
        MatrixA(n, m) = CDbl(FrmWork.MSFWork(0).TextMatrix(i, l))
        Debug.Print "MatrixA(" & n & "," & m & ")=" & MatrixA(n, m)
        If l < B2 Then '2020-6-7 改了
        CopyString = CopyString + FrmWork.MSFWork(0).TextMatrix(i, l) + vbTab
        Else
        CopyString = CopyString + FrmWork.MSFWork(0).TextMatrix(i, l)
        End If
    Next l
    m = 0
CopyString = CopyString + vbCrLf
Next i
FrmWork.TxtA.Text = CopyString
End Sub
```

'MSFWorkGetMatrixB 从 MSFlexGrid 中获得矩阵

Rem MSFWork 获得矩阵

```
Public Sub MSFWorkGetMatrixB(ByRef MatrixB() As Double)
Dim i As Long, l As Long, n As Long, m As Long
Dim a1 As Long, a2 As Long, B1 As Long, B2 As Long
Dim CopyString As String
GetMaxNum a1, a2, B1, B2
ReDim MatrixB(1 To Abs(FrmWork.MSFWork(0).RowSel - FrmWork.MSFWork(0).Row) + 1, 1 To _
Abs(FrmWork.MSFWork(0).ColSel - FrmWork.MSFWork(0).Col) + 1)
For i = a1 To a2
    n = n + 1
    For l = B1 To B2
        m = m + 1
        MatrixB(n, m) = CDbl(FrmWork.MSFWork(0).TextMatrix(i, l))
        If l < FrmWork.MSFWork(0).ColSel Then
        CopyString = CopyString + FrmWork.MSFWork(0).TextMatrix(i, l) + vbTab
        Else
        CopyString = CopyString + FrmWork.MSFWork(0).TextMatrix(i, l)
        End If
```

```
        Next l
        m = 0
CopyString = CopyString + vbCrLf
Next i
FrmWork.TxtB.Text = CopyString
End Sub
```

26. AugmentedMatrix

详细说明：

函数或方法名称	作用
AugmentedMatrixT	增广矩阵初等变换
AugmentedMatrixTMain	AugmentedMatrixT 子程序
AugmentedMatrixTLadderType	AugmentedMatrixTMain 子程序
AugmentedMatrixTSwapLines	AugmentedMatrixTLadderType 子程序
AugmentedMatrixTShowChange	AugmentedMatrixTLadderType 子程序
AugmentedMatrixTMaxLineNot0	AugmentedMatrixTLadderType 子程序
AugmentedMatrixTLineIs0	AugmentedMatrixTLadderType 子程序
AugmentedMatrixTSimplestLadderType	AugmentedMatrixTMain 子程序
AugmentedMatrixTLineChangeTo1	AugmentedMatrixTSimplestLadderType 子程序
AugmentedMatrixTRowNot0	AugmentedMatrixTSimplestLadderType 子程序
AugmentedMatrixTSimplify	AugmentedMatrixTRowNot0 子程序
AugmentedMatrixTFirstRow0	AugmentedMatrixTMain 子程序
AugmentedMatrixTCreateFirstRow0	AugmentedMatrixTMain 子程序

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*****************************************************************************
'名称 = AugmentedMatrix.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 6 月 8 日
'简介 = 增广矩阵
'声明 = 程序中的英文只做代号，不是标准翻译
'参考文献
'使用说明
'操作历史：
```

'**

'AugmentedMatrixT 矩阵初等变换

Rem 矩阵初等变换（矩阵初等变换 Augmented Matrix Elementary Transformation）

```
Public Function AugmentedMatrixT(ByRef Data2D() As Double, ByRef Value() As Double) As Long
Dim i As Long
For i = 1 To 2
AugmentedMatrixTMain Data2D(), Value
Next i
End Function
```

Rem AugmentedMatrixT 的子程序

```
Public Function AugmentedMatrixTMain(ByRef Data2D() As Double, ByRef Value() As Double) As Long
Rem 把含有 0 的项放在下面
Dim i As Long, l As Long
If AugmentedMatrixTLadderType(Data2D, Value) = True Or AugmentedMatrixTFirstRow0(Data2D) = True Then '通过交换行得到阶梯式
AugmentedMatrixTShowChange Data2D, Value
AugmentedMatrixTSimplestLadderType Data2D, Value
AugmentedMatrixTShowChange Data2D, Value
Else '没有发生过交换的行列式
AugmentedMatrixTCreateFirstRow0 Data2D, Value
AugmentedMatrixTShowChange Data2D, Value
AugmentedMatrixTLadderType Data2D, Value
AugmentedMatrixTShowChange Data2D, Value
AugmentedMatrixTSimplestLadderType Data2D, Value
AugmentedMatrixTShowChange Data2D, Value
End If
End Function
```

Rem AugmentedMatrixTMain 子程序

Rem 行阶梯型

Rem 把含有 0 的项放在下面

```
Public Function AugmentedMatrixTLadderType(ByRef Data2D() As Double, ByRef Value() As Double) As Boolean
Dim i As Long, l As Long
Dim Line As Long
Dim Swith As Boolean
For l = LBound(Data2D, 2) To UBound(Data2D, 2) Step 1
    For i = LBound(Data2D, 1) To UBound(Data2D, 1)
        If Data2D(i, l) = 0 And i + 1 <= UBound(Data2D, 1) Then '当前位置是 0，且不是最后排
            If AugmentedMatrixTLineIs0(Data2D, i, l) = True Then '当位置，同排前面位置都是 0
```

```
                    Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
                    AugmentedMatrixTMaxLineNot0 Data2D, Line, l
                If i < Line Then '防止和下面的 0 项交换
                    Swith = AugmentedMatrixTSwapLines(Data2D, Value, i, Line)
                    AugmentedMatrixTShowChange Data2D, Value
                    Debug.Print ""
                End If
            End If
        End If
    Next i
Next l
AugmentedMatrixTShowChange Data2D, Value
AugmentedMatrixTLadderType = Swith
End Function
```

Rem AugmentedMatrixTLadderType 子程序

Rem 调换行

```
Public Function AugmentedMatrixTSwapLines(ByRef Data2D() As Double, ByRef Value() As
Double, _
ByVal ALine As Long, ByVal BLine As Long) As Boolean
Dim i As Long, l As Long
Dim OldData2D() As Double, OldValue() As Double
OldData2D = Data2D
OldValue = Value
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
        If i = ALine Then
        Data2D(i, l) = OldData2D(BLine, l)
        Value(i) = OldValue(BLine)
        AugmentedMatrixTSwapLines = True
        End If
        If i = BLine Then
        Data2D(i, l) = OldData2D(ALine, l)
        Value(i) = OldValue(ALine)
        AugmentedMatrixTSwapLines = True
        End If
        'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
    Next l
Next i
End Function
```

Rem AugmentedMatrixTLadderType 子程序

Rem 显示转变结果在 TxtResult 中

```
Public Sub AugmentedMatrixTShowChange(ByRef Data2D() As Double, ByRef Value() As Double)
Dim i As Long, l As Long
Static n As Long
```

```
n = n + 1
FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + "编号:" + CStr(n) + vbCrLf
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
    'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
    FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + CStr(Data2D(i, l)) + vbTab
    Next l
    FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + vbCrLf
Next i
FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + vbCrLf
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
    FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + CStr(Value(i))
    FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + vbCrLf
Next i
    FrmWork.TxtResult.Text = FrmWork.TxtResult.Text + vbCrLf
End Sub
```

Rem AugmentedMatrixTLadderType 子程序

Rem 获得最大的不为 0 的排的编号

```
Public Sub AugmentedMatrixTMaxLineNot0(ByRef Data2D() As Double, _
ByRef Line As Long, ByVal Row As Long)
Dim i As Long
For i = UBound(Data2D, 1) To LBound(Data2D, 1) Step -1
        If Data2D(i, Row) <> 0 Then
        Line = i
        Exit For '必须有
        End If
        'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
Next i
End Sub
```

Rem AugmentedMatrixTLadderType 子程序

Rem 所在排当前位置前面是否为 0

Rem 为 0 的话就可以进行交换

```
Public Function AugmentedMatrixTLineIs0(ByRef Data2D() As Double, _
ByVal Line As Long, ByVal Row As Long) As Boolean
Dim l As Long
AugmentedMatrixTLineIs0 = True
If Row > LBound(Data2D, 2) Then '防止是第一行
    For l = LBound(Data2D, 2) To Row Step 1
            If Data2D(Line, l) <> 0 Then
                AugmentedMatrixTLineIs0 = False
                Exit For '必须有
            End If
            'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
    Next l
```

```
End If
End Function
```

Rem AugmentedMatrixTMain 子程序

Rem 最简行阶梯

```
Public Sub AugmentedMatrixTSimplestLadderType(ByRef Data2D() As Double, ByRef Value() As Double)
Dim i As Long, l As Long
For i = UBound(Data2D, 1) To LBound(Data2D, 1) Step -1
AugmentedMatrixTLadderType Data2D, Value '用于把 0 开始的行换到最下面，2021-6-11 更改位置
    For l = LBound(Data2D, 2) To UBound(Data2D, 2)
        If Data2D(i, l) <> 0 Then
        AugmentedMatrixTRowNot0 Data2D, Value, i, l
        Exit For
        End If
    Next l
Next i
AugmentedMatrixTShowChange Data2D, Value
AugmentedMatrixTLadderType Data2D, Value
AugmentedMatrixTShowChange Data2D, Value
AugmentedMatrixTLineChangeTo1 Data2D, Value
End Sub
```

Rem AugmentedMatrixTSimplestLadderType 子程序

Rem 化为 1

```
Public Sub AugmentedMatrixTLineChangeTo1(ByRef Data2D() As Double, ByRef Value() As Double)
Dim i As Long, l As Long, m As Long
Dim Coefficient() As Double
Dim Swith() As Boolean
Rem  获取每排除以的系数
For i = LBound(Data2D, 1) To UBound(Data2D, 1)
        For l = LBound(Data2D, 2) To UBound(Data2D, 2)
                If Data2D(i, l) <> 0 And Data2D(i, l) <> 1 Then '不是 0 或者 1
                    'If i + 1 <= UBound(Data2D, 1) Then '不是最后一排
                        'If Data2D(i + 1, l) = 0 Then '下面一个数是 0
                        ReDim Preserve Coefficient(i)
                        ReDim Preserve Swith(i)
                        Coefficient(i) = Data2D(i, l)
                        Swith(i) = True
                        'End If
                    'ElseIf i = UBound(Data2D, 1) Then '最后一排的情况
                        'ReDim Preserve Coefficient(i)
                        'ReDim Preserve Swith(i)
                        'Coefficient(i) = Data2D(i, l)
```

```
                                'Swith(i) = True
                            'End If
                        Exit For
                    ElseIf Data2D(i, l) = 1 Then
                    Exit For
                    End If
        Next l
Next i
Rem 化为 1
On Error Resume Next
            For i = LBound(Data2D, 1) To UBound(Data2D, 1)
                        For l = LBound(Data2D, 2) To UBound(Data2D, 2)
                                If Swith(i) = True Then
                                Data2D(i, l) = Data2D(i, l) / Coefficient(i)
                                End If
                        Next l
                        If Swith(i) = True Then
                        Value(i) = Value(i) / Coefficient(i)
                        End If
            Next i
End Sub
```

Rem AugmentedMatrixTSimplestLadderType 子程序

Rem 获得位置正上面不为 0 的单元的编号

```
Public Sub AugmentedMatrixTRowNot0(ByRef Data2D() As Double, ByRef Value() As Double, _
ByVal Line As Long, ByVal Row As Long) ' ByRef Coefficient As Double, ByRef CalculateLine As
Long)
Dim i As Long
Dim Coefficient As Double
For i = Line - 1 To LBound(Data2D, 1) Step -1 '上面所有排依次化简
            If Data2D(i, Row) <> 0 Then 'Data2D(i + 1, Row) = 0
            AugmentedMatrixTSimplify Data2D, Value, i, Line, Row
            End If
Next i
End Sub
```

Rem AugmentedMatrixTRowNot0 子程序

Rem 把上面一排化简

```
Public Sub AugmentedMatrixTSimplify(ByRef Data2D() As Double, ByRef Value() As Double, _
ByVal CalculateLine As Long, ByVal Line As Long, ByVal Row As Long)
Dim l As Long
Dim OldData2D() As Double
OldData2D = Data2D
Dim OldValue() As Double
OldValue = Value
For l = Row To UBound(Data2D, 2)
```

```
    If Data2D(Line, Row) <> 0 Then
    Data2D(CalculateLine, l) = Data2D(CalculateLine, l) + Data2D(Line, l) * _
    OldData2D(CalculateLine, Row) / Data2D(Line, Row) * (-1)
    End If
Next l
Value(CalculateLine) = Value(CalculateLine) + Value(Line) * _
OldData2D(CalculateLine, Row) / Data2D(Line, Row) * (-1)
AugmentedMatrixTShowChange Data2D, Value
End Sub
```

Rem AugmentedMatrixTFirstRow0 子程序

Rem 看第一列是否为 0

```
Public Function AugmentedMatrixTFirstRow0(ByRef Data2D() As Double) As Boolean
Dim i As Long
For i = LBound(Data2D) To UBound(Data2D)
            If Data2D(i, LBound(Data2D, 2)) = 0 Then
                AugmentedMatrixTFirstRow0 = True '第一列存在 0
                Exit For
             End If
             'Debug.Print "Data2D(" & i & "," & l & ")= "; Data2D(i, l)
Next i
End Function
```

Rem AugmentedMatrixTMain 子程序

```
Public Sub AugmentedMatrixTCreateFirstRow0(ByRef Data2D() As Double, ByRef Value() As
Double)
AugmentedMatrixTRowNot0 Data2D, Value, UBound(Data2D, 1), LBound(Data2D, 2)
End Sub
```

27. SerchPicture

详细说明：

函数或方法名称	作用
OtherLinesPosition	SerchOneLineOfPicture_Nomal 子程序
ShowOneLinePositionOnPicture	放置线搜索线和设置颜色
ShowAllLinePositionsOnPicture	放置所有线搜索线和设置颜色
SerchOneLineOfPicture_Slowest	搜索图片的一排（最慢的版本）
Array1D_CDataTypeLtoD	多程序子程序
SerchOneLineOfPicture_Slower	搜索图片的一排（较慢的版本）
SerchOneLineOfPicture_Slow	搜索图片的一排（慢的版本）
SerchOneLineOfPictureSerchPixel	搜索目标数据按像素长度（多程序子程序）
SerchOneLineOfPicture_Nomal	采用 10 个像素作为像素头，搜索全图
SerchOneLineOfPictureSerchPixel2D	搜索图片的一排的子程序，二维目标
Array1D_Max	SerchOneLineOfPicture_Nomal 子程序
SerchOneLineOfPictureThenAllPic_Slowest	采用 10 个像素作为像素头，搜索全图（最慢）
SearchPicturePixel2D	二维像素数据搜图（多程序子程序）
SerchOneLineOfPictureThenAllPic_Slower	采用 10 个像素作为像素头，搜索全图（较慢）
SerchOneLineOfPictureThenAllPic_Slow	采用 10 个像素作为像素头，搜索全图（慢）
SearchHead	搜索头文件 SerchOneLineOfPictureThenAllPic_Normal 子程序
SerchOneLineOfPictureThenAllPic_Normal	搜索图片先一排然后全部
SixPointSearch	找到 4 个线 r 都大于 0.9 的时候，判定基本找到最终点，然后就依照这个点寻找旁边 6 个点的 R 全图值，找到最大的 R
Array1D_Max_Which	SixPointSearch 子程序
SearchNext	头文件找到，进一步搜索 SerchOneLineOfPictureThenAllPic_Normal 子程序
RecordSearchResult	记录搜索结果 SerchOneLineOfPictureThenAllPic_Normal 子程序

代码：

```
Option Explicit
Option Base 1
Dim obj As New Statistics_CorrelationAnalysis002
'*************************************************************************
'名称 = SerchPicture.Cls
'作者 = 张宏
```

'最后修改时间 = 2018 年 12 月 18 日
'简介 = 寻图类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=Nomarl Slow Slower Slowest 是用的横向长条图片
'操作历史：
'**********

Rem SerchOneLineOfPicture_Nomal 子程序
Rem 垂直于检测行的行
Rem Position 现在

```
Private Sub OtherLinesPosition(ByVal Position As Long, ByVal LineNumber As Long, _
ByVal PictureSourceWidth As Long, ByRef Return_OtherLinesPosition As Long)
Return_OtherLinesPosition = Position + (LineNumber - 1) * PictureSourceWidth
End Sub
```

'ShowOneLinePositionOnPicture 放置线搜索线和设置颜色

Rem 放置线 Line 的位置和颜色
Rem Position=SerchOneLineOfPicture 传入位置
Rem PicSource=需要搜索的 Picturebox
Rem PicTarget =搜索的目标 Picturebox
Rem Line = Line 控件
Rem Return_PositionX=返回 Line 的 X 的坐标
Rem Return_PositionY=返回 Line 的 Y 的坐标
Rem Colour=颜色

```
Public Function ShowOneLinePositionOnPicture(ByVal Position As Long, _
ByRef PicSource As Object, ByRef PicTarget As Object, ByRef Line As Object, _
ByRef Return_PositionX As Long, ByRef Return_PositionY As Long, _
Optional ByVal Colour As Long = 0) As String
If Colour = 1 Then
Line.BorderColor = vbRed
End If
Line.Visible = True
Line.X1 = (Position Mod PicSource.ScaleWidth)
Return_PositionX = (Position Mod PicSource.ScaleWidth)
Line.Y1 = Int(Position / PicSource.ScaleWidth)
Return_PositionY = Int(Position / PicSource.ScaleWidth)
Line.Y2 = Line.Y1
Line.X2 = (Position Mod PicSource.ScaleWidth) + PicTarget.ScaleWidth
ShowOneLinePositionOnPicture = ShowOneLinePositionOnPicture + "Position=" + CStr(Position) +
vbCrLf
ShowOneLinePositionOnPicture = ShowOneLinePositionOnPicture + "Y=" + CStr(Int(Position /
PicSource.ScaleWidth)) + vbCrLf
ShowOneLinePositionOnPicture = ShowOneLinePositionOnPicture + "X=" + CStr(Position Mod
```

```
PicSource.ScaleWidth) + vbCrLf
End Function
```

'ShowAllLinePositionsOnPicture 放置所有线搜索线和设置颜色

Rem 放置线 Line()的位置和颜色
Rem Position()=SerchOneLineOfPicture 传入位置
Rem PicSource=需要搜索的 Picturebox
Rem PicTarget =搜索的目标 Picturebox
Rem LineShow= Line 控件
Rem Return_PositionX=返回 Line 的 X 的坐标
Rem Return_PositionY=返回 Line 的 Y 的坐标
Rem Colour=颜色

```
Public Function ShowAllLinePositionsOnPicture(ByRef Position() As Long, _
ByRef PicSource As Object, ByRef PicTarget As Object, ByRef LineShow As Object, _
ByRef Return_PositionX() As Long, ByRef Return_PositionY() As Long) As String
On Error GoTo a1:
Dim i As Long, n As Integer
ReDim Return_PositionX(LBound(Position) To UBound(Position))
ReDim Return_PositionY(LBound(Position) To UBound(Position))
For i = LBound(Position) To UBound(Position)
n = n + 1
Load LineShow(n)
LineShow(n).Container = PicSource
LineShow(n).Visible = True
LineShow(n).X1 = (Position(n) Mod PicSource.ScaleWidth)
Return_PositionX(n) = (Position(n) Mod PicSource.ScaleWidth)
LineShow(n).Y1 = Int(Position(n) / PicSource.ScaleWidth)
Return_PositionY(n) = Int(Position(n) / PicSource.ScaleWidth)
LineShow(n).Y2 = LineShow(n).Y1
LineShow(n).X2 = (Position(n) Mod PicSource.ScaleWidth) + PicTarget.ScaleWidth
ShowAllLinePositionsOnPicture = ShowAllLinePositionsOnPicture + "Position=" + CStr(Position(n))
+ vbCrLf
ShowAllLinePositionsOnPicture = ShowAllLinePositionsOnPicture + "Y=" + CStr(Int(Position(n) /
PicSource.ScaleWidth)) + vbCrLf
ShowAllLinePositionsOnPicture = ShowAllLinePositionsOnPicture + "X=" + CStr(Position(n) Mod
PicSource.ScaleWidth) + vbCrLf
Next i
Exit Function
a1:
MsgBox "出错 ShowAllLinePositionsOnPicture"
End Function
```

'SerchOneLineOfPicture_Slowest 搜索图片的一排（最慢的版本）

Rem 搜索图片的一排（最慢的版本）
Rem DataSource1D()=对比数据源
Rem DataTarget1D()=对比数据目标
Rem Return_rMax=返回相关系数最大值
Rem Return_Position=返回找到位置
Rem Return_r()=返回相关系数
Rem CalculateType=没有作用

```
Public Function SerchOneLineOfPicture_Slowest(ByRef DataSource1D() As Long, _
ByRef DataTarget1D() As Long, ByRef Return_rMax As Double, _
ByRef Return_Position As Long, ByRef Return_r() As Double, Optional ByVal CalculateType As
Long = 1) As Double
Dim DataCutFromDataSource1D() As Double, n As Long, r As Double, l As Long, i As Long
Dim DataTarget1D_Dbl() As Double
Dim DataTarget1D_Dbl1() As Double
Array1D_CDataTypeLtoD DataTarget1D, DataTarget1D_Dbl
Do While l <= (UBound(DataSource1D) - UBound(DataTarget1D))
For i = LBound(DataTarget1D) To UBound(DataTarget1D)
n = n + 1
ReDim Preserve DataCutFromDataSource1D(n)
DataCutFromDataSource1D(n) = DataSource1D(i + l)
'Debug.Print "DataCutFromDataSource1D(" + CStr(N) + ") =" +
CStr(DataCutFromDataSource1D(N))
'Debug.Print "DataTarget1D_Dbl(" + CStr(N) + ")=" + CStr(DataTarget1D_Dbl(N))
Next i
n = 0
l = l + 1
DataTarget1D_Dbl1 = DataTarget1D_Dbl
r = obj.ρ(DataCutFromDataSource1D, DataTarget1D_Dbl1)
ReDim Preserve Return_r(l)
Return_r(l) = r
'Debug.Print "r=" & r
'==================
Rem 获取最大相关系数值
If r > Return_rMax Then
Return_rMax = r
Return_Position = l '获得最大相关系数位置
End If
If r > 0.99 Then
GoTo a1:
End If
Loop
```

```
a1:
End Function
```

Rem SerchOneLineOfPicture_Slower 子程序

```
Private Function Array1D_CDataTypeLtoD(ByRef LngData() As Long, ByRef Return_DblData() As
Double) As Boolean
Dim i As Long
ReDim Return_DblData(LBound(LngData) To UBound(LngData))
For i = LBound(LngData) To UBound(LngData)
Return_DblData(i) = CDbl(LngData(i))
Next i
Array1D_CDataTypeLtoD = True
End Function
```

'SerchOneLineOfPicture_Slower 搜索图片的一排（较慢的版本）

Rem 搜索图片的一排（更慢的版本）
Rem DataSource1D()=对比数据源
Rem DataTarget1D()=对比数据目标
Rem PictureSourceWidth=搜索原图宽度
Rem Return_rMax=返回相关系数最大值
Rem Return_Position=返回找到位置
Rem Return_r()=返回相关系数
Rem CalculateType=没有作用

```
Public Function SerchOneLineOfPicture_Slower(ByRef DataSource1D() As Long, _
ByRef DataTarget1D() As Long, ByVal PictureSourceWidth As Long, ByRef Return_rMax As Double,
_
ByRef Return_Position As Long, ByRef Return_r() As Double, Optional ByVal CalculateType As
Long = 1) As Double
Dim DataCutFromDataSource1D() As Double, n As Long, r As Double, l As Long, i As Long
Dim DataTarget1D_Dbl() As Double
Dim DataTarget1D_Dbl1() As Double
Array1D_CDataTypeLtoD DataTarget1D, DataTarget1D_Dbl
Do While l <= (UBound(DataSource1D) - UBound(DataTarget1D))
'=======================================================
Rem 超出横向边框直接跳过
If ((l + 1) Mod PictureSourceWidth) + (UBound(DataTarget1D) - LBound(DataTarget1D) + 1) >
PictureSourceWidth Then
l = l + 1
GoTo a2
End If
'=======================================================
Rem 头像素比较 5 像素
For i = LBound(DataTarget1D) To UBound(DataTarget1D)
```

```
n = n + 1
ReDim Preserve DataCutFromDataSource1D(n)
DataCutFromDataSource1D(n) = DataSource1D(i + l)
Next i
n = 0
l = l + 1
DataTarget1D_Dbl1 = DataTarget1D_Dbl
r = obj.ρ(DataCutFromDataSource1D, DataTarget1D_Dbl1)
ReDim Preserve Return_r(l)
Return_r(l) = r
'Debug.Print "r=" & r
'=================
Rem 获取最大相关系数值
If r > Return_rMax Then
Return_rMax = r
Return_Position = l '获得最大相关系数位置
End If
If r > 0.99 Then
'GoTo a1:
End If
a2:
Loop
a1:
End Function
```

'SerchOneLineOfPicture_Slow 搜索图片的一排（慢的版本）

Rem 搜索图片的一排（慢的版本）
Rem DataSource1D()=对比数据源
Rem DataTarget1D()=对比数据目标
Rem PictureSourceWidth=搜索原图宽度
Rem Return_rMax=返回相关系数最大值
Rem Return_Position=返回找到位置
Rem Return_r()=返回相关系数
Rem CalculateType=没有作用

```
Public Function SerchOneLineOfPicture_Slow(ByRef DataSource1D() As Long, _
ByRef DataTarget1D() As Long, ByVal PictureSourceWidth As Long, ByRef Return_rMax As Double,
_
ByRef Return_Position As Long, ByRef Return_r() As Double, Optional ByVal CalculateType As
Long = 1) As Double

Dim DataCutFromDataSource1D() As Double, n As Long, r As Double, l As Long, i As Long
Dim DataTarget1D_Dbl() As Double
```

```
Dim DataTarget1D_Dbl1() As Double
Array1D_CDataTypeLtoD DataTarget1D, DataTarget1D_Dbl
Do While l <= (UBound(DataSource1D) - UBound(DataTarget1D))
'=======================================================
Rem 超出横向边框直接跳过
If ((l + 1) Mod PictureSourceWidth) + (UBound(DataTarget1D) - LBound(DataTarget1D) + 1) >
PictureSourceWidth Then
l = l + 1
GoTo a2
End If
'=======================================================
'For i = 1 To 4
'SerchOneLineOfPictureSerchPixel DataSource1D, DataTarget1D, 5 * i, L, r
'If r < 0.1 * i Then
'GoTo a3
'End If
'Next i
SerchOneLineOfPictureSerchPixel DataSource1D, DataTarget1D, 5, l, r
If r < 0.1 Then
GoTo a3
End If
SerchOneLineOfPictureSerchPixel DataSource1D, DataTarget1D, (UBound(DataTarget1D) -
LBound(DataTarget1D) + 1), l, r
a3:
l = l + 1
If r > 0.4 Then
Dim L1 As Long
L1 = L1 + 1
ReDim Preserve Return_r(L1)
Return_r(L1) = r
End If
'Debug.Print "r=" & r
'=================
Rem 获取最大相关系数值
If r > Return_rMax Then
Return_rMax = r
Return_Position = l '获得最大相关系数位置
End If
'=================
Rem 加速程序
If r > 0.99 Then
GoTo a1:
End If
'=================
```

```
a2:
Loop
a1:
End Function
```

'SerchOneLineOfPictureSerchPixel 搜索目标数据按像素长度（多程序子程序）

Rem 搜索图片的一排的子程序
Rem DataSource1D()=对比数据源
Rem DataTarget1D()=对比数据目标
Rem SerchOneLineOfPictureSerchPixel=搜索长度按多少像素
Rem Position=返回找到位置
Rem Return_r=返回相关系数
Rem CalculateType=没有作用

```
Private Sub SerchOneLineOfPictureSerchPixel(ByRef DataSource1D() As Long, _
ByRef DataTarget1D() As Long, ByVal SerchOneLineOfPictureSerchPixel As Long, _
ByRef Position As Long, ByRef Return_r As Double, Optional ByVal CalculateType As Long = 1)
Dim DataCutFromDataSource1D() As Double
Dim i As Long, n As Long
Dim DataTarget1D_Dbl1() As Double
'DataTarget1D_Dbl1 = DataTarget1D_Dbl
For i = LBound(DataTarget1D) To LBound(DataTarget1D) + SerchOneLineOfPictureSerchPixel - 1
n = n + 1
ReDim Preserve DataCutFromDataSource1D(n)
DataCutFromDataSource1D(n) = DataSource1D(i + Position)
ReDim Preserve DataTarget1D_Dbl1(n)
DataTarget1D_Dbl1(n) = DataTarget1D(n)
'Debug.Print "DataCutFromDataSource1D(" + CStr(N) + ") =" +
CStr(DataCutFromDataSource1D(N))
'Debug.Print "DataTarget1D_Dbl(" + CStr(N) + ")=" + CStr(DataTarget1D_Dbl(N))
Next i
n = 0
'Position = Position + 1
Return_r = obj.ρ(DataCutFromDataSource1D, DataTarget1D_Dbl1)
End Sub
```

'SerchOneLineOfPicture_Nomal 采用 10 个像素作为像素头，搜索全图

Rem 采用 **10** 个像素作为像素头，搜索全图。
Rem 搜索到了目标 **r>0.1** 的时候逐行搜索，所有排数都大于 **0.1** 的时候
Rem 进行全像素搜索，逐行比较得到的 **r** 值全部大于 **0.3** 的时候，取其中 **R** 值最大的
Rem 程序像素头搜索不到 **EVE** 目标，**EVE** 做了像素处理。
Rem 增加了，如果 **r>0.3** 找不到目标的话降低 **r** 所需要的值。直到找到 **1** 个 **R** 值

```
Public Function SerchOneLineOfPicture_Nomal(ByRef DataSource1D() As Long, _
ByRef DataTarget2D() As Long, ByVal PictureSourceWidth As Long, ByRef Return_rMax As Double,
_
ByRef Return_Position As Long, ByRef Return_r() As Double, ByRef Return_rPosition() As Long,
Optional ByVal CalculateType As Long = 1) As Double

Dim DataCutFromDataSource1D() As Double, n As Long, r As Double, l As Long, i As Long
Dim DataTarget2D_Dbl() As Double
Dim DataTarget2D_Dbl1() As Double
'Dim DataTarget1D() As Long
Dim RL As Long

Dim Num As Long
Dim rCount() As Double
Dim UnFind As Double
Static Times As Long
Array2D_CDataTypeLtoD DataTarget2D, DataTarget2D_Dbl
a4:
Do While l <= (UBound(DataSource1D) - UBound(DataTarget2D, 1))
'=======================================================
Rem 超出横向边框直接跳过
If ((l + 1) Mod PictureSourceWidth) + (UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1) + 1) >
PictureSourceWidth Then
l = l + 1
GoTo a2
End If
'=======================================================
'For i = 1 To 4
'SerchOneLineOfPictureSerchPixel DataSource1D, DataTarget1D, 5 * i, L, r
'If r < 0.1 * i Then
'GoTo a3
'End If
'Next i
```

```
Dim m As Long
m = 1
SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D, 10, l, m, r
If r < 0.1 Then
        GoTo a3
Else
        For i = LBound(DataTarget2D, 2) + 1 To UBound(DataTarget2D, 2)
        OtherLinesPosition l, i, PictureSourceWidth, RL
            If RL <= UBound(DataSource1D) Then
            SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D, 10, RL, i, r
            Else
            GoTo a1
            End If
        If r < 0.1 Then
        GoTo a3
        End If
        Next i
End If
        For i = LBound(DataTarget2D, 2) + 1 To UBound(DataTarget2D, 2)
        OtherLinesPosition l, i, PictureSourceWidth, RL
        SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D,
(UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1) + 1), RL, i, r

            Num = Num + 1


            ReDim Preserve rCount(Num)
            rCount(Num) = r

            If r >= 0.3 - UnFind Then
            Dim L1 As Long
            L1 = L1 + 1
            ReDim Preserve Return_r(L1)
            ReDim Preserve Return_rPosition(L1)
            Return_r(L1) = r
            Return_rPosition(L1) = l
            End If
            '========================================
            Rem  如果一回合没有找到，第 2 回合的 r
            If r > 0 And UnFind >= 0.1 Then
                    If r > Return_rMax Then
                        Return_rMax = r
                    Return_Position = l '获得最大相关系数位置
                    Num = 0
```

```
                    End If
                End If
                '=========================================
            If r < 0.3 - UnFind Then
            Num = 0
            GoTo a3
            End If
            Next i
            Rem 上面剔除程序顺利运行的话跳出程序

            'GoTo a1:
                '=================
                Rem 获取最大相关系数值
                    r = Array1D_Max(rCount)
                    If r > Return_rMax Then
                        Return_rMax = r
                        Return_Position = I '获得最大相关系数位置
                        Num = 0
                    End If
              '=================


a3:
I = I + 1
If r > 0.99 Then
L1 = L1 + 1
ReDim Preserve Return_r(L1)
Return_r(L1) = r
End If
'Debug.Print "r=" & r
'=================
Rem 获取最大相关系数值
'If r > Return_rMax Then
'Return_rMax = r
'Return_Position = I '获得最大相关系数位置
'End If
'=================
Rem 加速程序
If r > 0.99 Then
'GoTo a1:
End If
'=================
a2:
Loop
```

```
a1:
Rem  如果第一回合没找到到进行第 2 回合搜索
If Return_rMax = 0 Then
Times = Times + 1
UnFind = 0.1 * Times
l = 0
If UnFind = 0.3 Then
MsgBox "没有找到"
Exit Function
End If
GoTo a4:
End If
End Function
```

Rem 多程序子程序

```
Public Function Array2D_CDataTypeLtoD(ByRef LngData() As Long, ByRef Return_DblData() As
Double) As Boolean
Dim i As Long, l As Long
ReDim Return_DblData(LBound(LngData, 1) To UBound(LngData, 1), LBound(LngData, 2) To
UBound(LngData, 2))
For i = LBound(LngData, 1) To UBound(LngData, 1)
For l = LBound(LngData, 2) To UBound(LngData, 2)
  Return_DblData(i, l) = CDbl(LngData(i, l))
  'Debug.Print "Return_DblData(" & i & "," & l & ")=" & Return_DblData(i, l)
Next l
Next i
Array2D_CDataTypeLtoD = True
End Function
```

'SerchOneLineOfPictureSerchPixel2D 搜索图片的一排的子程序，二维目标

Rem 搜索图片的一排的子程序
Rem DataSource1D()=对比数据源
Rem DataTarget2D()=对比数据目标
Rem SerchOneLineOfPictureSerchPixel=搜索长度按多少像素
Rem Position=返回找到位置
Rem PictureSerchLineNumber=搜索的目标的行数
Rem Return_r=返回相关系数
Rem CalculateType=没有作用

```
Private Sub SerchOneLineOfPictureSerchPixel2D(ByRef DataSource1D() As Long, _
ByRef DataTarget2D() As Long, ByVal SerchOneLineOfPictureSerchPixel As Long, _
```

```
ByRef Position As Long, ByVal PictureSerchLineNumber As Long, ByRef Return_r As Double,
Optional ByVal CalculateType As Long = 1)
Dim DataCutFromDataSource1D() As Double
Dim i As Long, n As Long
Dim DataTarget1D_Dbl1() As Double
'DataTarget1D_Dbl1 = DataTarget2D_Dbl
For i = LBound(DataTarget2D, 1) To LBound(DataTarget2D, 1) + SerchOneLineOfPictureSerchPixel -
1
n = n + 1
ReDim Preserve DataCutFromDataSource1D(n)
DataCutFromDataSource1D(n) = DataSource1D(i + Position)
ReDim Preserve DataTarget1D_Dbl1(n)
DataTarget1D_Dbl1(n) = DataTarget2D(n, PictureSerchLineNumber)
'Debug.Print "DataCutFromDataSource1D(" + CStr(N) + ") =" +
CStr(DataCutFromDataSource1D(N))
'Debug.Print "DataTarget2D_Dbl(" + CStr(N) + ")=" + CStr(DataTarget2D_Dbl(N))
Next i
n = 0
'Position = Position + 1
Return_r = obj.ρ(DataCutFromDataSource1D, DataTarget1D_Dbl1)
End Sub
```

Rem SerchOneLineOfPicture_Nomal 子程序

```
Public Function Array1D_Max(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next

Rem 16-8-29 防止只有一个数字
If LBound(Data) = UBound(Data) Then
Array1D_Max = Data(LBound(Data))
End If

For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) > cData(i + 1) Then
Array1D_Max = cData(i)
cData(i + 1) = cData(i)
Else
Array1D_Max = cData(i + 1)
End If
End If
```

```
Next
End Function
```

'SerchOneLineOfPictureThenAllPic_Slowest 采用 10 个像素作为像素头，搜索全图（最慢）

Rem 采用 **10** 个像素作为像素头，搜索全图。
Rem 搜索到了目标 **r>0.3** 的时候对宽为 **10** 像素的部分源图片进行 **r** 比较，当 **r>0.3** 的时候
Rem 进行全像素搜索，取其中 **R** 值最大的

```
Public Function SerchOneLineOfPictureThenAllPic_Slowest(ByRef DataSource1D() As Long, _
ByRef DataTarget2D() As Long, ByVal PictureSourceWidth As Long, ByRef Return_rMax As Double,
_
ByRef Return_Position As Long, ByRef Return_r() As Double, ByRef Return_rPosition() As Long,
Optional ByVal CalculateType As Long = 1) As Double

Dim DataCutFromDataSource1D() As Double, n As Long, r As Double, l As Long, i As Long
Dim DataTarget2D_Dbl() As Double
Dim DataTarget2D_Dbl1() As Double
'Dim DataTarget1D() As Long
Dim RL As Long
Dim rPosition As Long
Dim Num As Long
Dim rCount() As Double
Array2D_CDataTypeLtoD DataTarget2D, DataTarget2D_Dbl
rPosition = UBound(DataTarget2D, 2) * PictureSourceWidth + PictureSourceWidth -
UBound(DataTarget2D, 1)
Do While l <= (UBound(DataSource1D) - rPosition)
'=====================================================
Rem 超出横向边框直接跳过
If ((l + 1) Mod PictureSourceWidth) + (UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1) + 1) >
PictureSourceWidth Then
l = l + 1
GoTo a2
End If
'=====================================================
'For i = 1 To 4
'SerchOneLineOfPictureSerchPixel DataSource1D, DataTarget1D, 5 * i, L, r
'If r < 0.1 * i Then
'GoTo a3
'End If
'Next i
```

```
Dim m As Long
m = 1
SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D, 10, l, m, r
If r < 0.3 Then
        GoTo a3
Else
        SearchPicturePixel2D DataSource1D, DataTarget2D, 10, l, PictureSourceWidth, r
        If r < 0.3 Then
        GoTo a3
        End If
End If

SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), l, PictureSourceWidth, r
            If r >= 0.3 Then
            Dim L1 As Long
            L1 = L1 + 1
            ReDim Preserve Return_r(L1)
            ReDim Preserve Return_rPosition(L1)
            Return_r(L1) = r
            Return_rPosition(L1) = l
            End If

        If r < 0.3 Then
        Num = 0
        GoTo a3
        End If

        Rem  上面剔除程序顺利运行的话跳出程序

        'GoTo a1:
            '=================
            Rem  获取最大相关系数值
                    If r > Return_rMax Then
                        Return_rMax = r
                    Return_Position = l '获得最大相关系数位置
                    Num = 0
                    End If
              '=================



a3:
l = l + 1
If r > 0.99 Then
```

```
L1 = L1 + 1
ReDim Preserve Return_r(L1)
Return_r(L1) = r
End If
'Debug.Print "r=" & r
'=================
Rem 获取最大相关系数数值
'If r > Return_rMax Then
'Return_rMax = r
'Return_Position = l '获得最大相关系数位置
'End If
'=================
Rem 加速程序
If r > 0.99 Then
'GoTo a1:
End If
'=================
a2:
Loop
a1:
End Function
```

Rem 多程序子程序

Rem DataSource1D()=对比数据源

Rem DataTarget2D()=对比数据目标

Rem PictureSerchPixelWidth=搜索目标数据按像素长度

Rem Position=返回找到位置

Rem PictureSourceWidth=搜索的目标的宽度

Rem Return_r=返回相关系数

Rem CalculateType=没有作用

```
Private Sub SearchPicturePixel2D(ByRef DataSource1D() As Long, _
ByRef DataTarget2D() As Long, ByVal PictureSerchPixelWidth As Long, _
ByRef Position As Long, ByVal PictureSourceWidth As Long, ByRef Return_r As Double, Optional
ByVal CalculateType As Long = 1)
On Error GoTo a2:
Dim DataCutFromDataSource1D() As Double
Dim i As Long, n As Long, l As Long
Dim DataTarget1D_Dbl1() As Double
Dim rPosition As Long
For l = LBound(DataTarget2D, 2) To UBound(DataTarget2D, 2)
OtherLinesPosition Position, l, PictureSourceWidth, rPosition
    For i = LBound(DataTarget2D, 1) To LBound(DataTarget2D, 1) + PictureSerchPixelWidth - 1
        If (i + rPosition) > UBound(DataSource1D) Then
        GoTo a1 '超出源图范围直接结束
        End If
```

```
            n = n + 1
            ReDim Preserve DataCutFromDataSource1D(n)
            DataCutFromDataSource1D(n) = DataSource1D(i + rPosition)
            ReDim Preserve DataTarget1D_Dbl1(n)
            DataTarget1D_Dbl1(n) = DataTarget2D(i, l)
        Next i
Next l
n = 0
Return_r = obj.ρ(DataCutFromDataSource1D, DataTarget1D_Dbl1)
a1:
Exit Sub
a2:
MsgBox "SearchPicturePixel2D 出错"
End Sub
```

'SerchOneLineOfPictureThenAllPic_Slower 采用 10 个像素作为像素头，搜索全图（较慢）

Rem 采用 10 个像素作为像素头，搜索全图。
Rem 搜索到了目标 r>0.3 的时候对宽为 10 像素的部分源图片进行 r 比较，当 r>0.3 的时候
Rem 进行全像素搜索，取其中 R 值最大的

```
Public Function SerchOneLineOfPictureThenAllPic_Slower(ByRef DataSource1D() As Long, _
ByRef DataTarget2D() As Long, ByVal PictureSourceWidth As Long, ByRef Return_rMax As Double,
_
ByRef Return_Position As Long, ByRef Return_r() As Double, ByRef Return_rPosition() As Long,
Optional ByVal CalculateType As Long = 1) As Double
Dim DataCutFromDataSource1D() As Double, n As Long, r As Double, l As Long, i As Long
Dim DataTarget2D_Dbl() As Double
Dim DataTarget2D_Dbl1() As Double
'Dim DataTarget1D() As Long
Dim RL As Long
Dim rPosition As Long
Dim Num As Long
Dim rCount() As Double

Array2D_CDataTypeLtoD DataTarget2D, DataTarget2D_Dbl
rPosition = UBound(DataTarget2D, 2) * PictureSourceWidth + PictureSourceWidth -
UBound(DataTarget2D, 1)
Do While l <= (UBound(DataSource1D) - rPosition)
'=====================================================
Rem 超出横向边框直接跳过
If ((l + 1) Mod PictureSourceWidth) + (UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1) + 1) >
```

```
PictureSourceWidth Then
l = l + 1
GoTo a2
End If
'=====================================================
'For i = 1 To 4
'SerchOneLineOfPictureSerchPixel DataSource1D, DataTarget1D, 5 * i, L, r
'If r < 0.1 * i Then
'GoTo a3
'End If
'Next i

Dim m As Long
m = 1
SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D, 10, l, m, r
If r < 0.3 Then
        GoTo a3
Else
        SearchPicturePixel2D DataSource1D, DataTarget2D, 10, l, PictureSourceWidth, r
        If r < 0.3 Then
        GoTo a3
        End If
End If

SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), l, PictureSourceWidth, r
            If r >= 0.3 Then
            Dim L1 As Long
            L1 = L1 + 1
            ReDim Preserve Return_r(L1)
            ReDim Preserve Return_rPosition(L1)
            Return_r(L1) = r
            Return_rPosition(L1) = l
            End If

        If r < 0.3 Then
        Num = 0
        GoTo a3
        End If

        Rem 上面剔除程序顺利运行的话跳出程序

        'GoTo a1:
            '=================
```

```
                Rem  获取最大相关系数值
                    If r > Return_rMax Then
                        Return_rMax = r
                        Return_Position = l '获得最大相关系数位置
                        Num = 0
                    End If
            '=================


a3:
l = l + 1
If r > 0.99 Then
L1 = L1 + 1
ReDim Preserve Return_r(L1)
Return_r(L1) = r
End If
'Debug.Print "r=" & r
'=================
Rem  获取最大相关系数值
'If r > Return_rMax Then
'Return_rMax = r
'Return_Position = l '获得最大相关系数位置
'End If
'=================
Rem  加速程序
If r > 0.99 Then
'GoTo a1:
End If
'=================
a2:
Loop
a1:
End Function
```

'SerchOneLineOfPictureThenAllPic_Slow 采用 10 个像素作为像素头，搜索全图（慢）

Rem 采用 10 个像素作为像素头，搜索全图。
Rem 搜索到了目标 r>0.3 的时候对宽为 10 像素的部分源图片进行 r 比较，当 r>0.3 的时候
Rem 进行全像素搜索，取其中 R 值最大的

```
Public Function SerchOneLineOfPictureThenAllPic_Slow(ByRef DataSource1D() As Long, _
ByRef DataTarget2D() As Long, ByVal PictureSourceWidth As Long, ByRef Return_rMax As Double,
```

```
_
ByRef Return_Position As Long, ByRef Return_r() As Double, ByRef Return_rPosition() As Long,
Optional ByVal CalculateType As Long = 1) As Double
Dim DataCutFromDataSource1D() As Double, n As Long, r As Double, l As Long, i As Long
Dim DataTarget2D_Dbl() As Double
Dim DataTarget2D_Dbl1() As Double
Dim RL As Long
Dim rPosition As Long
Dim Num As Long
Dim rCount() As Double
Static Times As Double
Array2D_CDataTypeLtoD DataTarget2D, DataTarget2D_Dbl
rPosition = UBound(DataTarget2D, 2) * PictureSourceWidth + PictureSourceWidth -
UBound(DataTarget2D, 1)
a4:
Do While l <= (UBound(DataSource1D) - rPosition)
'=======================================================
Rem 超出横向边框直接跳过
If ((l + 1) Mod PictureSourceWidth) + (UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1) + 1) >
PictureSourceWidth Then
l = l + 1
GoTo a2
End If
'=======================================================

Dim m As Long
m = 1
SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D, 10, l, m, r
If r < 0.1 Then
        GoTo a3
Else

        For i = LBound(DataTarget2D, 2) + 1 To UBound(DataTarget2D, 2)
        OtherLinesPosition l, i, PictureSourceWidth, RL
            If RL <= UBound(DataSource1D) Then
            SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D, 10, RL, i, r
            Else
            GoTo a1
            End If
        If r < 0.1 Then
        GoTo a3
        End If
        Next i
End If
```

```
SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), l, PictureSourceWidth, r
                'If r >= 0.2 - Times Then
                Dim L1 As Long
                L1 = L1 + 1
                ReDim Preserve Return_r(L1)
                ReDim Preserve Return_rPosition(L1)
                Return_r(L1) = r
                Return_rPosition(L1) = l
                'End If

            If r < 0.3 - Times Then
            Num = 0
            GoTo a3
            End If

            Rem  上面剔除程序顺利运行的话跳出程序

            'GoTo a1:
                '=================
                Rem  获取最大相关系数值
                        If r > Return_rMax Then
                             Return_rMax = r
                            Return_Position = l '获得最大相关系数位置
                            Num = 0
                        End If
                '=================


a3:
l = l + 1
If r > 0.99 Then
L1 = L1 + 1
ReDim Preserve Return_r(L1)
Return_r(L1) = r
End If
'Debug.Print "r=" & r
'=================
Rem  获取最大相关系数值
'If r > Return_rMax Then
'Return_rMax = r
'Return_Position = l '获得最大相关系数位置
'End If
```

```
'=================
Rem 加速程序
If r > 0.99 Then
'GoTo a1:
End If
'=================
a2:
Loop
a1:
If Return_rMax = 0 Then
Times = Times + 0.1
l = 0
If Times = 0.2 Then
MsgBox "没有找到"
Exit Function
End If
GoTo a4:
End If
Confirm Return_r, Return_rPosition, Return_rMax, Return_Position
End Function
```

```
Private Sub Confirm(ByRef r() As Double, ByRef Position() As Long, _
ByVal Return_rMax As Double, ByVal Return_Position As Long)
'SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D, 10, Return_Position, 1, r
End Sub
```

'SearchHead 搜索头文件

SerchOneLineOfPictureThenAllPic_Normal 子程序

Rem SerchOneLineOfPictureThenAllPic_Normal 子程序
Rem DataSource1D()=搜索源图片 1 维数据
Rem DataTarget2D()=搜索目标 2 维数据
Rem HeadSize=搜索头的宽度
Rem LboundHeadLine =头文件下限
Rem UboundHeadLine=头文件上限
Rem l=搜索目标像素位置
Rem PictureSourceWidth=搜索目标的宽度
Rem Return_Fix=Return_r>0.5 的时候是 True
Rem Return_r=返回相关系数

```
Private Sub SearchHead(ByRef DataSource1D() As Long, ByRef DataTarget2D() As Long, ByVal
HeadSize As Long, ByVal LboundHeadLine As Long, _
ByVal UboundHeadLine As Long, ByVal l As Long, ByVal PictureSourceWidth As Long, ByRef
```

```
Return_Fix As Boolean, ByRef Return_r As Double)
Dim m As Long, r As Double, RL As Long
m = Int((UboundHeadLine - LboundHeadLine) / 2) + LboundHeadLine
OtherLinesPosition l, m, PictureSourceWidth, RL
SerchOneLineOfPictureSerchPixel2D DataSource1D, DataTarget2D, HeadSize, RL, m, Return_r
If Return_r < 0.5 Then
Return_Fix = False
Else
Return_Fix = True
End If
End Sub
```

'SerchOneLineOfPictureThenAllPic_Normal 搜索图片先一排然后全部

Rem 使用关键：头文件位置大小是搜索成败的关键，尽量找不变的像素！
Rem 采用 HeadSize 个像素作为像素头，搜索全图。
Rem 搜索到了目标在 LboundHeadLine 和 UboundHeadLine，r>0.3 的时候，对宽为 HeadSize 像素的部分源图片进行 r 比较，当 r>0.3 的时候
Rem 进行全像素搜索，取其中 R 值最大的
Rem 如果像素头有（UboundHeadLine-LboundHeadLine)/2 个连续都大于 0.8 那么认为找到，进行全图 R 值计算，如果左边和右边一像素的 R 值差在 0.1 以下，认为找到并且跳出程序
Rem DataSource1D()搜索源图片 1 维数据
Rem DataTarget2D()搜索目标 2 维数据
Rem PictureSourceWidth 搜索源图片的宽度
Rem HeadSize 搜索头的宽度
Rem LboundHeadLine 搜索线下标 UboundHeadLine 搜索线上标，在图片上认为 r 值较大的区域，也就是图片不变的区域
Rem Return_rMax 返回最大的 r 值，返回 Return_Position 最大值位置
Rem Return_r()返回记录的 R 值，Return_rPosition()对应的位置
Rem UnChangePartOfTarget(1)=Left=HeadPosition
Rem UnChangePartOfTarget(2)=right
Rem UnChangePartOfTarget(3)=top= LboundHeadLine
Rem UnChangePartOfTarget(4)=buttom= UboundHeadLine

```
Public Function SerchOneLineOfPictureThenAllPic_Normal(ByRef DataSource1D() As Long, _
ByRef DataTarget2D() As Long, ByVal PictureSourceWidth As Long, ByRef UnChangePartOfTarget()
As Long, _
ByRef Return_rMax As Double, ByRef Return_Position As Long, _
ByRef Return_r() As Double, ByRef Return_rPosition() As Long, _
Optional ByVal CalculateType As Long = 1) As String
    Dim DataCutFromDataSource1D() As Double, n As Long, r As Double, l As Long, i As Long
    Dim DataTarget2D_Dbl() As Double
```

```
    Dim DataTarget2D_Dbl1() As Double
    Dim RL As Long
    Dim rPosition As Long
    'Dim Num As Long
    Dim rCount() As Double
    Static Times As Double
    Static L2 As Long
    Dim Find As Boolean
    Dim L3 As Long, L1 As Long
    Dim Return_Fix As Boolean, Return_OutDataSource1D As Boolean
    Dim JumpOverFullPicSerach As Boolean, JumpOverBecouseOutOfLineRange As Boolean

    Dim HeadPosition As Long, HeadSize As Long, LboundHeadLine As Long, UboundHeadLine
As Long
    HeadPosition = UnChangePartOfTarget(1) '暂时没有使用
    HeadSize = UnChangePartOfTarget(2) - UnChangePartOfTarget(1)
    LboundHeadLine = UnChangePartOfTarget(3)
    UboundHeadLine = UnChangePartOfTarget(4)

Array2D_CDataTypeLtoD DataTarget2D, DataTarget2D_Dbl
Rem rPosition 是一个数据块的长度
rPosition = UBound(DataTarget2D, 2) * PictureSourceWidth + PictureSourceWidth -
UBound(DataTarget2D, 1)
a4:
Do While l <= (UBound(DataSource1D) - rPosition)
'=====================================================
Rem  超出横向边框直接跳过
If ((l + 1) Mod PictureSourceWidth) + (UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1) + 1) >
PictureSourceWidth Then
l = l + 1 '一个一个像素向后移动
'GoTo a2
JumpOverBecouseOutOfLineRange = True
End If
'=====================================================
If JumpOverBecouseOutOfLineRange = False Then
'=====================================================
Rem  搜索头文件
SearchHead DataSource1D, DataTarget2D, HeadSize, LboundHeadLine, UboundHeadLine, l,
PictureSourceWidth, Return_Fix, r
If Return_Fix = False Then 'r<0.5 的时候 Return_Fix = False
        JumpOverFullPicSerach = True 'GoTo a3
Else
      '================================================
        Rem  头文件找到，进一步搜索：SearchNext
```

```
            SearchNext DataSource1D, DataTarget2D, HeadSize, LboundHeadLine,
UboundHeadLine, l, PictureSourceWidth, Find, L3, r, Return_rMax, Return_OutDataSource1D,
JumpOverFullPicSerach
            If Return_OutDataSource1D = True Or Find = True Then '超出了搜索图片范围
            Return_rMax = r
            Return_Position = L3 '
            RecordSearchResult DataSource1D, DataTarget2D, L1, l, r, Times, PictureSourceWidth,
Return_r, Return_rMax, Return_Position, Return_rPosition, JumpOverFullPicSerach
            Exit Do 'GoTo a1 '跳出 LOOP 循环
            End If
            '================================================
End If
'======================================================
If JumpOverFullPicSerach = False Then
'Debug.Print "全图搜索前最后一线 r=" & Round(r, 3)
Rem  全图搜索
SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), l, PictureSourceWidth, r
RecordSearchResult DataSource1D, DataTarget2D, L1, l, r, Times, PictureSourceWidth, Return_r,
Return_rMax, Return_Position, Return_rPosition, JumpOverFullPicSerach
End If
'a3:
JumpOverFullPicSerach = False
L2 = 0
l = l + 1
'================================
Rem  加速程序
            If r > 0.99 Then
            Return_rMax = r
            Return_Position = l '
            Exit Do
            End If
'================================
End If
JumpOverBecouseOutOfLineRange = False
Loop 'a2:
'a1:
Rem   最终结果大于 0.3,没有找到降低条件，继续寻找.但是基本 R 都大于 0.55
If Return_rMax = 0 Then
      Times = Times + 0.1
      l = 0
                  If Times = 0.2 Then
                  MsgBox "没有找到"
```

```
                Exit Function
                End If
        GoTo a4:
End If
'Confirm Return_r, Return_rPosition, Return_rMax, Return_Position
End Function
```

Rem SixPointSearch 子程序
Rem 得到数据中最大值
Rem 例如：110，12，130，1111,1111 最大值为 1111
Rem Data()原始数据
Rem 返回 Return_Which() 最大值的位置.例如：110，12，130，1111，1111 返回 4，5
Rem Array1D_Max_Which 返回最大值数值

```
Public Function Array1D_Max_Which(ByRef Data() As Double, ByRef Return_Which() As Long) As
Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next

Rem 16-8-29 防止只有一个数字
If LBound(Data) = UBound(Data) Then
Array1D_Max_Which = Data(LBound(Data))
ReDim Preserve Return_Which(1) As Long
Return_Which(1) = LBound(Data)
Exit Function
End If

For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) > cData(i + 1) Then
Array1D_Max_Which = cData(i)
cData(i + 1) = cData(i)
Else
Array1D_Max_Which = cData(i + 1)
End If
End If
Next
'************************************************
'遍历数组求得最大值的位置
Dim n As Long
n = 0
For i = LBound(Data()) To UBound(Data())
```

```
If (Array1D_Max_Which = Data(i)) Then
n = n + 1
ReDim Preserve Return_Which(n) As Long
Return_Which(n) = i
End If
Next i
End Function
```

'SixPointSearch 找到 4 个线 r 都大于 0.9 的时候，判定基本找到最终点，然后就依照这个点寻找旁边 6 个点的 R 全图值，找到最大的 R

Rem 找到 4 个线 r 都大于 0.9 的时候，判定基本找到最终点。
Rem 然后就依照这个点寻找旁边 6 个点的 R 全图值，找到最大的 R

```
Private Sub SixPointSearch(ByRef DataSource1D() As Long, ByRef DataTarget2D() As Long, _
ByVal PictureSerchPixelWidth As Long, ByRef Position As Long, _
ByVal PictureSourceWidth As Long, ByRef Return_r As Double, ByRef Return_Position As Long, _
ByRef Return_Find As Boolean)
Dim r(1 To 6) As Double
Dim RL As Long
Dim Which() As Long
Static RecordJumpNum As Double
OtherLinesPosition Position, 2, PictureSourceWidth, RL
Rem 这个点的 R 值
SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), Position, PictureSourceWidth, r(1)
Rem 左边一个点 R 值
If r(1) < 0.4 Then
Return_r = r(1)
Return_Position = Position
RecordJumpNum = RecordJumpNum + 1
Debug.Print "r=" & Round(r(1), 3) & "跳出 6 点查询" & RecordJumpNum
GoTo a1:
End If
Debug.Print "进行 6 点查询  Position=" & Position
If Position > 1 Then
SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), Position - 1, PictureSourceWidth, r(2)
SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), RL - 1, PictureSourceWidth, r(5)
End If
```

```
Rem  右边一个点的 R 值
If ((Position + 2) Mod PictureSourceWidth) + (UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1)
+ 1) <= PictureSourceWidth Then
SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), Position + 1, PictureSourceWidth, r(3)
SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), RL + 1, PictureSourceWidth, r(6)
End If
SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), RL, PictureSourceWidth, r(4)
Array1D_Max_Which r, Which
Return_r = r(Which(LBound(Which)))
If Which(LBound(Which)) = 1 Then
Return_Position = Position
Debug.Print "Position=" & Position & " r =" & Round(r(1), 3)
Debug.Print "Position=" & Position - 1 & " r =" & Round(r(2), 3)
Debug.Print "Position=" & Position + 1 & " r =" & Round(r(3), 3)
End If
If Which(LBound(Which)) = 2 Then
Return_Position = Position - 1
End If
If Which(LBound(Which)) = 3 Then
Return_Position = Position + 1
End If
If Which(LBound(Which)) = 4 Then
Return_Position = RL
Debug.Print "Position=" & RL & " r =" & Round(r(4), 3)
Debug.Print "Position=" & RL - 1 & " r =" & Round(r(5), 3)
Debug.Print "Position=" & RL + 1 & " r =" & Round(r(6), 3)
End If
If Which(LBound(Which)) = 5 Then
Return_Position = RL - 1
End If
If Which(LBound(Which)) = 6 Then
Return_Position = RL + 1
End If
Return_Find = False
If Return_r > 0.5 Then
    If Which(LBound(Which)) = 1 Then
        If (r(2) - r(3)) < 0.1 Then
            'If (r(1) - r(2)) > 0.2 And (r(1) - r(2)) < 0.5 Then
            Return_Find = True
            'End If
        End If
```

```
    End If
    If Which(LBound(Which)) = 4 Then
        If (r(5) - r(6)) < 0.1 Then
            ' If (r(4) - r(5)) > 0.2 And (r(4) - r(5)) < 0.5 Then
              Return_Find = True
            ' End If
        End If
    End If
End If
a1:
End Sub
```

'SearchNext 头文件找到，进一步搜索

SerchOneLineOfPictureThenAllPic_Normal 子程序

Rem 头文件找到，进一步搜索
Rem DataSource1D()搜索源图片 **1** 维数据
Rem DataTarget2D()搜索目标 **2** 维数据
Rem HeadSize 搜索头的宽度
Rem LboundHeadLine 搜索线下标 **UboundHeadLine** 搜索线上标，在图片上认为 **r** 值较大的区域，也就是图片不变的区域
Rem l=搜索目标像素位置
Rem PictureSourceWidth=搜索目标图片宽度
Rem Return_Find=是否找到目标
Rem Return_rMax=返回最大的 **r** 值，返回 **Return_Position=**最大值位置
Rem Return_r()=返回记录的 **R** 值，**Return_rPosition()=**对应的位置
Rem Return_OutDataSource1D=超出搜索目标图片范围
Rem JumpOverFullPicSerach=跳出全图片搜索

```
Private Sub SearchNext(ByRef DataSource1D() As Long, ByRef DataTarget2D() As Long, _
ByVal HeadSize As Long, ByVal LboundHeadLine As Long, _
ByVal UboundHeadLine As Long, ByVal l As Long, ByVal PictureSourceWidth As Long, _
ByRef Return_Find As Boolean, ByRef Return_Position As Long, ByRef Return_r As Double, _
ByRef Return_rMax As Double, ByRef Return_OutDataSource1D As Boolean, _
ByRef JumpOverFullPicSerach As Boolean)
Dim RL As Long
Dim L3 As Long
Static AboveRNumber As Long
Static UnderRNumber As Long
Dim i As Long
Dim TotalLines As Long
  For i = LBound(DataTarget2D, 2) To UBound(DataTarget2D, 2)
        OtherLinesPosition l, i, PictureSourceWidth, RL
```

```
                                    If RL <= UBound(DataSource1D) Then
                                    SerchOneLineOfPictureSerchPixel2D DataSource1D,
DataTarget2D, HeadSize, RL, i, Return_r
                                                If Return_r > 0.8 Then '看有多少条线 r 能
大于 0.8
                                                AboveRNumber = AboveRNumber + 1

If AboveRNumber > (UboundHeadLine - LboundHeadLine) Then

MsgBox "超过"

End If
                                                Else
                                                If Return_r < 0.3 Then
                                                UnderRNumber = UnderRNumber + 1
                                                End If
                                                End If
                                                TotalLines = UBound(DataTarget2D, 2) -
LBound(DataTarget2D, 2)
                                                If UnderRNumber >= TotalLines -
Int((UboundHeadLine - LboundHeadLine) / 2) + 1 Then

                                                    JumpOverFullPicSerach = True
                                                    Return_Find = False
                                                    UnderRNumber = 0
                                                    AboveRNumber = 0
                                                    Exit Sub

                                                End If
                                                If AboveRNumber >= Int((UboundHeadLine
- LboundHeadLine) / 2) Then '线数 r>0.8 数量大于给定数的一半然后进行 6 点查询
                                                            'SixPointSearch DataSource1D,
DataTarget2D, (UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1) + 1), L, PictureSourceWidth,
Return_r, L3, Find

                                                            SixPointSearch DataSource1D,
DataTarget2D, (UBound(DataTarget2D, 1) - LBound(DataTarget2D, 1) + 1), l, PictureSourceWidth,
Return_r, L3, Return_Find

                                                            If Return_r > Return_rMax Then
'6 点查询的最大 R 值和以前最大 R 值比较，大的存入最大值

                                                            Return_rMax = Return_r
                                                            Return_Position = L3 '获得最大
```

```
相关系数位置
                                                    AboveRNumber = 0
                                                    UnderRNumber = 0

                                                    Debug.Print "Return_rMax=" &
Round(Return_rMax, 3)

                                                    End If
                                                    If Return_Find = True Then '6 点
查询找到目标
                                                    Debug.Print "找到目标"
                                                    Return_rMax = Return_r
                                                    Return_Position = L3 '获得最大
相关系数位
                                                    Exit Sub
                                                    End If
                                                     'GoTo a5 '有 3 个大于 0.9 的线
就直接判定找到目标
                                            End If

                                    Else
                                    'GoTo a1
                                    Return_OutDataSource1D = True
                                    AboveRNumber = 0
                                    UnderRNumber = 0
                                    End If

        Next i
        AboveRNumber = 0
        UnderRNumber = 0
End Sub
```

Rem SerchOneLineOfPictureThenAllPic_Normal 子程序

Rem 记录搜索结果

```
Private Sub RecordSearchResult(ByRef DataSource1D() As Long, _
ByRef DataTarget2D() As Long, ByRef RecordNumber As Long, ByRef Position As Long, ByRef r As
Double, _
ByVal Times As Double, ByVal PictureSourceWidth As Long, ByRef Return_r() As Double, ByRef
Return_rMax As Double, ByRef Return_Position As Long, _
ByRef Return_rPosition() As Long, ByRef JumpOverFullPicSerach As Boolean, Optional ByVal
CalculateType As Long = 1)
If r >= 0.3 - Times Then '最终结果大于 0.3
                RecordNumber = RecordNumber + 1
                ReDim Preserve Return_r(RecordNumber)
                ReDim Preserve Return_rPosition(RecordNumber)
```

```
                Return_r(RecordNumber) = r
                Return_rPosition(RecordNumber) = Position
                SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), Position - 1, PictureSourceWidth, r
                RecordNumber = RecordNumber + 1
                ReDim Preserve Return_r(RecordNumber)
                ReDim Preserve Return_rPosition(RecordNumber)
                Return_r(RecordNumber) = r
                Return_rPosition(RecordNumber) = Position
                SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), Position + 1, PictureSourceWidth, r
                RecordNumber = RecordNumber + 1
                ReDim Preserve Return_r(RecordNumber)
                ReDim Preserve Return_rPosition(RecordNumber)
                Return_r(RecordNumber) = r
                Return_rPosition(RecordNumber) = Position
                SearchPicturePixel2D DataSource1D, DataTarget2D, (UBound(DataTarget2D, 1) -
LBound(DataTarget2D, 1) + 1), Position, PictureSourceWidth, r
                End If
                If r < 0.3 - Times Then
                JumpOverFullPicSerach = True
                End If
                Rem 上面剔除程序顺利运行的话跳出程序
                '================================
                Rem 获取最大相关系数值
                 If JumpOverFullPicSerach = False Then
                        If r > Return_rMax Then
                              Return_rMax = r
                              Return_Position = Position '获得最大相关系数位置
                              'Num = 0
                        End If
                End If
                '================================
End Sub
```

28. ShowReportTxt

详细说明：

函数或方法名称	作用
ReportWhiteInTxt	报告写在 Txt 之中

代码：

```
Option Explicit
'**************************************************************************
'名称 = ReportWhiteInTxt.Cls
'作者 = 张宏
'最后修改时间 = 2022 年 1 月 1 日
'简介 = 把报告显示在 TXT 中
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=Nomarl Slow Slower Slowest 是用的横向长条图片
'操作历史：
'**************************************************************************
```

'ReportWhiteInTxt 报告写在 Txt 之中

Rem 把报告写入 Txt 文件中，询问是否打开
Rem Report=输入数据
Rem TxtFileName=文件名称地址

```
Public Sub ReportWhiteInTxt(ByVal TxtFileName As String, ByVal Report As String)
On Error Resume Next
Dim Str As Variant
Dim Answer As Variant
Open TxtFileName For Output As #1
    Str = Report
Print #1, Str
Close #1
Answer = MsgBox("报告已经生成位置在:" & vbCrLf & TxtFileName & vbCrLf & "是否打开？",
vbYesNo)
If Answer = vbYes Then
Shell "Notepad " & Chr(34) & TxtFileName & Chr(34), 1
End If
End Sub
```

29. SimulateOperation

详细说明：

函数或方法名称	作用
FindAWindow	通过标题找窗体句柄
FindASubWindow	通过标题和父窗体句柄找下级窗体句柄
SendAMessageMouseDownAndUp	发送模拟的按下和抬起鼠标左键
SendAMessageMouseMove	发送模拟的鼠标移动
PostAMessageMouseDownAndUp	发送模拟的按下和抬起鼠标左键
GetHighWord	获取一个 32 位整数的高 16 位（不知何用）
GetLowWord	获取一个 32 位整数的低 16 位（不知何用）
MouseEventMoveAndClick	当前显示器分辨率为基础，移动和点击鼠标
SendAMessageMouseWheelScRollDown	鼠标滚轮向下
SendAMessageMouseWheel	鼠标滚轮
SendAMessageKeyPress	模拟按键（未使用）
PostAMessageKeyPress	模拟按键
ShowWindow	突前显示当前窗体

代码：

```
Option Explicit
Option Base 1
'**************************************************************************
'名称 = SimulateOperation.Cls
'作者 = 张宏
'最后修改时间 = 2022 年 1 月 1 日
'简介 = 模拟操作
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=
'操作历史：
'**************************************************************************
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
'=====================================================
'mouse_event
Private Declare Sub mouse_event Lib "user32" (ByVal dwFlags As Long, ByVal dx As Long, ByVal
dy As Long, ByVal cButtons As Long, ByVal dwExtraInfo As Long)
Private Const MOUSEEVENTF_ABSOLUTE = &H8000 '指定鼠标使用绝对坐标系，此时，屏幕在
水平和垂直方向上均匀分割成 65535×65535 个单元
Private Const MOUSEEVENTF_MOVE = &H1 '移动鼠标
```

```
Private Const MOUSEEVENTF_LEFTDOWN = &H2 '模拟鼠标左键按下
Private Const MOUSEEVENTF_LEFTUP = &H4 '模拟鼠标左键抬起
Private Const SW = 1920
Private Const SH = 1080
'=====================================================
'API PostMessage 声明
Private Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As Long,
ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
'常量声明
'=====================================================
'SetForegroundWindow
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hwnd As Long) As Long
'=====================================================
'FindWindow:
'寻找窗口列表中第一个符合指定条件的顶级窗口
'（在 vb 里使用：FindWindow 最常见的一个用途是获得 ThunderRTMain 类的隐藏窗口的句
柄；_
'该类是所有运行中 vb 执行程序的一部分。'
'获得句柄后，可用 api 函数 GetWindowText 取得这个窗口的名称；该名也是应用程序的标题）
Private Declare Function FindWindow Lib "user32.dll" Alias "FindWindowA" (ByVal lpClassName
As String, ByVal lpWindowName As String) As Long
'=====================================================
'FindWindowEx:
'在窗口列表中寻找与指定条件相符的第一个子窗口
Private Declare Function FindWindowEx Lib "user32.dll" Alias "FindWindowExA" (ByVal
hWndParent As Long, ByVal hWndChildAfter As Long, ByVal lpszClass As String, ByVal lpszWindow
As String) As Long
'=====================================================
'附加函数
'SendMessage:
'如果不理解 Windows 的消息机制，那么这个函数不是很好理解，
'最简单的说法就是给一个窗口发送消息
'Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long,
ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long,
ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
Private Const WM_CLOSE = &H10
Private Const WM_SYSCOMMAND = &H112
Private Const SC_CLOSE = &HF060&
Private Const SC_MINIMIZE = &HF020&
Private Const cch = 255
Private Const WM_MOUSEMOVE = &H200
Private Const WM_LBUTTONDOWN = &H201
Private Const WM_LBUTTONUP = &H202
```

```
Private Const WM_MBUTTONDBLCLK = &H209
Private Const WM_LBUTTONDBLCLK = &H203
Private Const WM_MBUTTONDOWN = &H207
Private Const WM_MBUTTONUP = &H208
Private Const WM_RBUTTONDBLCLK = &H206
Private Const WM_RBUTTONDOWN = &H204
Private Const WM_RBUTTONUP = &H205
Private Const WM_MOUSEWheel = &H20A
Private Const EM_SCROLL = &HB5
Private Const WM_KEYDOWN = &H100 '按下
Private Const WM_KEYUP = &H101 '释放
Private Type Point
X As Long
Y As Long
End Type
Private Const SM_CXHSCROLL = 21
Private Const GWL_STYLE = (-16)
Private Const WS_HSCROLL = &H100000
Private Const WS_VSCROLL = &H200000
Private Const SB_BOTH = 3
Private Const SB_HORZ = 0
Private Const SB_VERT = 1
'以下以 SB_开头的是用户的滚动请求
Private Const SB_LINEDOWN = 1
Private Const SB_LINELEFT = 0
Private Const SB_LINERIGHT = 1
Private Const SB_LINEUP = 0
Private Const SB_PAGERIGHT = 3
Private Const SB_PAGELEFT = 2
Private Const SB_PAGEDOWN = 3
Private Const SB_PAGEUP = 2
Private Const SB_ENDSCROLL = 8
Private Const SB_THUMBPOSITION = 4
Private Const SB_THUMBTRACK = 5
Private Const GWL_WNDPROC = (-4)
Private Const WM_HSCROLL = &H114
Private Const WM_VSCROLL = &H115
Private Const EM_LINESCROLL = &HB6 '以行为单位
Private Const WM_SETFOCUS = &H7

Private Const VK_LBUTTON = &H1
Private Const VK_RBUTTON = &H2
Private Const VK_CANCEL = &H3
Private Const VK_MBUTTON = &H4
```

```
Private Const VK_BACK = &H8
Private Const VK_TAB = &H9
Private Const VK_CLEAR = &HC
Private Const VK_RETURN = &HD
Private Const VK_SHIFT = &H10
Private Const VK_CONTROL = &H11
Private Const VK_MENU = &H12
Private Const VK_PAUSE = &H13
Private Const VK_CAPITAL = &H14
Private Const VK_ESCAPE = &H1B
Private Const VK_SPACE = &H20
Private Const VK_PRIOR = &H21
Private Const VK_NEXT = &H22
Private Const VK_END = &H23
Private Const VK_HOME = &H24
Private Const VK_LEFT = &H25
Private Const VK_UP = &H26
Private Const VK_RIGHT = &H27
Private Const VK_DOWN = &H28
Private Const VK_Select = &H29
Private Const VK_PRINT = &H2A
Private Const VK_EXECUTE = &H2B
Private Const VK_SNAPSHOT = &H2C
Private Const VK_Insert = &H2D
Private Const VK_Delete = &H2E
Private Const VK_HELP = &H2F
Private Const VK_0 = &H30
Private Const VK_1 = &H31
Private Const VK_2 = &H32
Private Const VK_3 = &H33
Private Const VK_4 = &H34
Private Const VK_5 = &H35
Private Const VK_6 = &H36
Private Const VK_7 = &H37
Private Const VK_8 = &H38
Private Const VK_9 = &H39
Private Const VK_A = &H41
Private Const VK_B = &H42
Private Const VK_C = &H43
Private Const VK_D = &H44
Private Const VK_E = &H45
Private Const VK_F = &H46
Private Const VK_G = &H47
Private Const VK_H = &H48
```

```
Private Const VK_I = &H49
Private Const VK_J = &H4A
Private Const VK_K = &H4B
Private Const VK_L = &H4C
Private Const VK_M = &H4D
Private Const VK_N = &H4E
Private Const VK_O = &H4F
Private Const VK_P = &H50
Private Const VK_Q = &H51
Private Const VK_R = &H52
Private Const VK_S = &H53
Private Const VK_wses = &H54
Private Const VK_U = &H55
Private Const VK_V = &H56
Private Const VK_W = &H57
Private Const VK_X = &H58
Private Const VK_Y = &H59
Private Const VK_Z = &H5A
Private Const VK_STARTKEY = &H5B
Private Const VK_CONTEXTKEY = &H5D
Private Const VK_NUMPAD0 = &H60
Private Const VK_NUMPAD1 = &H61
Private Const VK_NUMPAD2 = &H62
Private Const VK_NUMPAD3 = &H63
Private Const VK_NUMPAD4 = &H64
Private Const VK_NUMPAD5 = &H65
Private Const VK_NUMPAD6 = &H66
Private Const VK_NUMPAD7 = &H67
Private Const VK_NUMPAD8 = &H68
Private Const VK_NUMPAD9 = &H69
Private Const VK_MULTIPLY = &H6A
Private Const VK_ADD = &H6B
Private Const VK_SEPARATOR = &H6C
Private Const VK_SUBTRACT = &H6D
Private Const VK_DECIMAL = &H6E
Private Const VK_DIVIDE = &H6F
Private Const VK_F1 = &H70
Private Const VK_F2 = &H71
Private Const VK_F3 = &H72
Private Const VK_F4 = &H73
Private Const VK_F5 = &H74
Private Const VK_F6 = &H75
Private Const VK_F7 = &H76
Private Const VK_F8 = &H77
```

```
Private Const VK_F9 = &H78
Private Const VK_F10 = &H79
Private Const VK_F11 = &H7A
Private Const VK_F12 = &H7B
Private Const VK_F13 = &H7C
Private Const VK_F14 = &H7D
Private Const VK_F15 = &H7E
Private Const VK_F16 = &H7F
Private Const VK_F17 = &H80
Private Const VK_F18 = &H81
Private Const VK_F19 = &H82
Private Const VK_F20 = &H83
Private Const VK_F21 = &H84
Private Const VK_F22 = &H85
Private Const VK_F23 = &H86
Private Const VK_F24 = &H87
Private Const VK_NUMLOCK = &H90
Private Const VK_OEM_SCROLL = &H91
Private Const VK_OEM_1 = &HBA
Private Const VK_OEM_PLUS = &HBB
Private Const VK_OEM_COMMA = &HBC
Private Const VK_OEM_MINUS = &HBD
Private Const VK_OEM_PERIOD = &HBE
Private Const VK_OEM_2 = &HBF
Private Const VK_OEM_3 = &HC0
Private Const VK_OEM_4 = &HDB
Private Const VK_OEM_5 = &HDC
Private Const VK_OEM_6 = &HDD
Private Const VK_OEM_7 = &HDE
Private Const VK_OEM_8 = &HDF
Private Const VK_ICO_F17 = &HE0
Private Const VK_ICO_F18 = &HE1
Private Const VK_OEM102 = &HE2
Private Const VK_ICO_HELP = &HE3
Private Const VK_ICO_00 = &HE4
Private Const VK_ICO_CLEAR = &HE6
Private Const VK_OEM_RESET = &HE9
Private Const VK_OEM_JUMP = &HEA
Private Const VK_OEM_PA1 = &HEB
Private Const VK_OEM_PA2 = &HEC
Private Const VK_OEM_PA3 = &HED
Private Const VK_OEM_WSCTRL = &HEE
Private Const VK_OEM_CUSEL = &HEF
Private Const VK_OEM_ATTN = &HF0
```

```
Private Const VK_OEM_FINNISH = &HF1
Private Const VK_OEM_COPY = &HF2
Private Const VK_OEM_AUTO = &HF3
Private Const VK_OEM_ENLW = &HF4
Private Const VK_OEM_BACKTAB = &HF5
Private Const VK_ATTN = &HF6
Private Const VK_CRSEL = &HF7
Private Const VK_EXSEL = &HF8
Private Const VK_EREOF = &HF9
Private Const VK_PLAY = &HFA
Private Const VK_ZOOM = &HFB
Private Const VK_NONAME = &HFC
Private Const VK_PA1 = &HFD
Private Const VK_OEM_CLEAR = &HFE
```

'FindAWindow 通过标题找窗体句柄

Rem 通过标题找窗体句柄

```
Public Function FindAWindow(Name As String, Return_Hdc As Long) As Boolean
Return_Hdc = FindWindow(vbNullString, Name)
If Return_Hdc = 0 Then
FindAWindow = False
Else
FindAWindow = True
End If
End Function
```

'FindASubWindow 通过标题和父窗体句柄找下级窗体句柄

Rem 通过标题和父窗体句柄找下级窗体句柄

```
Public Function FindASubWindow(Name As String, FatherHdc As Long, Return_Hdc As Long) As
Boolean
Return_Hdc = FindWindowEx(FatherHdc, 0, vbNullString, Name)
If Return_Hdc = 0 Then
FindASubWindow = False
Else
FindASubWindow = True
End If
End Function
```

'SendAMessageMouseDownAndUp 发送模拟的按下和抬起鼠标左键

Rem 发送模拟的按下和抬起鼠标左键

```
Public Function SendAMessageMouseDownAndUp(hDC As Long, X As Long, Y As Long) As
Boolean
Dim P2 As Long
P2 = Y * 65536 + X
    'SetForegroundWindow Hdc
    SendMessage hDC, WM_LBUTTONDOWN, 0, P2
    Sleep 300
    SendMessage hDC, WM_LBUTTONUP, 0, P2
End Function
```

'SendAMessageMouseMove 发送模拟的鼠标移动

Rem 发送模拟的鼠标移动

```
Public Function SendAMessageMouseMove(hDC As Long, X As Long, Y As Long) As Boolean
Dim P2 As Long
P2 = Y * 65536 + X
    'SetForegroundWindow Hdc
    SendMessage hDC, WM_MOUSEMOVE, 0, P2
    Sleep 300
End Function
```

'PostAMessageMouseDownAndUp 发送模拟的按下和抬起鼠标左键

Rem 发送模拟的按下和抬起鼠标左键

```
Public Function PostAMessageMouseDownAndUp(hDC As Long, X As Long, Y As Long) As Boolean
'SetForegroundWindow Hdc
Dim wParam As Long, lParam As Long
wParam = 1
lParam = Y * 65536 + X
PostMessage hDC, WM_LBUTTONDOWN, wParam, lParam
Sleep 150
PostMessage hDC, WM_LBUTTONUP, wParam, lParam
Sleep 150
End Function
```

'GetHighWord 获取一个 32 位整数的高 16 位（不知何用）

```
Private Function GetHighWord(ByVal TheValue As Long) As Long
    ' 获取一个 32 位整数的高 16 位。
    Dim rtnVal As Long
    CopyMemory rtnVal, ByVal VarPtr(TheValue) + 2, 2&
    GetHighWord = rtnVal
End Function
```

'GetLowWord 获取一个 32 位整数的低 16 位（不知何用）

```
Private Function GetLowWord(ByVal TheValue As Long) As Long
    ' 获取一个 32 位整数的低 16 位。
    Dim rtnVal As Long
    CopyMemory rtnVal, TheValue, 2&
    GetLowWord = rtnVal
End Function
```

```
'X = GetLowWord(lParam)
'Y = GetHighWord(lParam)
'其他解决方案
'End Function
```

'MouseEventMoveAndClick 当前显示器分辨率为基础，移动和点击鼠标

Rem MouseEvent

Rem 当前显示器分辨率为基础，移动和点击鼠标

```
Public Function MouseEventMoveAndClick(X As Long, Y As Long)
Dim mw As Long
Dim mh As Long
mw = X / SW * 65535
mh = Y / SH * 65535
mouse_event MOUSEEVENTF_ABSOLUTE + MOUSEEVENTF_MOVE, mw, mh, 0, 0
mouse_event MOUSEEVENTF_LEFTDOWN Or MOUSEEVENTF_LEFTUP, 0, 0, 0, 0
End Function
```

'SendAMessageMouseWheel 鼠标滚轮

Rem 负值向下滚动

```
Public Function SendAMessageMouseWheel(hDC As Long, X As Long, Y As Long, RotateH As Long,
RotateL As Long) As Boolean
Dim P3 As Long
Dim P2 As Long
P3 = RotateH * 65536 + RotateL
P2 = Y * 65536 + X
    SendMessage hDC, WM_MOUSEWheel, P3, P2
    Sleep 300
End Function
```

'SendAMessageMouseWheelScRollDown 鼠标滚轮向下

Rem 负值向下滚动
Rem 这个 **EVE** 不能识别因为子窗体没找到句柄

```
Public Function SendAMessageMouseWheelScRollDown(hDC As Long, X As Long, Y As Long,
Rotate As Long) As Boolean
Dim P2 As Long
P2 = Y * 65536 + X
    SendMessage hDC, WM_VSCROLL, SB_LINEDOWN, 0
    Sleep 300
End Function
```

'SendAMessageKeyPress 模拟按键（未使用）

Rem 这个没使用
Rem 这个模拟按键要用 **PostMessage**

```
Public Function SendAMessageKeyPress(hDC As Long, KeyCode As Integer) As Boolean
    'SendMessage Hdc, WM_SETFOCUS, 0, ByVal 0&
    SendMessage hDC, WM_KEYDOWN, &H41, 0
    Sleep 150
    SendMessage hDC, WM_KEYUP, &H41, 0
    Sleep 150
End Function
```

'PostAMessageKeyPress 模拟按键

Rem 这个模拟按键
Rem WM_KEYUP 会让按 **2** 次
Rem EVE 游戏需要获得焦点 用 **SetForegroundWindow Hdc**

```
Public Function PostAMessageKeyPress(hDC As Long, KeyCode As Integer) As Boolean
    'SendMessage Hdc, WM_SETFOCUS, 0, ByVal 0&
```

```
    'PostMessage Hdc, WM_SETFOCUS, 0, ByVal 0&
    SetForegroundWindow hDC
    PostMessage hDC, WM_KEYDOWN, KeyCode, 0&
    Sleep 150
    'PostMessage Hdc, WM_KEYUP, KeyCode, 0&

End Function
```

'ShowWindow 突前显示当前窗体

Rem 突前显示当前窗体

```
Public Function ShowWindow(hDC As Long) As Boolean
SetForegroundWindow hDC
End Function
```

30. Statistics

详细说明：

函数或方法名称	作用
NumInt	Gamma 子程序
Group	按样本数分组
Group2	按样本数分组
Gamma	Gamma 函数
SampleStandardError	样本标准误
StandardError	标准误
ArithmeticMeanMethodOfWeighting	算术平均数加权法
Range	ClassInterval 子程序
ClassInterval	组距
ClassLowerLimit	组下限
FirstClassLowerLimit	第一组组下限
FirstClassMidValue	组中值
AverageNumber	平均值返回离差和平方和
Median	中位数
MedianMethodOfTable	已分组资料中位数的计算方法
GeometricMean	几何平均数
Mode	众数
HarmonicAverage	调和平均数
MeanSquare	样本方差
MeanSquareMethodOfWeighting	样本方差加权法
MeanSquareT	MeanSquareT 总体方差
fxSum	标准差
CoefficientOfVariation	变异系数
NormalDistributionPosibility	正态分布概率
NormalDistribution_μα	正态分布μα
Max	最大值
Mix	最小值
MeasurementData	计量资料的整理
CountData	计数资料
QualitativeCharacteristicsData	质量性状资料、半定量（等级）资料的整理
HarmonicMean	调和平均数
HarmonicAverage	调和平均数
BernoulliTrials	二项式
PoissonDistributionMean	泊松分布平均值
PoissonDistributionMeanSquare	泊松分布均方

PoissonDistributionTable	泊松分布表
PoissonDistribution	泊松分布
PoissonDistributionPosibility	泊松分布概率值
BernoulliTrialsPosibility	二项式分布概率
Factorial	阶乘
μ	服从二项分布 B(n,p)的随机变量之平均数μ
Squareσ	总体方差
σ	二项分布的标准差σ
σp	总体百分数标准误
SP	常以样本百分数来估计
TestOfSignificance_t	T 显著检验
nolinearR	非线性拟合度 R 值
LinearRegressionAnalysis_SPARE	线性回归程序（备用）
LinearRegressionAnalysis	线性回归程序
LinearRegressionAnalysis_Equation	这个和 LinearRegressionAnalysis 运行结果一样，多返回 2 个系数
TwoElementLinearAnalysis_Equation	二元线性分析函数式
OneVeriblePolyNomialRegression	多元线性回归分析函数式（只能分析 1 元 1 次到 7 次）
MultiElementLinearAnalysis_Equation2	多元线性分析函数式 2 号
MultiElementLinearAnalysis_Equation3	多元线性分析并返回多种值 3 号
TheBestMultipleLinearRegressionEquation	最好的多元线性回归
QuadraticPolyNomialWithOneVariableRegressionAnalysis	废弃程序
MultiElementLinearAnalysis_Equation4	多元线性回归分析 4 号
MultiElementLinearAnalysis_Equation5	多元线性回归分析 5 号
TheBestMultipleLinearRegressionEquation_R	最好的多元线性回归返回 r 值
PathAnalysis_Test1	通径分析测试 1
PathAnalysis_Test2	通径分析测试 2

代码：

```
Option Explicit
Option Base 1
'*************************************************************************************
'名称 = Statistics001.Cls
'作者 = 张宏
'最后修改时间 = 2022 年 1 月 2 日
'这个模块是数学统计模块
'多元线性分析                                    16-8-29
'多元线性分析完成                                16-9-6
'思考问题，F 值会不会为负？                      16-9-7 暂时看 F 值公式不会为负
'N 阶行列式为 0 的时候，矩阵不可逆。             2016-9-10
```

'也就是说有的数据会导致 N 阶行列式为 0，除数为 0 而程序出错:
'通径分析通径系数显著性检验出现问题，还没有更改 SSR 计算方法不清楚，没找到解决办法 16-9-25
'增加 Times 2019-4-30
'增加 Group 2019-4-30
'增加正太分布概率计算的说明 2019-5-7
'增加样本方差 S^2 加权法 2019-5-30
'增加总体方差 Squareσ 2019-5-30
'增加 NormalDistributionPosibilityRectangle 正太分布新矩形法 2019-6-2
'增加以前版本漏掉内容 2022-1-2
'**********

Rem 算数平均值

```
Const conPi = 3.14159265358979
Const conE = 2.71828182845905
Private PiUserDefined As Double
Private eUserDefined As Double
Private mvarPi As Double
Private mvarE As Double
```

Rem Gamma 子程序
Rem 判断整数

```
Public Function NumInt(ByVal NumZ As Double) As Boolean '为 1 时为整数，2 是为
If Int(NumZ) = NumZ Then
NumInt = True
Else: NumInt = False
'MsgBox "不是整数"
End If
End Function
```

'Group 按样本数分组

Rem 资料分组

```
Public Function Group(ByVal Samples As Long) As String '
If Samples >= 10 And Samples <= 100 Then
Group = "7~10"
Else
Group = "None"
End If
If Samples >= 100 And Samples <= 200 Then
Group = "9~12"
End If
If Samples >= 200 And Samples <= 500 Then
Group = "12~17"
End If
```

```
If Samples > 500 Then
Group = "17~30"
End If
If Samples = 100 Then
Group = "9~10"
End If
If Samples = 200 Then
Group = "12"
End If
End Function
```

'Group2 按样本数分组

Rem 资料分组

```
Public Function Group2(ByVal Samples As Long) As String '
If Samples >= 10 And Samples <= 100 Then
Group = "7~10"
Else
Group = "None"
End If
If Samples >= 100 And Samples <= 200 Then
Group = "9~12"
End If
If Samples >= 200 And Samples <= 500 Then
Group = "12~17"
End If
If Samples > 500 Then
Group = "17~30"
End If
If Samples = 100 Then
Group = "9~10"
End If
If Samples = 200 Then
Group = "12"
End If
End Function
```

'Gamma 函数

```
Public Function Gamma(ByVal Numg As Double) As Double '和 NumInt 一起使用。
If NumInt(Numg * 2) = False Then
MsgBox "错误参数，请输入 0.5 的倍数"
```

```
Gamma = 0
Exit Function
End If
Gamma = 1
If NumInt(Numg) = True Then '20 内的数字通过测试 19! = 1.21645100408832E+17
Dim i As Double
For i = 1 To Numg - 1
Gamma = Gamma * i
'Debug.Print Gamma
Next
End If
If NumInt(Numg) = False Then '4.5 内的数字通过测试
For i = 0.5 To Numg - 1
Gamma = Gamma * i
'Debug.Print Gamma
Next
Gamma = Gamma * Sqr(conPi)
End If
End Function
```

'SampleStandardError 样本标准误

Rem 样本标准误(和标准误比较只是样本方差还是总体方差)

```
Public Function SampleStandardError(ByRef Data() As Double, ByVal n As Long) As Double
Dim SOS As Double
Dim PSD As Double
Dim Part1 As Double
Dim Part2 As Double
Part1 = Sqr(MeanSquare(Data(), SOS, PSD))
Part2 = n
StandardError = Part1 / Sqr(Part2)
End Function
```

'StandardError 标准误

Rem 标准误

Rem 说明各样本平均数 间差异程度大，样本平均数的精确性低

```
Public Function StandardError(ByRef Data() As Double, ByVal n As Long) As Double
Dim SOS As Double
Dim PSD As Double
Dim Part1 As Double
Dim Part2 As Double
```

```
Part1 = Sqr(MeanSquareT(Data(), SOS, PSD))
Part2 = n
StandardError = Part1 / Sqr(Part2)
End Function
```

```
Public Property Get Pi() As Double '读属性
Pi = mvarPi
End Property
```

```
Public Property Let Pi(ByVal PiUserDefined As Double)
mvarPi = PiUserDefined
End Property
```

```
Public Property Get E() As Double '读属性
E = mvarE
End Property
Public Property Let E(ByVal eUserDefined As Double)
mvarE = eUserDefined
End Property
```

'ArithmeticMean 算术平均数

Rem 算术平均数（arithmetic mean）
Rem 算术平均数是指资料中各观测值的总和除以观测值个数所得的商，简称平均数或均数，记为
Rem 算术平均数可根据样本大小及分组情况而采用直接法或加权法计算。

```
Public Function ArithmeticMean(ByRef Data() As Double, ByRef Return_Sum As Double) As
Double   'text 改个 MULTIPLINE 就可以了
Dim Sum As Double, l As Long, i As Long
For i = LBound(Data) To UBound(Data)
Sum = Sum + Data(i)
l = l + 1
Next i
ArithmeticMean = Sum / l
Return_Sum = Sum
End Function
```

'ArithmeticMeanMethodOfWeighting 算术平均数加权法

Rem 算术平均数（arithmetic mean）
Rem 加权法

```
Public Function ArithmeticMeanMethodOfWeighting(ByRef ClassMidValue() As Double, ByRef
```

```
NumberOfTimes() As Long, ByVal Clusters As Long) As Double 'text 改个 MULTIPLINE 就可以了
Dim i As Long
Dim Sum1 As Double, Sum2 As Double
Dim Sum3 As Long
If ((UBound(ClassMidValue) - LBound(ClassMidValue)) = (UBound(NumberOfTimes) -
LBound(NumberOfTimes))) Then
For i = LBound(ClassMidValue) To UBound(ClassMidValue)
Sum1 = Sum1 + ClassMidValue(i) * NumberOfTimes(i)
Sum2 = Sum2 + NumberOfTimes(i)
Sum3 = Sum3 + 1
Next
ArithmeticMeanMethodOfWeighting = Sum1 / Sum2
If (Clusters <> 0) Then
If (Sum3 <> Clusters) Then
MsgBox "数据有问题"
End If
End If
Else
MsgBox "数据有问题"
End If
End Function
```

Rem ClassInterval 子程序

```
Public Function Range(ByRef Data() As Double) As Double
Range = Max(Data()) - Mix(Data())
End Function
```

'ClassInterval 组距

Rem 组距（i）
Rem 而且第一组的下限应低于最小变量值，最后一组的上限应高于最大变量值
Rem 选用的 4 舍 5 入，不知道对不

```
'Public Function ClassInterval(ByVal Range As Double, ByVal ClassNumber As Long) As Long
'ClassInterval = Math.Round(Range / ClassNumber)
'End Function
```

```
Public Function ClassInterval(ByRef Data() As Double, ByVal ClassNumber As Long) As Long
ClassInterval = Math.Round(Range(Data()) / ClassNumber)
Dim i As Long
i = 1
End Function
```

'ClassLowerLimit 组下限

Rem 组下限

```
Public Function ClassLowerLimit(ByRef Data() As Double, ByVal ClassNumber As Long, ByRef
Return_ClassLowerLimit() As Double)
Dim i As Long
ReDim Return_ClassLowerLimit(ClassNumber + 1)
For i = 0 To (ClassNumber)
Return_ClassLowerLimit(i + 1) = i * ClassInterval(Data(), ClassNumber) +
FirstClassLowerLimit(Data(), ClassNumber)
Next i
End Function
```

'FirstClassLowerLimit 第一组组下限

Rem 第一组组下限

```
Public Function FirstClassLowerLimit(ByRef Data() As Double, ByVal ClassNumber As Long) As
Double
FirstClassLowerLimit = FirstClassMidValue(Data()) - ClassInterval(Data(), ClassNumber) / 2
End Function
```

'FirstClassMidValue 组中值

Rem 组中值

Rem 一般第一组的组中值以接近于或等于资料中的最小值为好

```
Public Function FirstClassMidValue(ByRef Data() As Double) As Double
FirstClassMidValue = Mix(Data())
End Function
```

'AverageNumber 平均值返回离差和平方和

```
Public Function AverageNumber(ByRef Data() As Double, ByRef Return_DeviationFromAverage As
Double, ByRef Return_SumOfSquares As Double) As Double 'sum of squares
Dim i As Long
Dim Sum1 As Double, Sum2 As Long
For i = LBound(Data()) To UBound(Data())
Sum1 = Sum1 + Data(i)
Sum2 = Sum2 + 1
Next
AverageNumber = Sum1 / Sum2
```

```
For i = LBound(Data()) To UBound(Data())
Return_DeviationFromAverage = Return_DeviationFromAverage + Data(i) - AverageNumber
Return_SumOfSquares = Return_SumOfSquares + (Data(i) - AverageNumber) ^ 2
Next
End Function
```

'Median 中位数

Rem 中位数（**median**）
Rem 将资料内所有观测值从小到大依次排列，位于中间的那个观测值，称为中位数，记为**Md**。
Rem 当观测值的个数是偶数时，则以中间两个观测值的平均数作为中位数。中位数简称中数
Rem 当所获得的数据资料呈偏态分布时，中位数的代表性优于算术平均数。
Rem 中位数的计算方法因资料是否分组而有所不同。

```
Public Function Median(ByRef Data() As Double, ByVal ReturnTotalNumbersOdd As Boolean) As
Double 'Odd 奇数
Dim i As Long, n As Long, m As Long
n = 0
For i = LBound(Data) To UBound(Data)
'Debug.Print "data(" & i & ")=" & Data(i)
n = n + 1
Next
m = 1 - LBound(Data)
If (n Mod 2 = 1) Then
Median = Data(((n + 1) / 2) - m)
ReturnTotalNumbersOdd = True
End If
If (n Mod 2 = 0) Then
Median = (Data((n / 2) - m) + Data((n / 2 + 1) - m)) / 2
ReturnTotalNumbersOdd = False
End If
End Function
```

'MedianMethodOfTable 已分组资料中位数的计算方法

Rem 已分组资料中位数的计算方法
Rem 编制成次数分布表，则可利用次数分布表来计算中位数
'L-中位数所在组的下限；
'i-组距；
'f-中位数所在组的次数；
'n-总次数；

'c -小于中数所在组的累加次数

```
Public Function MedianMethodOfTable(ByRef LBoundOfTheGroupWhichMedianIn As Double, _
ByVal ClassInterval As Double, ByVal TimesOfTheGroupWhichMedianIn As Double, _
ByVal TotalTimes As Double, ByVal TimesOfTheGroupsWhichSmallerThanMedianIn) As Double
MedianMethodOfTable = LBoundOfTheGroupWhichMedianIn + ClassInterval / _
TimesOfTheGroupWhichMedianIn * (TotalTimes / 2 -
TimesOfTheGroupsWhichSmallerThanMedianIn)
End Function
```

'GeometricMean 几何平均数

```
Rem 几何平均数 geometric mean
Rem 这个更能够代表平均值
Rem 如畜禽、水产养殖的增长率，抗体的滴度，药物的效价，畜禽疾病的潜伏期等
Rem 用几何平均数比用算术平均数更能代表其平均水平。
```

```
Public Function GeometricMean(ByRef Data() As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
n = 0
For i = LBound(Data()) To UBound(Data())
Sum1 = Sum1 + Log(Data(i))
n = n + 1
Next i
GeometricMean = Exp(Sum1 / n)
End Function
```

'Mode 众数

```
Rem 众数（mode）
Rem 资料中出现次数最多的那个观测值或次数最多一组的组中值，称为众数，记为 M0。
Rem 次数最多，其组中值。
Rem 非组中值众数
```

```
Public Function Mode(ByRef Data() As Double, ByRef Return_Mode() As Double) As Double
Dim i As Long, l As Long, n As Long, m As Long

Dim CompareData() As Double
For i = LBound(Data) To UBound(Data)
m = m + 1
ReDim Preserve CompareData(m)
n = 0
For l = (i + 1) To UBound(Data)
'Debug.Print "data(" & i & ")=" & Data(i) & "    data(" & L & ")=" & Data(L)
```

```
If (Data(i) = Data(l)) Then
n = n + 1
End If
Next l
CompareData(m) = n
'Debug.Print "CompareData(" + CStr(m) + ")=" + CStr(CompareData(m))
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim O As Long, P As Long
For i = 1 To UBound(CompareData)
If (CompareData(i) = Max(CompareData())) Then
O = O + 1
'~~~~~~~~~~~~~~~~~~~~~~~~
P = i - (1 - LBound(Data()))                    '可能有问题
'~~~~~~~~~~~~~~~~~~~~~~~~
Mode = Data(P)
ReDim Preserve Return_Mode(O)
Return_Mode(O) = Data(P)
End If
Next i
If (O > 1) Then
MsgBox "有 2 个或以上值，数据在 Return_Mode()"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'MsgBox "非组中值众数"
End Function
```

'HarmonicAverage 调和平均数

```
'Public Function HarmonicAverage(ByRef Data() As Double) As Double
'Dim i As Long, n As Long, Sum1 As Double
'For i = LBound(Data()) To UBound(Data())
'Sum1 = Sum1 + (1 / Data(i))
'n = n + 1
'Next i
'HarmonicAverage = 1 / (Sum1 * (1 / n))
'End Function
```

'MeanSquare 样本方差

Rem 样本方差 S^2

```
Public Function MeanSquare(ByRef Data() As Double, ByRef Return_SumOfSquares As Double, _
```

```
ByRef Return_SampleStandardDeviation As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
Dim Average As Double, SumOfSquares As Double
'Dim Sum2 As Double
For i = LBound(Data()) To UBound(Data())
Sum1 = Sum1 + Data(i)
'Sum2 = Sum2 + Data(i) ^ 2
n = n + 1
Next i
'Average = Sum2 / n
Average = Sum1 / n
For i = LBound(Data()) To UBound(Data())
Return_SumOfSquares = Return_SumOfSquares + (Data(i) - Average) ^ 2
Next i
MeanSquare = Return_SumOfSquares / (n - 1)
Return_SampleStandardDeviation = Sqr(MeanSquare)
End Function
```

'MeanSquareMethodOfWeighting 样本方差加权法

Rem 样本方差 S^2 加权法
Rem NumberOfTimes()分组的次数
Rem ClassMidValue() 分组的组中值
Rem SampleNumber 样本数

```
Public Function MeanSquareMethodOfWeighting(ByRef NumberOfTimes() As Long, _
ByRef ClassMidValue() As Double, ByVal SampleNumber As Long) As Double
Dim i As Long, l As Long, Part1 As Double, Part2 As Double, n As Long
Dim Part3 As Double
n = SampleNumber
For i = LBound(NumberOfTimes) To UBound(NumberOfTimes)
Part1 = Part1 + NumberOfTimes(i) * ClassMidValue(i) ^ 2
Part2 = Part2 + NumberOfTimes(i) * ClassMidValue(i)
Next i
Part3 = Part1 - Part2 ^ 2 / n
MeanSquareMethodOfWeighting = Sqr(Part3 / (n - 1))
Rem 自检程序
Part1 = UBound(NumberOfTimes) - LBound(NumberOfTimes)
Part2 = UBound(ClassMidValue) - LBound(ClassMidValue)
If Part1 <> Part2 Then
MsgBox "次数个数不等于组中值个数"
End If
End Function
```

'MeanSquareT 总体方差

Rem 总体方差σ^2

```
Public Function MeanSquareT(ByRef Data() As Double, ByRef Return_SumOfSquares As Double,
ByRef Return_PopulationStandardDeviation As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
Dim Average As Double, SumOfSquares As Double
For i = LBound(Data()) To UBound(Data())
Sum1 = Sum1 + Data(i)
n = n + 1
Next i
Average = Sum1 / n
For i = LBound(Data()) To UBound(Data())
Return_SumOfSquares = Return_SumOfSquares + (Data(i) - Average) ^ 2
Next i
MeanSquareT = Return_SumOfSquares / n
Return_PopulationStandardDeviation = Sqr(MeanSquareT)
End Function
```

'fxSum 标准差

Rem 标准差

```
Public Function fxSum(ByRef ClassMiddleValue() As Double, ByRef Time() As Long, _
ByRef Return_fx() As Double, ByRef Return_fxSquare() As Double, ByRef Return_fxSquareSum As
Double) As Double
Dim i As Long, n As Long
For i = LBound(ClassMiddleValue) To UBound(ClassMiddleValue)
n = n + 1
ReDim Preserve Return_fx(n)
ReDim Preserve Return_fxSquare(n)
Return_fx(n) = ClassMiddleValue(i) * Time(i)
Return_fxSquare(n) = ClassMiddleValue(i) ^ 2 * Time(i)
fxSum = fxSum + ClassMiddleValue(i) * Time(i)
Return_fxSquareSum = Return_fxSquareSum + ClassMiddleValue(i) ^ 2 * Time(i)
Next
End Function
```

'CoefficientOfVariation 变异系数

Rem 变异系数
Rem SampleStandardDeviation 样本标准差

Rem AverageNumber 样本平均数

Rem DecimalDigits 小数位数

```
Public Function CoefficientOfVariation(ByVal SampleStandardDeviation As Double, _
ByVal AverageNumber As Double, ByVal DecimalDigits As Long) As String
CoefficientOfVariation = SampleStandardDeviation / AverageNumber
If DecimalDigits = 0 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0%")
ElseIf DecimalDigits = 1 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0.0%")
ElseIf DecimalDigits = 2 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0.00%")
ElseIf DecimalDigits = 3 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0.000%")
ElseIf DecimalDigits = 4 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0.0000%")
Else
CoefficientOfVariation = SampleStandardDeviation / AverageNumber
End If
End Function
```

'NormalDistributionPosibility 正态分布概率

Rem 正态分布概率

Rem μ是位置参数

Rem σ是函数胖瘦参数

Rem 函数会出现问题，范围值越小，分割块数越多越准确。所以还是需要正态分布表加以校对

Rem 例子：标准正太分布，μ=0 σ^2=1 PrecisionPieces=1000 μ1=-5 μ2=-2.44 需要给定值 NormalDistributionPosibility=0.0073434

```
Public Function NormalDistributionPosibility(ByVal μ As Double, ByVal σ As Double, _
ByVal PrecisionPieces As Double, ByVal μ1 As Double, ByVal μ2 As Double, _
ByRef Return_RecordForShape() As Double, ByVal CalculateType As Long) As Double
Dim X As Double, i As Long, Distance As Double
ReDim Return_RecordForShape(1 To PrecisionPieces)
Dim NDP As Double, μD As Double
If (CalculateType <> 1 And CalculateType <> 2 And CalculateType <> 3) Then
CalculateType = 3
'MsgBox "请输入数字 1 代表矩形法,2 代表梯形法,3 代表抛物线法 。 现在执行默认 3：抛物线法"
End If
'If (σ <> 1 Or μ <> 0) Then
'μ1 = (μ1 - μ) / σ
'μ2 = (μ2 - μ) / σ
```

```
'μ = 0
'σ = 1
'End If
Rem 返回各分割块的最左的边的 Y 值
Dim Part1 As Double
μD = μ2 - μ1
Distance = μD / PrecisionPieces
For i = 0 To PrecisionPieces - 1
X = μ1 + Distance * i
Part1 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X - μ) ^ 2) / (2 * σ ^ 2))
Return_RecordForShape(i + 1) = Part1
Next
Rem 矩形法
'矩形法用于单调函数误差很大
If CalculateType = 1 Then
For i = 0 To PrecisionPieces - 1
X = μ1 + Distance * i
Part1 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X - μ) ^ 2) / (2 * σ ^ 2))
'Part1 = x '实验程序是否正确的代码
NormalDistributionPosibility = NormalDistributionPosibility + Part1 * Distance
'Debug.Print Part1 * Distance
'Debug.Print x & "=x " & NormalDistributionPosibility & "=NDP" & "              1"
Next i
End If

Rem 梯形法
Dim Part2 As Double, X1 As Double
If CalculateType = 2 Then
For i = 0 To PrecisionPieces - 1
X = μ1 + Distance * i
X1 = μ1 + Distance * (i + 1)
Part1 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X - μ) ^ 2) / (2 * σ ^ 2))
Part2 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X1 - μ) ^ 2) / (2 * σ ^ 2))
'Part1 = x: Part2 = x1 '实验程序是否正确的代码
NormalDistributionPosibility = NormalDistributionPosibility + ((Part1 + Part2) * Distance) / 2
'Debug.Print X & "=x " & NormalDistributionPosibility & "=NDP" & "              2"
Next i
End If

Rem 抛物线法
Dim Part3 As Double, X2 As Double
If CalculateType = 3 Then
For i = 0 To PrecisionPieces - 1 Step 2
X = μ1 + Distance * i
```

```
X1 = μ1 + Distance * (i + 1)
X2 = μ1 + Distance * (i + 2)
Part1 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X - μ) ^ 2) / (2 * σ ^ 2))
Part2 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X1 - μ) ^ 2) / (2 * σ ^ 2))
Part3 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X2 - μ) ^ 2) / (2 * σ ^ 2))
'Part1 = x: Part2 = x1: Part3 = x2 '实验程序是否正确的代码
If (PrecisionPieces Mod 2) = 0 Then
        NormalDistributionPosibility = NormalDistributionPosibility + 1 / 3 * (Part1 + 4 * Part2 +
Part3) * Distance
        'Debug.Print x & "=x " & NormalDistributionPosibility & "=NDP" & "              3"
        'Debug.Print 1 / 3 * (Part1 + 4 * Part2 + Part3) * Distance
End If

If (PrecisionPieces Mod 2 <> 0) Then '奇数
        If (i <> PrecisionPieces - 1) Then
        NormalDistributionPosibility = NormalDistributionPosibility + 1 / 3 * (Part1 + 4 * Part2 +
Part3) * Distance
        'Debug.Print x & "=x " & NormalDistributionPosibility & "=NDP" & "              3"
        End If
        If (i = PrecisionPieces - 1) Then
        NormalDistributionPosibility = NormalDistributionPosibility + ((Part1 + Part2) * Distance)
/ 2
        'Debug.Print x & "=x " & NormalDistributionPosibility & "=NDP" & "              3"
        End If
End If
Next i
End If
DoEvents

End Function
```

'NormalDistribution_μα正态分布μα

```
Public Function NormalDistribution_μα(ByVal μ As Double, ByVal σ As Double, _
ByVal α As Double, ByVal PrecisionPieces As Double, ByVal WidthOfFunction As Double, _
ByRef Return_μ1 As Double, ByRef Return_μ2 As Double, ByRef Return_RecordForShape() As
Double, _
ByVal CalculateType As Long) As Double
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim X As Double, i As Long, Distance As Double, Y As Double
Dim μ2 As Double, P As Double, μ1 As Double
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  防止输入错误
```

```
If (CalculateType <> 1 And CalculateType <> 2 And CalculateType <> 3) Then
CalculateType = 3
'MsgBox "请输入数字 1 代表矩形法,2 代表梯形法,3 代表抛物线法 。 现在执行默认 3：抛物线法"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 数据限定
If (α > 1 Or α < 0) Then
MsgBox "α数值不合理"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 数据限定
If WidthOfFunction < 8 Then
MsgBox " WidthOfFunction 数值偏小"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 数据限定
If PrecisionPieces < 10 Then
MsgBox "PrecisionPieces 分块数太少"
ElseIf PrecisionPieces > 10000 Then
MsgBox "PrecisionPieces 分块数过多，等待时间较长"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 加速程序 2
Dim SpeedUp As Double
If PrecisionPieces >= 10 Then
Select Case α
'Case Is >= 0.1
Case 0.1 To 0.11
μ2 = WidthOfFunction / 2 - 1.7
Case 0.5 To 1
μ2 = 1
End Select
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
If α <= 0.5 Then
P = 0
'If (σ <> 1 Or μ <> 0) Then
μ1 = -WidthOfFunction / 2
'WidthOfFunction = -(-WidthOfFunction / 2 - μ) / σ * 2
'End If
Do While P < α / 2 '应该是小于，不应该是小于等于
μ2 = μ2 + 1 / PrecisionPieces
P = NormalDistributionPosibility(0, 1, PrecisionPieces, μ1, μ2 + μ1, Return_RecordForShape(),
```

```
CalculateType)
'i = i + 1
'Debug.Print "μ1= " & μ1 & "        μ2 + μ1=" & (μ2 + μ1) & "      P=" & P & "        Times=" & i
Loop
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
If α > 0.5 And α <= 1 Then
P = 1
'If (σ <> 1 Or μ <> 0) Then
μ1 = -WidthOfFunction / 2
'WidthOfFunction = (WidthOfFunction / 2 - μ) / σ * 2
'End If
Do While P >= α / 2
μ2 = μ2 + 1 / PrecisionPieces
Rem 下面这个问题还没解决
P = 1 - NormalDistributionPosibility(0, 1, PrecisionPieces, μ1, μ2 + μ1, Return_RecordForShape(),
CalculateType)
'Debug.Print "μ1= " & μ1 & "        μ2 + μ1=" & (μ2 + μ1) & "      P=" & p
Loop
End If
Rem 结果
NormalDistribution_μα = -μ1 - μ2
Return_μ1 = (-NormalDistribution_μα) * σ + μ
Return_μ2 = (NormalDistribution_μα) * σ + μ
End Function
```

'Max 最大值

```
Public Function Max(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next

For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) > cData(i + 1) Then
Max = cData(i)
cData(i + 1) = cData(i)
Else
Max = cData(i + 1)
```

```
End If
End If
Next
End Function
```

'Mix 最小值

```
Public Function Mix(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(Data()) To UBound(Data())
cData(i) = Data(i)
Next
For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) < cData(i + 1) Then
Mix = cData(i)
cData(i + 1) = cData(i)
Else
Mix = cData(i + 1)
End If
End If
Next
End Function
```

'MeasurementData 计量资料的整理

Rem 计量资料的整理
Rem ClassNumber=分组数
Rem DecimalDigits=精确度=精确度 3=0.001
Rem ClassMiddleValueModification=中位数修正值
Rem Return_ClassInterval=返回组距
Rem Return_ClassMiddleValue()=返回组中值
Rem Return_ClassLowerLimit()=返回组下限
Rem Return_ClassUpperLimit()=返回组上限
Rem Return_Time()=返回次数
Rem Return_Mode=返回众数

```
Public Function MeasurementData(ByRef Data() As Double, ByVal ClassNumber As Long, ByVal
DecimalDigits As Long, _
ByVal ClassMiddleValueModification As Double, ByRef Return_ClassInterval As Double, _
ByRef Return_ClassMiddleValue() As Double, _
```

```
ByRef Return_ClassLowerLimit() As Double, ByRef Return_ClassUpperLimit() As Double, _
ByRef Return_Time() As Long, ByRef Return_Mode As Double, ByRef Return_ReportTXT As String)
As Boolean
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim FirstClassMidValue As Double
Dim FirstClassLowerLimit    As Double
Dim i As Long
Dim l As Double, P As Double, q As Double
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   Rem  分组数小于 30 的时候不用分组提示
   Dim Array1D_Num As Long
   For i = LBound(Data) To UBound(Data)
   Array1D_Num = Array1D_Num + 1
   Next
   If Array1D_Num < 30 Then
   Return_ReportTXT = Return_ReportTXT + "总组数=" + CStr(Array1D_Num) + "<30 " + "【不用分
组】" + Chr$(13) + Chr$(10)
   End If
a1:
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem ClassInterval  组距的计算
l = 10 ^ -(DecimalDigits)
P = Math.Round(Range(Data()) / ClassNumber, DecimalDigits)
q = Range(Data()) / ClassNumber
If (q > P) Then
Return_ClassInterval = P + l '增加  L  防止 4 舍的时候让值变小
Else
Return_ClassInterval = P '5 入的时候，或相等的时候保持值不变
End If
'Return_ClassInterval = Math.Round(Range(Data()) / ClassNumber, DecimalDigits) + 10 ^
-(DecimalDigits) '增加  10 ^ -(DecimalDigits)防止 4 舍 5 入让值变小
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


FirstClassMidValue = Mix(Data()) + ClassMiddleValueModification
FirstClassLowerLimit = FirstClassMidValue - Return_ClassInterval / 2



ReDim Return_ClassMiddleValue(ClassNumber)
For i = 0 To (ClassNumber - 1)
Return_ClassMiddleValue(i + 1) = i * Return_ClassInterval + FirstClassMidValue
Next i

ReDim Return_ClassLowerLimit(ClassNumber)
For i = 0 To (ClassNumber - 1)
```

```
Return_ClassLowerLimit(i + 1) = i * Return_ClassInterval + FirstClassLowerLimit
Next i
'For i = 1 To ClassNumber
'Debug.Print Return_ClassLowerLimit(i)
'Next
ReDim Return_ClassUpperLimit(ClassNumber)
For i = 1 To ClassNumber
Return_ClassUpperLimit(i) = i * Return_ClassInterval + FirstClassLowerLimit
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 自动修正
Dim n As Long
If Return_ClassUpperLimit(ClassNumber) >= Max(Data()) Then
        If Return_ClassLowerLimit(1) <= Mix(Data()) Then
          Return_ReportTXT = Return_ReportTXT + "第一组下限" + CStr(Return_ClassLowerLimit(1))
+ "<=最小值" + CStr(Mix(Data())) + "【正常】" + Chr$(13) + Chr$(10)
        Else
        'MsgBox "最小值小于第 1 组下限.自动修正失败"
        Return_ReportTXT = Return_ReportTXT + "第一组下限" + CStr(Return_ClassLowerLimit(1))
+ ">最小值" + CStr(Mix(Data())) + "【不正常】" + Chr$(13) + Chr$(10)
        End If
Else
'MsgBox "不符合要求最大值超过末组上限.进行参数自动修正"
Return_ReportTXT = Return_ReportTXT + "最大值>末组上限【不正常】.进行参数自动修正" +
Chr$(13) + Chr$(10)
ClassMiddleValueModification = Max(Data()) - Return_ClassUpperLimit(ClassNumber) '让上限包
括住最大值
Return_ReportTXT = Return_ReportTXT + "中位数修正值(不含中位数)=" +
CStr(ClassMiddleValueModification) + Chr$(13) + Chr$(10)
'MsgBox "参数修正值=" & ClassMiddleValueModification
        If Return_ClassLowerLimit(1) <= Mix(Data()) Then '最小值也在下限内
        GoTo a1
        Else
        Return_ReportTXT = Return_ReportTXT + "第一组下限" + CStr(Return_ClassLowerLimit(1))
+ ">最小值" + CStr(Mix(Data())) + "【不正常】" + Chr$(13) + Chr$(10)
        'Return_ReportTXT    = "False  最小值小于第 1 组下限.自动修正失败"
        End If

End If

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Rem 次数计算
   Dim m As Long
   ReDim Return_Time(1 To ClassNumber)
```

```
For i = LBound(Data()) To UBound(Data())
      For m = 1 To ClassNumber

           If (Data(i) >= Return_ClassLowerLimit(m) And Data(i) < Return_ClassUpperLimit(m))
Then
           Return_Time(m) = Return_Time(m) + 1
           m = ClassNumber   ' 找到后跳出循环，增加速度
           End If
           If m = ClassNumber Then
               If (Data(i) = Return_ClassUpperLimit(m)) Then '最大数 = 上限的情况
               Return_Time(m) = Return_Time(m) + 1
               m = ClassNumber   ' 找到后跳出循环，增加速度
               End If
           End If
           'Debug.Print "Data(" & i & ")= " & Data(i) & "Return_ClassLowerLimit(" & m & ")=" &
Return_ClassLowerLimit(m) & "Return_ClassUpperLimit(" & m & ")=" &
Return_ClassUpperLimit(m)
           If (Data(i) = Return_ClassLowerLimit(m) And Data(i) = Return_ClassUpperLimit(m)) Then
           Return_Time(m) = Return_Time(m) + 1
           m = ClassNumber ' 找到后跳出循环，增加速度
           End If
      Next m
Next i

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 判断总次数不等于总数据数
Dim R As Long, O As Long
For i = 1 To ClassNumber
R = R + Return_Time(i)
Next
O = (UBound(Data) - LBound(Data)) + 1
If (R <> O) Then
'MsgBox "异常"
Return_ReportTXT = Return_ReportTXT + "总次数" + CStr(R) + "<>" + "总数据数" + CStr(O) + "
【不正常】" + Chr$(13) + Chr$(10)
Else
Return_ReportTXT = Return_ReportTXT + "总次数" + CStr(R) + "=" + "总数据数" + CStr(O) + "【正
常】" + Chr$(13) + Chr$(10)
MeasurementData = True
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 返回 MODE 众数
Dim Num As Double
Dim obj As New Array1D002
```

```
Dim a As Boolean
Dim R1() As Double
  a = obj.Array1D_CDataTypeLtoD(Return_Time(), R1())
Num = Max(R1())
For i = LBound(Return_Time) To UBound(Return_Time)
If Num = Return_Time(i) Then
Return_Mode = Return_ClassMiddleValue(i)
End If
Next
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  最终报告
Return_ReportTXT = Return_ReportTXT + "全距=" + CStr(Range(Data())) + Chr$(13) + Chr$(10)
Return_ReportTXT = Return_ReportTXT + "组距=" + CStr(Return_ClassInterval) + Chr$(13) +
Chr$(10)
Return_ReportTXT = Return_ReportTXT + "组数=" + CStr(ClassNumber) + Chr$(13) + Chr$(10)
Return_ReportTXT = Return_ReportTXT + "中位数修正值（不包括中位数）=" +
CStr(ClassMiddleValueModification) + Chr$(13) + Chr$(10)
Return_ReportTXT = Return_ReportTXT + "第一组下限=" + CStr(FirstClassLowerLimit) + Chr$(13)
+ Chr$(10)
Return_ReportTXT = Return_ReportTXT + "第一组中位数=" + CStr(FirstClassMidValue) + Chr$(13)
+ Chr$(10)
Return_ReportTXT = Return_ReportTXT + "第一组上限=" + CStr(Return_ClassUpperLimit(1)) +
Chr$(13) + Chr$(10)
Return_ReportTXT = Return_ReportTXT + "众数=" + CStr(Return_Mode) + Chr$(13) + Chr$(10)
Return_ReportTXT = Return_ReportTXT + "小数位数=" + CStr(DecimalDigits) + Chr$(13) +
Chr$(10)
Return_ReportTXT = Return_ReportTXT + "程序完结"
End Function
```

'CountData 计数资料

Rem CountData 计数资料

```
Public Function CountData(ByRef Data() As Double, ByRef Return_Class() As Double, _
ByRef Return_Time() As Long, ByRef Return_ReportTXT As String) As Boolean
Dim i As Long
   Rem  分组数小于 30 的时候不用分组提示
   Dim Array1D_Num As Long
   For i = LBound(Data) To UBound(Data)
   Array1D_Num = Array1D_Num + 1
   Next
   If Array1D_Num < 30 Then
   Return_ReportTXT = Return_ReportTXT + "总组数=" + CStr(Array1D_Num) + "<30 " + "【不用分
组】" + Chr$(13) + Chr$(10)
```

```
    End If
  '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim obj As New Array1D001
Dim DNum As Long
DNum = obj.Array1D_DifferentNum(Data(), Return_Class(), Return_Time())
'DNum =  分组数
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
End Function
```

'QualitativeCharacteristicsData 质量性状资料、半定量（等级）资料的整理

Rem 质量性状资料、半定量（等级）资料的整理

```
Public Function QualitativeCharacteristicsData(ByRef Time() As Long, _
ByRef Return_Frequency() As Double) As Boolean
Dim Sum As Double
Dim obj As New Array1D001
Dim R() As Double
Dim a As Boolean
a = obj.Array1D_CDataTypeLtoD(Time(), R())
Sum = obj.Array1D_Sum(R())
Dim i As Long
ReDim Preserve Return_Frequency(LBound(R) To UBound(R))
For i = LBound(R) To UBound(R)
Return_Frequency(i) = R(i) / Sum
Next
QualitativeCharacteristicsData = True
End Function
```

'HarmonicMean 调和平均数

Rem 调和平均数（harmonic mean）

```
Public Function HarmonicMean(ByRef Data() As Double) As Double
Dim Sum As Double, n As Long, i As Long
Dim Part1 As Double
For i = LBound(Data) To UBound(Data)
Sum = Sum + 1 / Data(i)
n = n + 1
Next
Part1 = Sum / n
HarmonicMean = 1 / Part1
```

```
End Function
```

'HarmonicAverage 调和平均数

```
Public Function HarmonicAverage(ByRef Data() As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
For i = LBound(Data()) To UBound(Data())
Sum1 = Sum1 + (1 / Data(i))
n = n + 1
Next i
HarmonicAverage = 1 / (Sum1 * (1 / n))
End Function
```

'BernoulliTrials 二项式

Rem 二项式

```
Public Function BernoulliTrials(ByVal n As Long, ByVal P As Double, ByVal X As Long) As Double
BernoulliTrials = Factorial(n) / (Factorial(X) * Factorial(n - X)) * (P ^ X) * ((1 - P) ^ (n - X))
End Function
```

'PoissonDistributionMean 泊松分布平均值

```
Public Function PoissonDistributionMean(ByRef k() As Long, ByRef F() As Long) As Double
  Dim Sum As Double, Part1 As Double
  Dim i As Long
  For i = LBound(F) To UBound(F)
  Part1 = Part1 + k(i) * F(i)
  Sum = Sum + F(i)
  Next
  PoissonDistributionMean = Part1 / Sum
End Function
```

'PoissonDistributionMeanSquare 泊松分布均方

```
Public Function PoissonDistributionMeanSquare(ByRef k() As Long, ByRef F() As Long) As Double
Dim i As Long
Dim Part1 As Double, Part2 As Double, Sum As Double
  For i = LBound(F) To UBound(F)
  Part1 = Part1 + F(i) * (k(i) ^ 2)
  Part2 = Part2 + k(i) * F(i)
```

```
 Sum = Sum + F(i)
 Next
 PoissonDistributionMeanSquare = (Part1 - Part2 ^ 2 / Sum) / (Sum - 1)
End Function
```

'PoissonDistributionTable 泊松分布表

Rem 泊松分布表

```
Public Function PoissonDistributionTable(ByRef k() As Long, ByRef F() As Long, _
ByRef Return_Frequency() As Double, ByRef Return_Probability() As Double, _
ByRef Return_TheoreticalNumber() As Double) As Boolean
Dim i As Long, n As Long
Dim Data() As Double
Dim Sum As Double
'For i = LBound(k) To UBound(k)
'n = n + 1
'ReDim Preserve data(n)
'data(n) = f(i)
'Debug.Print data(n)
'Next
Dim a As Boolean
Dim f1() As Double
Dim k1() As Double
Dim obj As New Array1D001
a = obj.Array1D_CDataTypeLtoD(F(), f1())
a = obj.Array1D_CDataTypeLtoD(k(), k1())
n = 0

For i = LBound(k) To UBound(k)
Sum = Sum + F(i)
Next

Dim Test1 As Double, Test2 As Double, Test3 As Double
Dim Part1 As Double, Part2 As Double
For i = LBound(k) To UBound(k)
n = n + 1
ReDim Preserve Return_Frequency(n)
ReDim Preserve Return_Probability(n)
ReDim Preserve Return_TheoreticalNumber(n)
Return_Probability(n) = PoissonDistribution(PoissonDistributionMean(k(), F()), k(i))
Return_Frequency(n) = F(i) / Sum
Return_TheoreticalNumber(n) = Return_Probability(n) * Sum
'Debug.Print "p=" & Return_Probability(n) & "    F=" & Return_Frequency(n) & "    TN=" &
```

```
Return_TheoreticalNumber(n)
Next



Rem 最后的一组值是>=k(最后值)，所以用 1-前面的总值
For i = 1 To (n - 1)
Part1 = Part1 + Return_Probability(i)
Part2 = Part2 + Return_TheoreticalNumber(i)
Next
Return_Probability(n) = 1 - Part1
Return_TheoreticalNumber(n) = Sum - Part2

PoissonDistributionTable = True
End Function
```

'PoissonDistribution 泊松分布

Rem 泊松分布计算

```
Public Function PoissonDistribution(ByVal λ As Double, ByVal k As Double) As Double
PoissonDistribution = (λ ^ k / Factorial(k)) * (conE ^ (-λ))
End Function
```

Rem 泊松分布判断

```
Public Function PoissonDistributionEstimate()


End Function
```

'PoissonDistributionPosibility 泊松分布概率值

Rem 波松分布
Rem 波松分布作为一种离散型随机变量的概率分布有一个重要的特征，这就是它的平均数和方差相等，_
Rem 都等于常数λ，即μ=σ2=λ。利用这一特征， 可以初步判断一个离散型随机变量是否服从波松分布。
Rem 分布用 **2** 和 **3** 号计算方法不准确

```
Public Function PoissonDistributionPosibility(ByVal λ As Double, ByVal k As Double, _
ByVal CalculateType As Long, ByRef Return_RecordForShape() As Double) As Double
Dim PrecisionPieces As Long
Dim Distance    As Double
Dim Part1 As Double, i As Long, X As Double

If (CalculateType <> 1 And CalculateType <> 2 And CalculateType <> 3) Then
CalculateType = 1
```

```
End If

PrecisionPieces = k ' 因为阶乘都是整数
'Distance = K / PrecisionPieces
Distance = 1

ReDim Preserve Return_RecordForShape(1 To PrecisionPieces)
For i = 0 To PrecisionPieces - 1
X = Distance * i
Part1 = (λ ^ X / Factorial(X)) * (conE ^ (-λ))
Return_RecordForShape(i + 1) = Part1
Next

Rem 矩形法
Rem 波松分布首选计算方法
If CalculateType = 1 Then
For i = 0 To PrecisionPieces - 1
X = Distance * i
Part1 = (λ ^ X / Factorial(X)) * (conE ^ (-λ))
PoissonDistributionPosibility = PoissonDistributionPosibility + Part1 * Distance
Next
End If

Rem 梯形法
Dim Part2 As Double, X1 As Double
If CalculateType = 2 Then
For i = 0 To PrecisionPieces - 1
X = Distance * i
X1 = Distance * (i + 1)
Part1 = (λ ^ X / Factorial(X)) * (conE ^ (-λ))
Part2 = (λ ^ X1 / Factorial(X1)) * (conE ^ (-λ))
'Part1 = x: Part2 = x1 '实验程序是否正确的代码
PoissonDistributionPosibility = PoissonDistributionPosibility + ((Part1 + Part2) * Distance) / 2
'Debug.Print x & "=x " & BernoulliTrials& "=NDP" & "              2"
Next i
End If

Rem 抛物线法
Dim Part3 As Double, X2 As Double
If CalculateType = 3 Then
For i = 0 To PrecisionPieces - 1 Step 2
X = Distance * i
X1 = Distance * (i + 1)
X2 = Distance * (i + 2)
```

```
Part1 = (λ ^ X / Factorial(X)) * (conE ^ (-λ))
Part2 = (λ ^ X1 / Factorial(X1)) * (conE ^ (-λ))
Part3 = (λ ^ X2 / Factorial(X2)) * (conE ^ (-λ))
'Part1 = x: Part2 = x1: Part3 = x2 '实验程序是否正确的代码

If (PrecisionPieces Mod 2) = 0 Then
        PoissonDistributionPosibility = PoissonDistributionPosibility + 1 / 3 * (Part1 + 4 * Part2 +
Part3) * Distance
        'Debug.Print x & "=x " & BernoulliTrials& "=NDP" & "             3"
        'Debug.Print 1 / 3 * (Part1 + 4 * Part2 + Part3) * Distance
End If

If (PrecisionPieces Mod 2 <> 0) Then '奇数
        If (i <> PrecisionPieces - 1) Then
        PoissonDistributionPosibility = PoissonDistributionPosibility + 1 / 3 * (Part1 + 4 * Part2 +
Part3) * Distance
        'Debug.Print x & "=x " & BernoulliTrials& "=NDP" & "             3"
        End If
        If (i = PrecisionPieces - 1) Then
        PoissonDistributionPosibility = PoissonDistributionPosibility + ((Part1 + Part2) * Distance)
/ 2
        'Debug.Print x & "=x " & BernoulliTrials& "=NDP" & "             3"
        End If
End If
Next i
End If
DoEvents
End Function
```

'BernoulliTrialsPosibility 二项式分布概率

Rem 二项式分布图

```
Public Function BernoulliTrialsPosibility(ByVal n As Long, ByVal P As Double, _
ByVal k As Long, ByVal CalculateType As Long, ByRef Return_RecordForShape() As Double) As
Double
Dim PrecisionPieces As Long
Dim Distance    As Double
Dim Part1 As Double, i As Long, X As Double

If (CalculateType <> 1 And CalculateType <> 2 And CalculateType <> 3) Then
CalculateType = 1
End If
```

```
PrecisionPieces = k '  因为阶乘都是整数
'Distance = K / PrecisionPieces
Distance = 1
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  阶乘运算范围
If (k > 171) Then
MsgBox "无法计算 171 以上数字阶乘"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  图像
ReDim Preserve Return_RecordForShape(1 To PrecisionPieces)
For i = 0 To PrecisionPieces - 1
X = Distance * i
Part1 = BernoulliTrials(n, P, X)
Return_RecordForShape(i + 1) = Part1
Next

Rem  矩形法
If CalculateType = 1 Then
For i = 0 To PrecisionPieces - 1
X = Distance * i
Part1 = BernoulliTrials(n, P, X)
BernoulliTrialsPosibility = BernoulliTrialsPosibility + Part1 * Distance
Next
End If

Rem  梯形法
Dim Part2 As Double, X1 As Double
If CalculateType = 2 Then
For i = 0 To PrecisionPieces - 1
X = Distance * i
X1 = Distance * (i + 1)
Part1 = BernoulliTrials(n, P, X)
Part2 = BernoulliTrials(n, P, X1)
'Part1 = x: Part2 = x1 '实验程序是否正确的代码
BernoulliTrialsPosibility = BernoulliTrialsPosibility + ((Part1 + Part2) * Distance) / 2
'Debug.Print x & "=x " & BernoulliTrials& "=NDP" & "               2"
Next i
End If

Rem  抛物线法
Dim Part3 As Double, X2 As Double
If CalculateType = 3 Then
For i = 0 To PrecisionPieces - 1 Step 2
```

```
X = Distance * i
X1 = Distance * (i + 1)
X2 = Distance * (i + 2)
Part1 = BernoulliTrials(n, P, X)
Part2 = BernoulliTrials(n, P, X1)
Part3 = BernoulliTrials(n, P, X2)
'Part1 = x: Part2 = x1: Part3 = x2 '实验程序是否正确的代码

If (PrecisionPieces Mod 2) = 0 Then
        BernoulliTrialsPosibility = BernoulliTrialsPosibility + 1 / 3 * (Part1 + 4 * Part2 + Part3) *
Distance
        'Debug.Print x & "=x " & BernoulliTrials& "=NDP" & "              3"
        'Debug.Print 1 / 3 * (Part1 + 4 * Part2 + Part3) * Distance
End If

If (PrecisionPieces Mod 2 <> 0) Then '奇数
        If (i <> PrecisionPieces - 1) Then
        BernoulliTrialsPosibility = BernoulliTrialsPosibility + 1 / 3 * (Part1 + 4 * Part2 + Part3) *
Distance
        'Debug.Print x & "=x " & BernoulliTrials& "=NDP" & "              3"
        End If
        If (i = PrecisionPieces - 1) Then
        BernoulliTrialsPosibility = BernoulliTrialsPosibility + ((Part1 + Part2) * Distance) / 2
        'Debug.Print x & "=x " & BernoulliTrials& "=NDP" & "              3"
        End If
End If
Next i
End If
DoEvents
End Function
```

'Factorial 阶乘

Rem 阶乘

```
Public Function Factorial(ByVal n As Long) As Double
Dim i As Long
Factorial = 1
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Part1 As Double, Part2 As Double
Part1 = 1
Part2 = 1
If (n > 171) Then
'MsgBox "无法计算 171 以上数字阶乘"
```

```
        If (n Mod 2 = 1) Then
          For i = 1 To (n / 2 + 0.5)
          Part1 = Part1 * i
          Next
          For i = (n / 2 + 0.5 + 1) To n
          Part2 = Part2 * i
          Next
          Factorial = Part1 * Part2
        End If

        If (n Mod 2 = 0) Then
        For i = 1 To (n / 2)
        Part1 = Part1 * i
        Next
        For i = (n / 2 + 1) To n
          Part2 = Part2 * i
        Next
        Factorial = Part1 * Part2
        End If
Else
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~`
For i = 1 To n
Factorial = Factorial * i
Next
End If
End Function
```

'μ 服从二项分布 B(n,p)的随机变量之平均数μ

Rem 服从二项分布 **B(n,p)**的随机变量之平均数**μ**

```
Public Function μ(ByVal n As Long, ByVal P As Double) As Double
μ = n * P
End Function
```

'Squareσ总体方差

Rem 总体方差

```
Public Function Squareσ(ByRef Data() As Double)
Dim Sum As Double, Total As Long, AverageNumber As Double, Part1 As Double
For i = LBound(Data) To UBound(Data)
Sum = Sum + Data(i)
```

```
Next i
Total = UBound(Data) - LBound(Data) + 1
AverageNumber = Sum / Total
For i = LBound(Data) To UBound(Data)
Part1 = Part1 + (Data(i) - AverageNumber) ^ 2
Next i
Squareσ = Part1 / Total
End Function
```

'σ 二项分布的标准差σ

Rem 二项分布的标准差σ

```
Public Function σ(ByVal n As Long, ByVal P As Double) As Double
  σ = Sqr(n * P * (1 - P))
End Function
```

'σp 总体百分数标准误

Rem 也称为总体百分数标准误

```
Public Function σp(ByVal n As Long, ByVal P As Double) As Double
σp = Sqr(P * (1 - P) / n)
End Function
```

'SP 常以样本百分数来估计

Rem 常以样本百分数 来估计
Rem 称为样本百分数标准误

```
Public Function SP(ByVal n As Long, ByVal SamplePercentage As Double) As Double
SP = Sqr(SamplePercentage * (1 - SamplePercentage) / n)
End Function
```

'TestOfSignificance_t T 显著检验

Rem 中心极限定理
Rem 它是概率论中最重要的一类定理，有广泛的实际应用背景。在自然界与生产中，
Rem 一些现象受到许多相互独立的随机因素的影响，如果每个因素所产生的影响都很微小时，
Rem 总的影响可以看作是服从正态分布的。中心极限定理就是从数学上证明了这一现象。

```
Public Function TestOfSignificance_t(ByRef Data1() As Double, _
ByRef Data2() As Double, ByVal WidthOfFunction As Double, ByVal μ1 As Double, ByVal μ2 As
```

```
Double, _
ByVal PrecisionPieces As Long, ByVal CalculateType As Long, _
ByRef Return_df As Long, ByRef Return_T As Double) As Boolean
Dim obj As New T001
Dim X1 As Double, X2 As Double
Dim Part1 As Double, Part2 As Double
Dim n1 As Long, n2 As Long
Dim S As Double
Dim R As Double
X1 = ArithmeticMean(Data1, R)
X2 = ArithmeticMean(Data2, R)
Dim i As Long
For i = LBound(Data1) To UBound(Data1)
Part1 = Part1 + (Data1(i) - X1) ^ 2
n1 = n1 + 1
Next
For i = LBound(Data2) To UBound(Data2)
Part2 = Part2 + (Data2(i) - X2) ^ 2
n2 = n2 + 1
Next
S = Sqr((Part1 + Part2) / ((n1 - 1) + (n2 - 1)) * (1 / n1 + 1 / n2))
Return_T = (X1 - X2) / S
Return_df = (n1 - 1) + (n2 - 1)

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim R1() As Double
Dim u1 As Double, u2 As Double
Dim p1 As Double, p2 As Double, p3 As Double
Dim Qtest As Long
If PrecisionPieces > 500 Then
'Select Case Qtest
'Case 10
'P1 = t_Distribution_μα(Return_df, 0.05, Qtest, 20, u1, u2, r1(), CalculateType)
'P2 = t_Distribution_μα(Return_df, 0.001, Qtest, 20, u1, u2, r1(), CalculateType)
'End Select
Qtest = 10
For i = 1 To 3
p1 = obj.t_Distribution_μα(Return_df, 0.05, Qtest, WidthOfFunction, u1, u2, R1(), CalculateType)
p2 = obj.t_Distribution_μα(Return_df, 0.001, Qtest, WidthOfFunction, u1, u2, R1(),
CalculateType)
If (Return_T >= p1 And Return_T <= p2) Then
    If i = 1 Then
    Qtest = 100
    End If
```

```
        If i = 2 Then
        Qtest = 500
        End If
        If i = 3 Then
        Qtest = PrecisionPieces
        End If
Else
Exit For
End If
Next
End If
'Qtest = 100
'P1 = t_Distribution_μα(Return_df, 0.05, Qtest, 20, u1, u2, r1(), CalculateType)
'P2 = t_Distribution_μα(Return_df, 0.001, Qtest, 20, u1, u2, r1(), CalculateType)
'If Return_t > P1 And Return_t < P2 Then
'Qtest = 500
'P1 = t_Distribution_μα(Return_df, 0.05, Qtest, 20, u1, u2, r1(), CalculateType)
'P2 = t_Distribution_μα(Return_df, 0.001, Qtest, 20, u1, u2, r1(), CalculateType)
'End If
'End If
'End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TestOfSignificance_t = True
i = 0
DoEvents
End Function
```

'nolinearR 非线性拟合度 R 值

Rem 非线性拟合度 R 值
Rem Data() 原始数据
Rem FittedValue() 拟合数据
Rem 要求原始值和拟合值数量一样

```
Public Function nolinearR(ByRef Data() As Double, ByRef FittedValue() As Double) As Double
Dim Num As Long, Num1 As Long
Num1 = UBound(FittedValue) - LBound(FittedValue) + 1
Num = UBound(Data) - LBound(Data) + 1
If (Num <> Num1) Then
MsgBox "数据有问题，退出程序"
nolinearR = -2
Exit Function
End If
Dim i As Long
```

```
Dim Part1 As Double
Dim Sum1 As Double
Dim Mean As Double
Dim Part2 As Double

For i = 1 To Num
    Part1 = Part1 + (Data(i) - FittedValue(i)) ^ 2
    Sum1 = Sum1 + Data(i)
    'Totalsample = Totalsample + 1
Next i
Mean = Sum1 / Num
For i = 1 To Num
    Part2 = Part2 + (Data(i) - Mean) ^ 2
Next i
nolinearR = (1 - Part1 / Part2) ^ (1 / 2)
End Function
```

'LinearRegressionAnalysis_SPARE 线性回归程序（备用）

'Dim Totalsample As Double
'线性回归分析（备用方法）
Rem 使用的须知
'1.某性状作为自变量或依变量的确定等等，都必须由生物科学相应的专业知识来决定，并且还要用到生物科学实践中去检验。如果不以一定的生物科学依据为前提，把风马牛不相及的资料随意凑到一块作直线回归分析或相关分析，那将是根本性的错误。
'2.在研究两个变量间关系时，要求其余变量应尽量保持在同一水平.例如人的身高和胸围之间的关系，如果体重固定，身高越高的人，胸围越小，但当体重在变化时，其结果就会相反。
'3.一般至少有 5 对以上的观测值
'4.外推要谨慎，否则会得出错误的结果。
'5.一个不显著的相关系数并不意味着变量 x 和 y 之间没有关系，而只有能说明两变量间没有显著的直线关系；一个显著的相关系数或回归系数亦并不意味着 x 和 y 的关系必定为直线，因为并不排除有能够更好地描述它们关系的非线性方程的存在。
'6.一个显著的回归方程并不一定具有实践上的预测意义　如一个资料 x、y 两个变量间的相关系数 r=0.5，在 df=24 时，r0.01(24) =0.496，r>r0.01(24)，表明相关系数极显著。而 r2=0.25，即 x 变量或 y 变量的总变异能够通过 y 变量或 x 变量以直线回归的关系来估计的比重只占 25%，其余的 75%的变异无法借助直线回归来估计。

```
Public Function LinearRegressionAnalysis_SPARE(ByRef DataX() As Double, ByRef DataY() As Double, _
ByRef Return_SSx As Double, ByRef Return_SPxy As Double, ByRef Return_SSy As Double, _
ByRef Return_Syx As Double, ByRef Return_Fitting() As Double, ByRef Return_A As Double, _
ByRef Return_B As Double, ByRef Return_r As Double, ByRef Return_SSr As Double, ByRef Return_SS_r As Double, _
```

```
ByRef Return_dfR As Double, ByRef Return_dfy As Double, ByRef Return_df_r As Double, ByRef
Return_F As Double, ByRef Return_MSR As Double, _
ByRef Return_MS_r As Double, ByRef Return_F001 As Double, ByRef Return_F005 As Double, _
ByRef Return_FReport As String) As Boolean
Dim MeanX As Double
Dim MeanY As Double
Dim Sum1 As Double
Dim Sum2 As Double
Dim Part1 As Double, Part2 As Double, Part3 As Double, Part4 As Double, Part5 As Double

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Num As Long, Num1 As Long
Num1 = UBound(DataY) - LBound(DataY) + 1
Num = UBound(DataX) - LBound(DataX) + 1
If (Num <> Num1) Then
MsgBox "数据有问题，退出程序"
LinearRegressionAnalysis_SPARE = False
Exit Function
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MeanX = ArithmeticMean(DataX(), Sum1)
MeanY = ArithmeticMean(DataY(), Sum2)
Dim i As Long
For i = LBound(DataX) To UBound(DataX)
Part1 = Part1 + DataX(i) ^ 2
Part2 = Part2 + DataX(i)
Next
Return_SSx = Part1 - Part2 ^ 2 / Num
For i = LBound(DataX) To UBound(DataX)
Part3 = Part3 + DataX(i) * DataY(i)
Part4 = Part4 + DataY(i)
Next
Return_SPxy = Part3 - Part2 * Part4 / Num
For i = LBound(DataX) To UBound(DataX)
Part5 = Part5 + DataY(i) ^ 2
Next
'y 的总变异程度，称为 y 的总平方和
Return_SSy = Part5 - Part4 ^ 2 / Num
Return_B = Return_SPxy / Return_SSx
Return_A = MeanY - Return_B * MeanX
Return_r = Return_SPxy / Sqr(Return_SSx * Return_SSy)
Dim Part6 As Double, Part7 As Double, Part8 As Double
Dim n As Long
For i = LBound(DataX) To UBound(DataX)
```

```
n = n + 1
ReDim Preserve Return_Fitting(n)
Return_Fitting(n) = Return_A + Return_B * DataX(i)
Part6 = Part6 + (DataY(i) - Return_Fitting(n)) ^ 2
Part7 = Part7 + (Return_Fitting(n) - MeanY) ^ 2
Next
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'直线回归的偏离度估计
'离回归均方的平方根叫离回归标准误，记为 Syx
Return_Syx = Sqr(Part6 / (Num - 2))
LinearRegressionAnalysis_SPARE = True
'y 与 x 间存在直线关系所引起的 y 的变异程度，称为回归平方和
Rem 下面没有验证!!!!!!!!!!!!!!!!!!
Return_SSr = Part7
Return_SS_r = Return_SSy - Return_SSr
Return_dfR = 1
Return_dfy = Num - 1
Return_df_r = Num - 2

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Return_F = Return_SSr / (Return_SS_r / (Num - 2))
Return_MSR = Return_SSr / Return_dfR
Return_MS_r = Return_SS_r / Return_df_r
Dim obj As New F001
Return_F001 = obj.PFaN1N2X2(0.01, 0.00001, Return_dfR, Return_df_r) '0.00001（1,4）运算时间 30’
Return_F005 = obj.PFaN1N2X2(0.05, 0.00001, Return_dfR, Return_df_r) '0.00001（1,4）运算时间 30’
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
If (Return_F > Return_F001) Then
Return_FReport = "显著的直线关系"
ElseIf (Return_F001 > Return_F > Return_F005) Then
Return_FReport = "当 AERF=0.05 有显著的直线关系，AERF=0.01 无显著直线关系"
ElseIf (Return_F005 > Return_F) Then
Return_FReport = "无显著的直线关系"
End If
End Function
```

'LinearRegressionAnalysis 线性回归程序

Rem 线性回归分析
Rem 书上的数据通过测试，但是还需要更多的例子来测试，应为有 IF 的情况。
Rem 大体上可以用了

Rem F001 没有加到类模块里的情况

```
Public Function LinearRegressionAnalysis(ByRef DataX() As Double, ByRef DataY() As Double) As String
Dim MeanX As Double
Dim MeanY As Double
Dim Sum1 As Double
Dim Sum2 As Double
Dim Part1 As Double, Part2 As Double, Part3 As Double, Part4 As Double, Part5 As Double
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Num As Long, Num1 As Long
Num1 = UBound(DataY) - LBound(DataY) + 1
Num = UBound(DataX) - LBound(DataX) + 1
If (Num <> Num1) Then
MsgBox "数据有问题，退出程序"
LinearRegressionAnalysis = "None"
Exit Function
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

MeanX = ArithmeticMean(DataX(), Sum1)
MeanY = ArithmeticMean(DataY(), Sum2)
Dim i As Long
For i = LBound(DataX) To UBound(DataX)
Part1 = Part1 + DataX(i) ^ 2
Part2 = Part2 + DataX(i)
Next
Dim Return_SSx As Double
Return_SSx = Part1 - Part2 ^ 2 / Num
For i = LBound(DataX) To UBound(DataX)
Part3 = Part3 + DataX(i) * DataY(i)
Part4 = Part4 + DataY(i)
Next
Dim Return_SPxy As Double
Return_SPxy = Part3 - Part2 * Part4 / Num
For i = LBound(DataX) To UBound(DataX)
Part5 = Part5 + DataY(i) ^ 2
Next

'y 的总变异程度，称为 y 的总平方和
Dim Return_SSy As Double
Dim Return_B As Double
Dim Return_A As Double

Return_SSy = Part5 - Part4 ^ 2 / Num
```

```
Return_B = Return_SPxy / Return_SSx
Return_A = MeanY - Return_B * MeanX
Dim Return_r As Double
Return_r = Return_SPxy / Sqr(Return_SSx * Return_SSy)

Dim Part6 As Double, Part7 As Double, Part8 As Double
Dim n As Long
Dim Return_Fitting() As Double
For i = LBound(DataX) To UBound(DataX)
n = n + 1
ReDim Preserve Return_Fitting(n)
Return_Fitting(n) = Return_A + Return_B * DataX(i)
Part6 = Part6 + (DataY(i) - Return_Fitting(n)) ^ 2
Part7 = Part7 + (Return_Fitting(n) - MeanY) ^ 2
LinearRegressionAnalysis = LinearRegressionAnalysis & "拟合数据" & n & "=" & Return_Fitting(n)
& Chr(13) & Chr(10)
Next
n = 0
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'直线回归的偏离度估计
'离回归均方的平方根叫离回归标准误，记为 Syx
Dim Return_Syx As Double
Return_Syx = Sqr(Part6 / (Num - 2))
'y 与 x 间存在直线关系所引起的 y 的变异程度，称为回归平方和
Dim Return_SSr    As Double
Dim Return_SS_r As Double
Dim Return_dfR As Long
Dim Return_dfy As Long
Dim Return_df_r As Long
Rem  没有验证
Return_SSr = Part7
Return_SS_r = Return_SSy - Return_SSr
Return_dfR = 1
Return_dfy = Num - 1
Return_df_r = Num - 2
'F 检验~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Return_F As Double
Dim Return_MSR As Double
Dim Return_MS_r As Double
Return_F = Return_SSr / (Return_SS_r / (Num - 2))
Return_MSR = Return_SSr / Return_dfR
Return_MS_r = Return_SS_r / Return_df_r
Dim Return_F001 As Double, Return_F005 As Double
Dim obj As New F001
```

```
Rem 这个精度有点小，但是在增加运算量将会变慢，还要找加速方法提高精度 2016-6-20
'*********************************************************************
'Return_F001 = obj.PFaN1N2X2(0.01, 0.00001, Return_dfR, Return_df_r)
'Return_F005 = obj.PFaN1N2X2(0.05, 0.00001, Return_dfR, Return_df_r)
Rem 16-9-7
Return_F001 = obj.PFaN1N2X2(0.01, 10000000, Return_dfR, Return_df_r) ' 0.01 用千万
Return_F005 = obj.PFaN1N2X2(0.05, 1000000, Return_dfR, Return_df_r)     ' 0.05 用百万
'*********************************************************************
Dim Return_FReport As String
If (Return_F > Return_F001) Then
Return_FReport = "显著的直线关系"
ElseIf (Return_F001 > Return_F And Return_F > Return_F005) Then
Return_FReport = "当 AERF=0.05 有显著的直线关系，AERF=0.01 无显著直线关系"
ElseIf (Return_F005 > Return_F) Then
Return_FReport = "无显著的直线关系"
End If
'T 检验~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Return_Sb As Double, Return_T As Double, u1 As Double, u2 As Double, Shape() As Double
Dim Return_T001 As Double, Return_T005 As Double
Dim obj1 As New T001
Return_Sb = Return_Syx / Sqr(Return_SSx)
Return_T = Return_B / Return_Sb
'Return_T001 = obj1.t_DistributionRectangle_μα(Num - 2, 0.01, 100, 20, u1, u2, Shape()) '数据问题较大
'Return_T005 = obj1.t_DistributionRectangle_μα(Num - 2, 0.05, 100, 20, u1, u2, Shape())
Rem 16-9-7
Return_T001 = obj1.t(0.005, Return_df_r) '电子书上是 T（0.01 ， 10）=3.169  恰好是 t(0.005, Return_df_r)，原因可能是单双尾
Return_T005 = obj1.t(0.025, Return_df_r)
'*********************************************************************
Rem 16-9-7  参照田间统计，T 取绝度值
Dim Return_T1    As Double
Return_T1 = Abs(Return_T)
'*********************************************************************
Dim Return_TReport As String
If (Return_T1 > Return_T001) Then
Return_TReport = "显著的直线关系"
ElseIf (Return_T001 > Return_T1 And Return_T1 > Return_T005) Then
Return_TReport = "当 AERF=0.05 有显著的直线关系，AERF=0.01 无显著直线关系"
ElseIf (Return_T005 > Return_T1) Then
Return_TReport = "无显著的直线关系"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'直线回归的区间估计  2016-6-20  下面的程序未写入备用程序中
```

```
Rem α是当 x=0 时 y^的值，即直线在 y 轴上的截距，叫回归截距。
Dim Return_Sa As Double
Dim Return_LαConfidenceInterval95Percent    As Double,
Return_UαConfidenceInterval95Percent As Double
Dim Return_LαConfidenceInterval99Percent    As Double,
Return_UαConfidenceInterval99Percent As Double
Return_Sa = Return_Syx * Sqr(1 / Num + MeanX ^ 2 / Return_SSx)
'Return_LαConfidenceInterval95Percent = Return_a - Return_Sa * 2.228
'Return_UαConfidenceInterval95Percent = Return_a + Return_Sa * 2.228
'Return_LαConfidenceInterval99Percent = Return_a - Return_Sa * 3.169
'Return_UαConfidenceInterval99Percent = Return_a + Return_Sa * 3.169
Return_LαConfidenceInterval95Percent = Return_A - Return_Sa * Return_T005
Return_UαConfidenceInterval95Percent = Return_A + Return_Sa * Return_T005
Return_LαConfidenceInterval99Percent = Return_A - Return_Sa * Return_T001
Return_UαConfidenceInterval99Percent = Return_A + Return_Sa * Return_T001
'******************
'总体回归系数β的置信区间
Dim Return_LβConfidenceInterval95Percent    As Double,
Return_UβConfidenceInterval95Percent As Double
Dim Return_LβConfidenceInterval99Percent    As Double,
Return_UβConfidenceInterval99Percent As Double
Return_LβConfidenceInterval95Percent = Return_B - Return_Sb * Return_T005
Return_UβConfidenceInterval95Percent = Return_B + Return_Sb * Return_T005
Return_LβConfidenceInterval99Percent = Return_B - Return_Sb * Return_T001
Return_UβConfidenceInterval99Percent = Return_B + Return_Sb * Return_T001
'Return_LβConfidenceInterval95Percent = Return_b - Return_Sb * 2.228
'Return_UβConfidenceInterval95Percent = Return_b + Return_Sb * 2.228
'Return_LβConfidenceInterval99Percent = Return_b - Return_Sb * 3.169
'Return_UβConfidenceInterval99Percent = Return_b + Return_Sb * 3.169
'******************
'总体平均数α+βx 的置信区间
Dim Return_LPopulationMeanConfidenceInterval95Percent() As Double
Dim Return_UPopulationMeanConfidenceInterval95Percent() As Double
Dim Return_LPopulationMeanConfidenceInterval99Percent() As Double
Dim Return_UPopulationMeanConfidenceInterval99Percent() As Double
Dim Return_SFittingy() As Double
For i = LBound(DataX) To UBound(DataX)
n = n + 1
ReDim Preserve Return_LPopulationMeanConfidenceInterval95Percent(n)
ReDim Preserve Return_UPopulationMeanConfidenceInterval95Percent(n)
ReDim Preserve Return_LPopulationMeanConfidenceInterval99Percent(n)
ReDim Preserve Return_UPopulationMeanConfidenceInterval99Percent(n)
ReDim Preserve Return_SFittingy(n)
Return_SFittingy(i) = Return_Syx * Sqr(1 / Num + (DataX(i) - MeanX) ^ 2 / Return_SSx)
```

```
Return_LPopulationMeanConfidenceInterval95Percent(n) = Return_Fitting(i) - Return_SFittingy(i)
* Return_T005
Return_UPopulationMeanConfidenceInterval95Percent(n) = Return_Fitting(i) + Return_SFittingy(i)
* Return_T005
Return_LPopulationMeanConfidenceInterval99Percent(n) = Return_Fitting(i) - Return_SFittingy(i)
* Return_T001
Return_UPopulationMeanConfidenceInterval99Percent(n) = Return_Fitting(i) + Return_SFittingy(i)
* Return_T001
'Return_LPopulationMeanConfidenceInterval95Percent(n) = Return_Fitting(i) - Return_SFittingy(i)
* 2.228
'Return_UPopulationMeanConfidenceInterval95Percent(n) = Return_Fitting(i) +
Return_SFittingy(i) * 2.228
'Return_LPopulationMeanConfidenceInterval99Percent(n) = Return_Fitting(i) - Return_SFittingy(i)
* 3.169
'Return_UPopulationMeanConfidenceInterval99Percent(n) = Return_Fitting(i) +
Return_SFittingy(i) * 3.169
Next
n = 0
'********************
'单个 y 值的置信区间
Dim Return_LyConfidenceInterval95Percent() As Double
Dim Return_UyConfidenceInterval95Percent() As Double
Dim Return_LyConfidenceInterval99Percent() As Double
Dim Return_UyConfidenceInterval99Percent() As Double
Dim Return_Sy() As Double
For i = LBound(DataX) To UBound(DataX)
n = n + 1
ReDim Preserve Return_LyConfidenceInterval95Percent(n)
ReDim Preserve Return_UyConfidenceInterval95Percent(n)
ReDim Preserve Return_LyConfidenceInterval99Percent(n)
ReDim Preserve Return_UyConfidenceInterval99Percent(n)
ReDim Preserve Return_Sy(n)
Return_Sy(i) = Return_Syx * Sqr(1 + 1 / Num + (DataX(i) - MeanX) ^ 2 / Return_SSx)
Return_LyConfidenceInterval95Percent(n) = Return_Fitting(i) - Return_Sy(i) * Return_T005
Return_UyConfidenceInterval95Percent(n) = Return_Fitting(i) + Return_Sy(i) * Return_T005
Return_LyConfidenceInterval99Percent(n) = Return_Fitting(i) - Return_Sy(i) * Return_T001
Return_UyConfidenceInterval99Percent(n) = Return_Fitting(i) + Return_Sy(i) * Return_T001
'Return_LyConfidenceInterval95Percent(n) = Return_Fitting(i) - Return_Sy(i) * 2.228
'Return_UyConfidenceInterval95Percent(n) = Return_Fitting(i) + Return_Sy(i) * 2.228
'Return_LyConfidenceInterval99Percent(n) = Return_Fitting(i) - Return_Sy(i) * 3.169
'Return_UyConfidenceInterval99Percent(n) = Return_Fitting(i) + Return_Sy(i) * 3.169
Next
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'相关系数的 T 显著性检验
```

```
'相关系数 r 是样本相关系数，  它是双变量正态总体中的总体相关系数ρ的估计值。
'样本相关系数 r 是否来自ρ≠0 的总体，还须对样本相关系数 r  进行显著性检验。

'两种分析所进行的显著性检验都是解决 y 与 x 间是否存在直线关系。因而二者的检验是等价的。
'即相关系数显著，  回归系数亦显著；相关系数不显著，回归系数也必然不显著。
'****************************************************
'所以方程的显著性检验和 r 的显著检验是数值相等的'****
'****************************************************
Dim Return_Sr As Double, Return_rT As Double
Return_Sr = Sqr((1 - Return_r ^ 2) / (Num - 2))
Return_rT = Return_r / Return_Sr
'相关系数的 F 显著性检验
Dim Return_rF As Double
Return_rF = Return_r ^ 2 / ((1 - Return_r ^ 2) / (Num - 2))
'*************************
Dim Return_rTReport As String
If (Return_rT > Return_T001) Then
Return_rTReport = "极显著的 r"
ElseIf (Return_T001 > Return_rT And Return_rT > Return_T005) Then
Return_rTReport = "当 AERF=0.05 显著的 r，AERF=0.01 无显著 r"
ElseIf (Return_T005 > Return_rT) Then
Return_rTReport = "无显著 r"
End If
'*************************
Dim Return_rF001 As Double, Return_rF005 As Double
'Return_rF001 = obj.PFaN1N2X2(0.01, 0.00001, 1, Num - 2) '自由度带考证 2016-6-20
'Return_rF005 = obj.PFaN1N2X2(0.05, 0.00001, 1, Num - 2)
Rem 16-9-7
Return_rF001 = obj.PFaN1N2X2(0.01, 10000000, 1, Return_df_r) ' 0.01 用千万
Return_rF005 = obj.PFaN1N2X2(0.05, 1000000, 1, Return_df_r)      ' 0.05 用百万
'************************************
Dim Return_rFReport As String
If (Return_rF > Return_rF001) Then
Return_rFReport = "极显著的 r"
ElseIf (Return_rF001 > Return_rF And Return_rF > Return_rF005) Then
Return_rFReport = "当 AERF=0.05 有显著 r，AERF=0.01 无显著 r"
ElseIf (Return_rF005 > Return_rF) Then
Return_rFReport = "无显著的 r"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis = LinearRegressionAnalysis & "(DataY-拟合 Y)^2 总和=                    " & Chr(9) & Part6 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "DataX 总和=
```

```
" & Chr(9) & Part2 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "DataY 总和=
" & Chr(9) & Part4 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "DataX^2 总和=
" & Chr(9) & Part1 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "DataY^2 总和=
" & Chr(9) & Part5 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "DataX*DataY 总和=
" & Chr(9) & Part3 & Chr(13) & Chr(10)

LinearRegressionAnalysis = LinearRegressionAnalysis & "DataX 平均值=
" & Chr(9) & MeanX & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "DataY 平均值=
" & Chr(9) & MeanY & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "DataX 离均差平方和 SSx=                "
& Chr(9) & Return_SSx & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "DataY 离均差平方和 SSy=                "
& Chr(9) & Return_SSy & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "乘积和 SPxy=
" & Chr(9) & Return_SPxy & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "离回归标准误 Syx=
" & Chr(9) & Return_Syx & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "回归平方和 SSR=
" & Chr(9) & Part7 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "离回归平方和或剩余平方和 SSr=" &
Chr(9) & Return_SS_r & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "总自由度 dfy=
" & Chr(9) & Return_dfy & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "离回归自由度 dfr=
" & Chr(9) & Return_df_r & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "回归自由度 dfR=
" & Chr(9) & Return_dfR & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "a=
" & Chr(9) & Return_A & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "b=
" & Chr(9) & Return_B & Chr(13) & Chr(10)
If Return_B >= 0 Then
LinearRegressionAnalysis = LinearRegressionAnalysis & "线性回归函数 y=
" _
& Chr(9) & Return_A & "+" & Return_B & "x" & Chr(13) & Chr(10)
Else
Rem 这里只是符号的问题 2016-6-20 待进一步验证
LinearRegressionAnalysis = LinearRegressionAnalysis & "线性回归函数 y=
" _
```

```
& Chr(9) & Return_A & Return_B & "x" & Chr(13) & Chr(10)
End If
LinearRegressionAnalysis = LinearRegressionAnalysis & "r=
" & Chr(9) & Return_r & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "r^2=
" & Chr(9) & (Return_r ^ 2) & Chr(13) & Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis = LinearRegressionAnalysis & "F 检验
**********" & Chr(13) &
Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "F=
" & Return_F & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "F001（不准确值）=
" & Return_F001 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "F005（不准确值）=
" & Return_F005 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "MSR=
" & Return_MSR & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "MSr=
" & Return_MS_r & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "Return_FReport（显著性检验报告）=" &
Return_FReport & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "F 检验结束
**********" & Chr(13) &
Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis = LinearRegressionAnalysis & "T 检验
**********" & Chr(13) &
Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "T=
" & Return_T & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "T0.01（错误值待改正）=
" & Return_T001 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "T0.05（错误值待改正）=
" & Return_T005 & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "Return_TReport（显著性检验报告）=" &
Return_TReport & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "T 检验结束
**********" & Chr(13) &
Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis = LinearRegressionAnalysis & "直线回归的区间估计
**********" & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "总体回归截距 a 的置信区间" & Chr(13)
```

```
& Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "Sa=              " & Return_Sa & Chr(13) &
Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "α95%=[" &
Return_LαConfidenceInterval95Percent & " , " & Return_UαConfidenceInterval95Percent & "]" &
Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "α99%=[" &
Return_LαConfidenceInterval99Percent & " , " & Return_UαConfidenceInterval99Percent & "]" &
Chr(13) & Chr(10)
'*******************
LinearRegressionAnalysis = LinearRegressionAnalysis & "总体回归系数β的置信区间" & Chr(13)
& Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "β95%=[" &
Return_LβConfidenceInterval95Percent & " , " & Return_UβConfidenceInterval95Percent & "]" &
Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "β99%=[" &
Return_LβConfidenceInterval99Percent & " , " & Return_UβConfidenceInterval99Percent & "]" &
Chr(13) & Chr(10)
'*******************
'总体平均数α+βx 的置信区间
LinearRegressionAnalysis = LinearRegressionAnalysis & "总体平均数α+βx 的置信区间" & Chr(13)
& Chr(10)
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis = LinearRegressionAnalysis & "Sy^(" & "i" & ")=" & Return_SFittingy(i) &
Chr(13) & Chr(10)
Next
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis = LinearRegressionAnalysis & "X=" & DataX(i) & "Y 总体平均数  95%=["
& Return_LPopulationMeanConfidenceInterval95Percent(i) & " , " &
Return_UPopulationMeanConfidenceInterval95Percent(i) & "]" & Chr(13) & Chr(10)
Next
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis = LinearRegressionAnalysis & "X=" & DataX(i) & "Y 总体平均数  99%=["
& Return_LPopulationMeanConfidenceInterval99Percent(i) & " , " &
Return_UPopulationMeanConfidenceInterval99Percent(i) & "]" & Chr(13) & Chr(10)
Next
'**********************
'单个 y 值的置信区间
LinearRegressionAnalysis = LinearRegressionAnalysis & "单个 y 值的置信区间" & Chr(13) &
Chr(10)
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis = LinearRegressionAnalysis & "Sy(" & "i" & ")=" & Return_Sy(i) & Chr(13)
& Chr(10)
Next
```

```
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis = LinearRegressionAnalysis & "X=" & DataX(i) & _
"Y 总体平均数  95%=[" & Return_LyConfidenceInterval95Percent(i) & " , " &
Return_UyConfidenceInterval95Percent(i) & "]" & Chr(13) & Chr(10)
Next
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis = LinearRegressionAnalysis & "X=" & DataX(i) & _
"Y 总体平均数  99%=[" & Return_LyConfidenceInterval99Percent(i) & " , " &
Return_UyConfidenceInterval99Percent(i) & "]" & Chr(13) & Chr(10)
Next
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis = LinearRegressionAnalysis & "相关系数的 T 显著性检验" & Chr(13) &
Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "Sr=
" & Return_Sr & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "相关系数 T=                    " &
Return_rT & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & Return_rTReport & Chr(13) & Chr(10)
'*************************
LinearRegressionAnalysis = LinearRegressionAnalysis & "相关系数的 F 显著性检验" & Chr(13) &
Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & "相关系数 F=                    " &
Return_rF & Chr(13) & Chr(10)
LinearRegressionAnalysis = LinearRegressionAnalysis & Return_rFReport & Chr(13) & Chr(10)
End Function
```

'LinearRegressionAnalysis_Equation 这个和 LinearRegressionAnalysis 运行结果一样，多返回 2 个系数

Rem 这个和 LinearRegressionAnalysis 运行结果一样，只是返回 2 个系数 16-9-7
Rem 田间统计学单列子通过 16-9-7

```
Public Function LinearRegressionAnalysis_Equation(ByRef DataX() As Double, ByRef DataY() As
Double, _
ByRef Return_A As Double, ByRef Return_B As Double) As String
Dim MeanX As Double
Dim MeanY As Double
Dim Sum1 As Double
Dim Sum2 As Double
Dim Part1 As Double, Part2 As Double, Part3 As Double, Part4 As Double, Part5 As Double
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Num As Long, Num1 As Long
Num1 = UBound(DataY) - LBound(DataY) + 1
```

```
Num = UBound(DataX) - LBound(DataX) + 1
If (Num <> Num1) Then
MsgBox "数据有问题，退出程序"
LinearRegressionAnalysis_Equation = "None"
Exit Function
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MeanX = ArithmeticMean(DataX(), Sum1)
MeanY = ArithmeticMean(DataY(), Sum2)
Dim i As Long
For i = LBound(DataX) To UBound(DataX)
Part1 = Part1 + DataX(i) ^ 2
Part2 = Part2 + DataX(i)
Next
Dim Return_SSx As Double
Return_SSx = Part1 - Part2 ^ 2 / Num
For i = LBound(DataX) To UBound(DataX)
Part3 = Part3 + DataX(i) * DataY(i)
Part4 = Part4 + DataY(i)
Next
Dim Return_SPxy As Double
Return_SPxy = Part3 - Part2 * Part4 / Num
For i = LBound(DataX) To UBound(DataX)
Part5 = Part5 + DataY(i) ^ 2
Next

'y 的总变异程度，称为 y 的总平方和
Dim Return_SSy As Double
'Dim Return_b As Double
'Dim Return_a As Double

Return_SSy = Part5 - Part4 ^ 2 / Num
Return_B = Return_SPxy / Return_SSx
Return_A = MeanY - Return_B * MeanX
Dim Return_r As Double
Return_r = Return_SPxy / Sqr(Return_SSx * Return_SSy)

Dim Part6 As Double, Part7 As Double, Part8 As Double
Dim n As Long
Dim Return_Fitting() As Double
For i = LBound(DataX) To UBound(DataX)
n = n + 1
ReDim Preserve Return_Fitting(n)
Return_Fitting(n) = Return_A + Return_B * DataX(i)
```

```
Part6 = Part6 + (DataY(i) - Return_Fitting(n)) ^ 2
Part7 = Part7 + (Return_Fitting(n) - MeanY) ^ 2
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "拟合数据" & n & "="
& Return_Fitting(n) & Chr(13) & Chr(10)
Next
n = 0
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'直线回归的偏离度估计
'离回归均方的平方根叫离回归标准误，记为 Syx
Dim Return_Syx As Double
Return_Syx = Sqr(Part6 / (Num - 2))
'y 与 x 间存在直线关系所引起的 y 的变异程度，称为回归平方和
Dim Return_SSr    As Double
Dim Return_SS_r As Double
Dim Return_dfR As Long
Dim Return_dfy As Long
Dim Return_df_r As Long
Rem  没有验证
Return_SSr = Part7
Return_SS_r = Return_SSy - Return_SSr
Return_dfR = 1
Return_dfy = Num - 1
Return_df_r = Num - 2
'F 检验~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Return_F As Double
Dim Return_MSR As Double
Dim Return_MS_r As Double
Rem  出问题下面语句除数为 0 16-8-15 ! ! ! ! ! ! ! ! !
If Return_SS_r <> 0 Then
Return_F = Return_SSr / (Return_SS_r / (Num - 2))
Else
Return_F = 9999999999#
MsgBox "完全重合"
End If
Return_MSR = Return_SSr / Return_dfR
Return_MS_r = Return_SS_r / Return_df_r
Dim Return_F001 As Double, Return_F005 As Double
Dim obj As New F001
Rem  这个精度有点小，但是在增加运算量将会变慢，还要找加速方法提高精度 2016-6-20
'*********************************************************************
'Return_F001 = obj.PFaN1N2X2(0.01, 0.01, Return_dfR, Return_df_r)
'Return_F005 = obj.PFaN1N2X2(0.05, 0.01, Return_dfR, Return_df_r)
Return_F001 = obj.PFaN1N2X2(0.01, 10000000, Return_dfR, Return_df_r) ' 0.01 用千万  16-8-28
Return_F005 = obj.PFaN1N2X2(0.05, 1000000, Return_dfR, Return_df_r)     ' 0.05 用百万
```

```
'*****************************************************************
Dim Return_FReport As String
If (Return_F > Return_F001) Then
Return_FReport = "显著的直线关系"
ElseIf (Return_F001 > Return_F And Return_F > Return_F005) Then
Return_FReport = "当 AERF=0.05 有显著的直线关系，AERF=0.01 无显著直线关系"
ElseIf (Return_F005 > Return_F) Then
Return_FReport = "无显著的直线关系"
End If
'T 检验~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Return_Sb As Double, Return_T As Double, u1 As Double, u2 As Double, Shape() As Double
Dim Return_T001 As Double, Return_T005 As Double
Dim obj1 As New T001
Return_Sb = Return_Syx / Sqr(Return_SSx)
If Return_Sb <> 0 Then
Return_T = Return_B / Return_Sb
Else
Return_T = 99999999999#
MsgBox "完全重合"
End If
'Return_T001 = obj1.t(0.01, Return_df_r)
'Return_T005 = obj1.t(0.05, Return_df_r)
Return_T001 = obj1.t(0.005, Return_df_r) ' 16-9-7 参照电子书修改.田间统计已经验证
Return_T005 = obj1.t(0.025, Return_df_r)

'Return_T001 = obj1.t_DistributionRectangle_μα(Num - 2, 0.01, 100, 20, u1, u2, Shape()) '数据问
题较大
'Return_T005 = obj1.t_DistributionRectangle_μα(Num - 2, 0.05, 100, 20, u1, u2, Shape())
'*****************************************************************
Rem 16-9-7  参照田间统计，T 取绝度值
Dim Return_T1    As Double
Return_T1 = Abs(Return_T)
'*****************************************************************
Dim Return_TReport As String
If (Return_T1 > Return_T001) Then
Return_TReport = "显著的直线关系"
ElseIf (Return_T001 > Return_T1 And Return_T1 > Return_T005) Then
Return_TReport = "当 AERF=0.05 有显著的直线关系，AERF=0.01 无显著直线关系"
ElseIf (Return_T005 > Return_T1) Then
Return_TReport = "无显著的直线关系"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'直线回归的区间估计  2016-6-20  下面的程序未写入备用程序中
Rem α是当 x=0 时 y^的值，即直线在 y 轴上的截距，叫回归截距。
```

```
Dim Return_Sa As Double
Dim Return_LαConfidenceInterval95Percent    As Double,
Return_UαConfidenceInterval95Percent As Double
Dim Return_LαConfidenceInterval99Percent    As Double,
Return_UαConfidenceInterval99Percent As Double
Return_Sa = Return_Syx * Sqr(1 / Num + MeanX ^ 2 / Return_SSx)
'Return_LαConfidenceInterval95Percent = Return_a - Return_Sa * 2.228
'Return_UαConfidenceInterval95Percent = Return_a + Return_Sa * 2.228
'Return_LαConfidenceInterval99Percent = Return_a - Return_Sa * 3.169
'Return_UαConfidenceInterval99Percent = Return_a + Return_Sa * 3.169
Return_LαConfidenceInterval95Percent = Return_A - Return_Sa * Return_T005
Return_UαConfidenceInterval95Percent = Return_A + Return_Sa * Return_T005
Return_LαConfidenceInterval99Percent = Return_A - Return_Sa * Return_T001
Return_UαConfidenceInterval99Percent = Return_A + Return_Sa * Return_T001
'******************
'总体回归系数β的置信区间
Dim Return_LβConfidenceInterval95Percent    As Double,
Return_UβConfidenceInterval95Percent As Double
Dim Return_LβConfidenceInterval99Percent    As Double,
Return_UβConfidenceInterval99Percent As Double
Return_LβConfidenceInterval95Percent = Return_B - Return_Sb * Return_T005
Return_UβConfidenceInterval95Percent = Return_B + Return_Sb * Return_T005
Return_LβConfidenceInterval99Percent = Return_B - Return_Sb * Return_T001
Return_UβConfidenceInterval99Percent = Return_B + Return_Sb * Return_T001
'Return_LβConfidenceInterval95Percent = Return_b - Return_Sb * 2.228
'Return_UβConfidenceInterval95Percent = Return_b + Return_Sb * 2.228
'Return_LβConfidenceInterval99Percent = Return_b - Return_Sb * 3.169
'Return_UβConfidenceInterval99Percent = Return_b + Return_Sb * 3.169
'******************
'总体平均数α+βx 的置信区间
Dim Return_LPopulationMeanConfidenceInterval95Percent() As Double
Dim Return_UPopulationMeanConfidenceInterval95Percent() As Double
Dim Return_LPopulationMeanConfidenceInterval99Percent() As Double
Dim Return_UPopulationMeanConfidenceInterval99Percent() As Double
Dim Return_SFittingy() As Double
For i = LBound(DataX) To UBound(DataX)
n = n + 1
ReDim Preserve Return_LPopulationMeanConfidenceInterval95Percent(n)
ReDim Preserve Return_UPopulationMeanConfidenceInterval95Percent(n)
ReDim Preserve Return_LPopulationMeanConfidenceInterval99Percent(n)
ReDim Preserve Return_UPopulationMeanConfidenceInterval99Percent(n)
ReDim Preserve Return_SFittingy(n)
Return_SFittingy(i) = Return_Syx * Sqr(1 / Num + (DataX(i) - MeanX) ^ 2 / Return_SSx)
Return_LPopulationMeanConfidenceInterval95Percent(n) = Return_Fitting(i) - Return_SFittingy(i)
```

```
* Return_T005
Return_UPopulationMeanConfidenceInterval95Percent(n) = Return_Fitting(i) + Return_SFittingy(i)
* Return_T005
Return_LPopulationMeanConfidenceInterval99Percent(n) = Return_Fitting(i) - Return_SFittingy(i)
* Return_T001
Return_UPopulationMeanConfidenceInterval99Percent(n) = Return_Fitting(i) + Return_SFittingy(i)
* Return_T001
'Return_LPopulationMeanConfidenceInterval95Percent(n) = Return_Fitting(i) - Return_SFittingy(i)
* 2.228
'Return_UPopulationMeanConfidenceInterval95Percent(n) = Return_Fitting(i) +
Return_SFittingy(i) * 2.228
'Return_LPopulationMeanConfidenceInterval99Percent(n) = Return_Fitting(i) - Return_SFittingy(i)
* 3.169
'Return_UPopulationMeanConfidenceInterval99Percent(n) = Return_Fitting(i) +
Return_SFittingy(i) * 3.169
Next
n = 0
'*******************
'单个 y 值的置信区间
Dim Return_LyConfidenceInterval95Percent() As Double
Dim Return_UyConfidenceInterval95Percent() As Double
Dim Return_LyConfidenceInterval99Percent() As Double
Dim Return_UyConfidenceInterval99Percent() As Double
Dim Return_Sy() As Double
For i = LBound(DataX) To UBound(DataX)
n = n + 1
ReDim Preserve Return_LyConfidenceInterval95Percent(n)
ReDim Preserve Return_UyConfidenceInterval95Percent(n)
ReDim Preserve Return_LyConfidenceInterval99Percent(n)
ReDim Preserve Return_UyConfidenceInterval99Percent(n)
ReDim Preserve Return_Sy(n)
Return_Sy(i) = Return_Syx * Sqr(1 + 1 / Num + (DataX(i) - MeanX) ^ 2 / Return_SSx)
Return_LyConfidenceInterval95Percent(n) = Return_Fitting(i) - Return_Sy(i) * Return_T005
Return_UyConfidenceInterval95Percent(n) = Return_Fitting(i) + Return_Sy(i) * Return_T005
Return_LyConfidenceInterval99Percent(n) = Return_Fitting(i) - Return_Sy(i) * Return_T001
Return_UyConfidenceInterval99Percent(n) = Return_Fitting(i) + Return_Sy(i) * Return_T001
'Return_LyConfidenceInterval95Percent(n) = Return_Fitting(i) - Return_Sy(i) * 2.228
'Return_UyConfidenceInterval95Percent(n) = Return_Fitting(i) + Return_Sy(i) * 2.228
'Return_LyConfidenceInterval99Percent(n) = Return_Fitting(i) - Return_Sy(i) * 3.169
'Return_UyConfidenceInterval99Percent(n) = Return_Fitting(i) + Return_Sy(i) * 3.169
Next
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'相关系数的 T 显著性检验
'相关系数 r 是样本相关系数， 它是双变量正态总体中的总体相关系数ρ的估计值。
```

```
'样本相关系数 r 是否来自ρ≠0 的总体，还须对样本相关系数 r 进行显著性检验。

'两种分析所进行的显著性检验都是解决 y 与 x 间是否存在直线关系。因而二者的检验是等价的。
'即相关系数显著， 回归系数亦显著；相关系数不显著，回归系数也必然不显著。
'*******************************************************
'所以方程的显著性检验和 r 的显著检验是数值相等的'****
'*******************************************************
Dim Return_Sr As Double, Return_rT As Double
Return_Sr = Sqr((1 - Return_r ^ 2) / (Num - 2))
If Return_Sr <> 0 Then
Return_rT = Return_r / Return_Sr
Else
Return_rT = 9999999999#
End If
'相关系数的 F 显著性检验
Dim Return_rF As Double
If (1 - Return_r ^ 2) <> 0 Then
Return_rF = Return_r ^ 2 / ((1 - Return_r ^ 2) / (Num - 2))
Else
Return_rF = 9999999999#
End If
'*************************
Dim Return_rTReport As String
If (Return_rT > Return_T001) Then
Return_rTReport = "极显著的 r"
ElseIf (Return_T001 > Return_rT And Return_rT > Return_T005) Then
Return_rTReport = "当 AERF=0.05 显著的 r，AERF=0.01 无显著 r"
ElseIf (Return_T005 > Return_rT) Then
Return_rTReport = "无显著 r"
End If
'*************************
Dim Return_rF001 As Double, Return_rF005 As Double
'Return_rF001 = obj.PFaN1N2X2(0.01, 0.001, 1, Num - 2) '自由度带考证 2016-6-20
'Return_rF005 = obj.PFaN1N2X2(0.05, 0.001, 1, Num - 2)
Dim Return_rFReport As String
If (Return_rF > Return_rF001) Then
Return_rFReport = "极显著的 r"
ElseIf (Return_rF001 > Return_rF And Return_rF > Return_rF005) Then
Return_rFReport = "当 AERF=0.05 有显著 r，AERF=0.01 无显著 r"
ElseIf (Return_rF005 > Return_rF) Then
Return_rFReport = "无显著的 r"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "(DataY-拟合 Y)^2 总和=                    " & Chr(9) & Part6 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataX 总和=
" & Chr(9) & Part2 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataY 总和=
" & Chr(9) & Part4 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataX^2 总和=
" & Chr(9) & Part1 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataY^2 总和=
" & Chr(9) & Part5 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataX*DataY 总和=
" & Chr(9) & Part3 & Chr(13) & Chr(10)

LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataX 平均值=
" & Chr(9) & MeanX & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataY 平均值=
" & Chr(9) & MeanY & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataX 离均差平方和 SSx=                    " & Chr(9) & Return_SSx & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "DataY 离均差平方和 SSy=                    " & Chr(9) & Return_SSy & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "乘积和 SPxy=
" & Chr(9) & Return_SPxy & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "离回归标准误 Syx=
" & Chr(9) & Return_Syx & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "回归平方和 SSR=
" & Chr(9) & Part7 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "离回归平方和或剩余平方和 SSr=" & Chr(9) & Return_SS_r & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "总自由度 dfy=
" & Chr(9) & Return_dfy & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "离回归自由度 dfr=
" & Chr(9) & Return_df_r & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "回归自由度 dfR=
" & Chr(9) & Return_dfR & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "a=
" & Chr(9) & Return_A & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "b=
" & Chr(9) & Return_B & Chr(13) & Chr(10)
If Return_B >= 0 Then
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "线性回归函数 y=
" _
& Chr(9) & Return_A & "+" & Return_B & "x" & Chr(13) & Chr(10)
Else
```

```
Rem 这里只是符号的问题 2016-6-20 待进一步验证
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "线性回归函数 y=
" _
& Chr(9) & Return_A & Return_B & "x" & Chr(13) & Chr(10)
End If
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "r=
" & Chr(9) & Return_r & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "r^2=
" & Chr(9) & (Return_r ^ 2) & Chr(13) & Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "F 检验
**********************************************************************" & Chr(13) &
Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "F=
" & Return_F & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "F001（不准确值）=
" & Return_F001 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "F005（不准确值）=
" & Return_F005 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "MSR=
" & Return_MSR & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "MSr=
" & Return_MS_r & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "MSr=
" & Return_MS_r & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "Return_FReport（显
著性检验报告）=" & Return_FReport & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "F 检验结束
**********************************************************************" & Chr(13) &
Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "T 检验
**********************************************************************" & Chr(13) &
Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "T=
" & Return_T & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "T0.01（错误值待改
正）=                              " & Return_T001 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "T0.05（错误值待改
正）=                              " & Return_T005 & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "Return_TReport（显
著性检验报告）=" & Return_TReport & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "T 检验结束
**********************************************************************" & Chr(13) &
```

```
Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "直线回归的区间估计********************" & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "总体回归截距 a 的置信区间" & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "Sa=            " & Return_Sa & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "α95%=[" & Return_LαConfidenceInterval95Percent & " , " & Return_UαConfidenceInterval95Percent & "]" & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "α99%=[" & Return_LαConfidenceInterval99Percent & " , " & Return_UαConfidenceInterval99Percent & "]" & Chr(13) & Chr(10)
'******************
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "总体回归系数β的置信区间" & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "β95%=[" & Return_LβConfidenceInterval95Percent & " , " & Return_UβConfidenceInterval95Percent & "]" & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "β99%=[" & Return_LβConfidenceInterval99Percent & " , " & Return_UβConfidenceInterval99Percent & "]" & Chr(13) & Chr(10)
'******************
'总体平均数α+βx 的置信区间
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "总体平均数α+βx 的置信区间" & Chr(13) & Chr(10)
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "Sy^(" & "i" & ")=" & Return_SFittingy(i) & Chr(13) & Chr(10)
Next
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "X=" & DataX(i) & "Y总体平均数  95%=[" & Return_LPopulationMeanConfidenceInterval95Percent(i) & " , " & Return_UPopulationMeanConfidenceInterval95Percent(i) & "]" & Chr(13) & Chr(10)
Next
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "X=" & DataX(i) & "Y总体平均数  99%=[" & Return_LPopulationMeanConfidenceInterval99Percent(i) & " , " & Return_UPopulationMeanConfidenceInterval99Percent(i) & "]" & Chr(13) & Chr(10)
Next
'**********************
'单个 y 值的置信区间
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "单个 y 值的置信区
```

```
间" & Chr(13) & Chr(10)
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "Sy(" & "i" & ")=" &
Return_Sy(i) & Chr(13) & Chr(10)
Next
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "X=" & DataX(i) & _
"Y 总体平均数  95%=[" & Return_LyConfidenceInterval95Percent(i) & " , " &
Return_UyConfidenceInterval95Percent(i) & "]" & Chr(13) & Chr(10)
Next
For i = LBound(DataX) To UBound(DataX)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "X=" & DataX(i) & _
"Y 总体平均数  99%=[" & Return_LyConfidenceInterval99Percent(i) & " , " &
Return_UyConfidenceInterval99Percent(i) & "]" & Chr(13) & Chr(10)
Next
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "相关系数的 T 显著
性检验" & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "Sr=
" & Return_Sr & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "相关系数 T=
" & Return_rT & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & Return_rTReport &
Chr(13) & Chr(10)
'*************************
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "相关系数的 F 显著
性检验" & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & "相关系数 F=
" & Return_rF & Chr(13) & Chr(10)
LinearRegressionAnalysis_Equation = LinearRegressionAnalysis_Equation & Return_rFReport &
Chr(13) & Chr(10)
End Function
```

'TwoElementLinearAnalysis_Equation 二元线性分析函数式

Rem 2 元线性分析函数式

```
Public Function TwoElementLinearAnalysis_Equation(ByRef DataX() As Double, ByRef DataY() As
Double, _
ByRef Return_2PowerCoefficient As Double, ByRef Return_1PowerCoefficient As Double, _
ByRef Return_ConstantCoefficient As Double) As String
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  得到行列式
Dim DataMatrix() As Double
```

```
Dim DataMatriy() As Double
Dim Num As Long
Dim i As Long, l As Long
Num = UBound(DataX) - LBound(DataX) + 1
ReDim DataMatrix(1 To Num, 1 To 3) As Double
For l = 1 To 3
For i = 1 To Num
If l = 1 Then
DataMatrix(i, l) = 1
ElseIf l = 2 Then
DataMatrix(i, l) = DataX(i)
ElseIf l = 3 Then
DataMatrix(i, l) = DataX(i) ^ 2
End If
Next i
Next l
'************************************************
Num = UBound(DataY) - LBound(DataY) + 1
ReDim DataMatriy(1 To Num, 1) As Double
For i = 1 To Num
DataMatriy(i, 1) = DataY(i)
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim obj As New Matrix001
Dim DataMatrixT() As Double
obj.TransposedMatrix DataMatrix, DataMatrixT
Dim R() As Double
Dim R1() As Double
Dim R2() As Double
obj.MatrixMultiplication DataMatrixT, DataMatrix, R
obj.InverseMatrix R, R1
obj.MatrixMultiplication DataMatrixT, DataMatriy, R2
Dim b() As Double
obj.MatrixMultiplication R1, R2, b
Return_ConstantCoefficient = b(1, 1)
Return_1PowerCoefficient = b(2, 1)
Return_2PowerCoefficient = b(3, 1)
End Function
```

'OneVeriblePolyNomialRegression 多元线性回归分析函数式（只能分析 1 元 1 次到 7 次）

Rem 多元线性回归分析函数式，只能分析 1 元 1 次到 7 次。（16-8-19）补充，为什么书上吧 2 元 2 次方程放到，多元线性。是因为 2 元 2 次可以看成是 3 元一次方程
y=b0+b1x+b2x^2+E ==》 y= b0+b1x1+b2x2+E
Rem Return_Coefficient1D() 返回的是常数项，第一个是常数，第二个是 X1 的常数，第三个是 X2 的常数，以此类推
Rem Degree 是次数 次数=2 的时候 DATAX 必须是 2 维数组 16-9-6 但是现在又发现 Degree 在这里是次数
Rem 测试 2 元，3 元是正确的（16-8-19）
Rem DATAX 数据格式是 DATAX DATAY 数据只能是一维，也就是只能两条数据 16-9-7 所以增加判别
Rem polynomial regression

```
Public Function OneVeriblePolyNomialRegression(ByRef DataX() As Double, ByRef DataY() As
Double, _
ByVal Degree As Long, ByRef Return_Coefficient1D() As Double) As String
'Public Function MultiDegreeLinearAnalysis_Equation(ByRef DataX() As Double, ByRef DataY() As
Double, _
'ByVal Degree As Long, ByRef Return_Coefficient1D() As Double) As String
If (Degree >= 7 Or Degree < 2) Then
MsgBox "暂时不支持"
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 得到行列式
Dim DataMatrix() As Double
Dim DataMatriy() As Double
Dim Num As Long
Dim i As Long, l As Long
Num = UBound(DataX) - LBound(DataX) + 1
ReDim DataMatrix(1 To Num, 1 To (Degree + 1)) As Double
For l = 1 To (Degree + 1)
For i = 1 To Num
DataMatrix(i, l) = DataX(i) ^ (l - 1)
'Debug.Print "DataMatrix(" & i & "," & l & ")=" & DataX(i) ^ (l - 1)
Next i
Next l
'**************************************************
Num = UBound(DataY) - LBound(DataY) + 1
ReDim DataMatriy(1 To Num, 1) As Double
For i = 1 To Num
DataMatriy(i, 1) = DataY(i)
```

```
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim obj As New Matrix001
Dim DataMatrixT() As Double
obj.TransposedMatrix DataMatrix, DataMatrixT '求 X 得转置矩阵 X^T
Dim R() As Double
Dim R1() As Double
Dim R2() As Double
obj.MatrixMultiplication DataMatrixT, DataMatrix, R '求 X^T * X
obj.InverseMatrix R, R1 '求(X^T * X)^(-1)
obj.MatrixMultiplication DataMatrixT, DataMatriy, R2 '求 X^T * Y

Dim b() As Double
obj.MatrixMultiplication R1, R2, b '求(X^T * X)^(-1) * (X^T * Y) =  系数矩阵 b
For i = LBound(b, 1) To UBound(b, 1)
ReDim Preserve Return_Coefficient1D(i)
Return_Coefficient1D(i) = b(i, 1)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + CStr(i - 1) + "次项系数="
+ CStr(Return_Coefficient1D(i)) + Chr(13) + Chr(10)
Next

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~



Dim PartX() As Double
Dim PartXSquare() As Double
ReDim PartX(1 To Degree)
ReDim PartXSquare(1 To Degree)
For i = 1 To Degree
For l = 1 To Num
PartX(i) = PartX(i) + DataMatrix(l, i + 1)
'PartX(i) = PartX(i) + DataX(i, l)
PartXSquare(i) = PartXSquare(i) + DataMatrix(l, i + 1)
'PartXSquare(i) = PartXSquare(i) + DataX(i, l) ^ 2
Next l
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "X" + CStr(i) + "项累加=
" + Chr(9) + CStr(PartX(i)) + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "X" + CStr(i) + "项平均值
=          " + Chr(9) + CStr(PartX(i) / Num) + Chr(13) + Chr(10)
```

```
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "X" + CStr(i) + "项平方累加=          " + Chr(9) + CStr(PartXSquare(i)) + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "X" + CStr(i) + "项平方累加平均值=" + Chr(9) + CStr(PartXSquare(i) / Num) + Chr(13) + Chr(10)
Next i
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************
Rem 前题是 Y 项和 X 项数量是相同的
Dim PartY As Double
Dim PartYSquare As Double
For l = 1 To Num
PartY = PartY + DataY(l)
PartYSquare = PartYSquare + DataY(l) ^ 2
Next l
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "Y 项累加             =" + CStr(PartY) + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "Y 项平均值     =" + CStr(PartY / Num) + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "Y 项平方累加=" + CStr(PartYSquare) + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'***********************************
Rem 求 SSi 的值
Dim SS() As Double
ReDim SS(1 To Degree) As Double
For i = 1 To Degree
For l = 1 To Num
SS(i) = SS(i) + (DataMatrix(l, i + 1) - PartX(i) / Num) ^ 2
'SS(i) = SS(i) + (DataX(i, l) - PartX(i) / Num) ^ 2
Next l
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "SS" + CStr(i) + "=" + CStr(SS(i)) + Chr(13) + Chr(10)
Next i
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'************************************
Rem 求 SSy 的值
Dim SSy As Double
For i = 1 To Num
SSy = SSy + (DataY(i) - PartY / Num) ^ 2
Next i
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "y 变量的总平方和 SSy"
```

```
+ "=" + CStr(SSy) + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************************************************************
Rem  求 SPik 的值
Rem  田间统计分析书上测试已经通过（16-8-24）
Dim n As Long, m As Long
Dim SP() As Double
ReDim SP(1 To Degree, 1 To Degree) As Double
ReDim Return_SP(1 To Degree, 1 To Degree) As Double
For i = 1 To Degree
For l = 1 To Degree
      '*********************************
    Rem 16-9-5
    'For m = 1 To Num
    'Return_SP(i, l) = Return_SP(i, l) + (DataX(m, i) - PartX(i) / Num) * (DataX(m, l) - PartX(l) / Num)
    'Next m
    '*********************************
    If (i <> l) Then
    If (i < l) Then

        For n = 1 To Num
        SP(i, l) = SP(i, l) + (DataMatrix(n, i + 1) - PartX(i) / Num) * (DataMatrix(n, l + 1) - PartX(l) /
Num)
        'SP(i, l) = SP(i, l) + (DataX(n, i) - PartX(i) / Num) * (DataX(n, l) - PartX(l) / Num)
        Next n
        OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "SP(" + CStr(i) + ","
+ CStr(l) + ")=" + CStr(SP(i, l)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i, l))
    End If
    End If
Next l
Next i
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)

'*************************************************************************
    Rem  求 SPi0 的值
  Dim SP0() As Double
  ReDim SP0(1 To Degree) As Double
 For i = 1 To Degree
 For n = 1 To Num
          SP0(i) = SP0(i) + (DataMatrix(n, i + 1) - PartX(i) / Num) * (DataY(n) - PartY / Num)
        'SP0(i) = SP0(i) + (DataX(n, i) - PartX(i) / Num) * (DataY(n) - PartY / Num)
```

```
        Next n
        OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "SP0(" + CStr(i) +
")=" + CStr(SP0(i)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(L) + ")=" + CStr(SP(i, L))
    Next i
    OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
    '
*************************************************************************
    Rem 离回归标准误
    Dim Se As Double
    Dim Uy As Double
    Dim Qy As Double

    For i = 1 To Degree
    Uy = Uy + Return_Coefficient1D(i + 1) * SP0(i)
    Next i

    Qy = SSy - Uy
    OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "~~~~~~~~~~~~多元
线性显著性检验~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim SSr As Double
Dim SS_r As Double
Dim DFy As Double
Dim DFR As Double
Dim DF_r As Double
Dim MSR As Double
Dim MS_r As Double
SSr = Uy
SS_r = Qy
DFy = Num - 1
DFR = Degree
DF_r = Num - Degree - 1
MSR = SSr / DFR
MS_r = SS_r / DF_r
Dim F As Double

    If SS_r = 0 Then
    MsgBox "完全重合"
    F = 999999999
    Else
    F = MSR / MS_r ' 16-9-6 更改到 IF 内
    End If
```

```
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "变异原因              DF
SS                              MS                          F" + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "多元回归              " +
CStr(DFR) + "          " + Chr(9) + CStr(SSr) + "          " + Chr(9) + CStr(MSR) + "          " + Chr(9) +
CStr(F) + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "离回归                  " +
CStr(DF_r) + "          " + Chr(9) + CStr(SS_r) + "          " + Chr(9) + CStr(MS_r) + "          " + Chr(13) +
Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "总和                    " +
CStr(DFy) + "          " + Chr(9) + CStr(SSy) + Chr(13) + Chr(10)
'*********************************************************************

Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj2 As New F001
Return_F001 = obj2.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj2.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万
Dim Return_FReport As String

If (F > Return_F001) Then
Return_FReport = Return_FReport + "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" +
CStr(Return_F001)
ElseIf (Return_F001 > F And F > Return_F005) Then
Rem  下面没有验证
Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归
关系" + Chr(13) + Chr(10)
Return_FReport = Return_FReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_F005)
ElseIf (Return_F005 > F) Then
End If
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "F 检验结果" +
Return_FReport + Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************************************************************
***********
Rem T 检验

'OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "偏回归系数的检验
~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "T 检验:" + Chr(13) +
Chr(10)

'*****************************************
```

```
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "偏回归系数 F 检验:" +
Chr(13) + Chr(10)
Rem F 检验
    '
*************************************************************************
    Dim a() As Double
ReDim a(1 To Degree, 1 To Degree)
For i = 1 To Degree
For l = 1 To Degree
    If (i <> l) Then
    If (i < l) Then
    SP(l, i) = SP(i, l)
    End If
    End If
Next l
Next i
For i = 1 To Degree
For l = 1 To Degree
   If i = l Then
   a(i, l) = SS(i)
   Else
   a(i, l) = SP(i, l)
   End If
   OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "A(" + CStr(i) + "," +
CStr(l) + ")=" + CStr(a(i, l)) + Chr(13) + Chr(10)
  'Debug.Print "A(" + CStr(i) + "," + CStr(l) + ")=" + CStr(A(i, l))
Next l
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim obj3 As New Matrix001
Dim InverseA() As Double
obj3.InverseMatrix a, InverseA
For i = 1 To Degree
For l = 1 To Degree
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "InverseA(" + CStr(i) + ","
+ CStr(l) + ")=" + CStr(InverseA(i, l)) + Chr(13) + Chr(10)
Next l
Next i
'*************************************************************************
**

Dim c() As Double
ReDim c(1 To Degree)
```

```
For i = 1 To Degree
For l = 1 To Degree
   If i = l Then
   c(i) = InverseA(i, l)
   OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "C(" + CStr(i) + ")=" +
CStr(c(i)) + Chr(13) + Chr(10)
   End If
Next l
Next i
'*******************************************
Dim Up() As Double
ReDim Up(1 To Degree)
For i = 1 To Degree
Up(i) = Return_Coefficient1D(i + 1) ^ 2 / c(i)
Next i
Dim FF() As Double
ReDim FF(1 To Degree)
For i = 1 To Degree
    If Qy = 0 Then
    MsgBox "完全重合"
    FF(i) = 999999999
    Else
    FF(i) = Up(i) / (Qy / DF_r)
    End If
    OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "F" + CStr(i) + "=" +
CStr(FF(i)) + Chr(13) + Chr(10)
Next i

OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "变异原因              DF
SS                              MS                              F" + Chr(13) + Chr(10)
For i = 1 To Degree
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "x" + CStr(i) + "的偏回归"
+ "         " + CStr(1) + "         " + Chr(9) + CStr(Up(i)) + "         " + Chr(9) + CStr(Up(i)) + "         " +
Chr(9) + CStr(FF(i)) + "         " + Chr(13) + Chr(10)
Next i
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "离回归              " +
CStr(DF_r) + "         " + Chr(9) + CStr(SS_r) + "         " + Chr(9) + CStr(MS_r) + Chr(13) + Chr(10)

Dim Return_FP001 As Double
Dim Return_FP005 As Double
Return_FP001 = obj2.PFaN1N2X2(0.01, 10000000, 1, DF_r) ' 0.01 用千万
Return_FP005 = obj2.PFaN1N2X2(0.05, 1000000, 1, DF_r)     ' 0.05 用百万

Dim Remove() As Double '  为剔除 Upi 值
```

```
Dim NumRemove As Long

Dim Return_FPReport As String
For i = 1 To Degree
   If (FF(i) > Return_FP001) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "显著的回归关系" + "FF" + CStr(i) +
"=" + CStr(FF(i)) + ">" + "F0.01=" + CStr(Return_FP001) + Chr(13) + Chr(10)
   ElseIf (Return_FP001 > FF(i) And FF(i) > Return_FP005) Then
   Rem 下面没有验证
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "FF" + CStr(i) + "=" + CStr(FF(i)) + ">" + "F0.05=" +
CStr(Return_FP005) + Chr(13) + Chr(10)
   ElseIf (Return_FP005 > FF(i)) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "无显著的回归关系" + "F0.05=" +
CStr(Return_FP005) + ">" + "FF" + CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
'*****************************************
   NumRemove = NumRemove + 1
   ReDim Preserve Remove(NumRemove)
   Remove(NumRemove) = i
'*****************************************
   End If
Next i
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "F 检验结果" + Chr(13) +
Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + Return_FPReport +
Chr(13) + Chr(10)
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim StringY As String
For i = 1 To Degree + 1
   If i = 1 Then
   StringY = StringY + " (" + CStr(Return_Coefficient1D(i)) + ")" + "+"
   Else
       If i = Degree + 1 Then
       StringY = StringY + " (" + CStr(Return_Coefficient1D(i)) + ")" + "x^" + CStr(i - 1)
       Else
       StringY = StringY + " (" + CStr(Return_Coefficient1D(i)) + ")" + "x^" + CStr(i - 1) + "+"
       End If
   End If
Next i
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression + "Y=" + StringY + Chr(13) +
Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 拟合数据
```

```
Dim Return_Fitting() As Double
n = 0
For i = LBound(DataX) To UBound(DataX)
n = n + 1
ReDim Preserve Return_Fitting(n)
For l = 1 To Degree + 1
Return_Fitting(n) = Return_Fitting(n) + Return_Coefficient1D(l) * DataX(i) ^ (l - 1)
Next l
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression & "拟合数据" & n & "=" &
Return_Fitting(n) & Chr(13) & Chr(10)
Next i
n = 0

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 相关指数的计算
Dim SquareR As Double
Rem 另外的算法
'**************************************************
'Dim Part1 As Double, Part2 As Double
'For i = 1 To Num
'Part1 = Part1 + (DataY(i) - Return_Fitting(i)) ^ 2
'Part2 = Part2 + (DataY(i) - PartY / Num) ^ 2
'Next i
'SquareR = 1 - Part1 / Part2
'**************************************************
SquareR = 1 - SS_r / SSy
OneVeriblePolyNomialRegression = OneVeriblePolyNomialRegression & "相关指数 R^2" & "=" &
SquareR & Chr(13) & Chr(10)
End Function
```

'MultiElementLinearAnalysis_Equation2 多元线性分析

Rem 2，3 次是正确的
Rem 兼容 MultiElementLinearAnalysis_Equation 16-9-7 兼容错误，前面是多项式的公式，但是不知道为什么也可以求 3 元 1 次田间统计学列子
Rem 16-9-7 发现是如何兼容的？这是个问题
Rem 已经通过书上例子的测试，一次测试 16-8-19
Rem 增加 Kick
Rem 4 元线性，表达式系数通过，F 检验通过，偏相关 T 检验通过。其他没有数据 16-8-28

```
Public Function MultiElementLinearAnalysis_Equation2(ByRef DataX() As Double, _
ByRef DataY() As Double, ByVal Element As Long, ByRef Return_Coefficient1D() As Double, _
ByRef Kick As Long) As String
```

```
'*******************************************************
Rem 16-9-12
        Dim obj1 As New FileWrite001
        Dim ArrayRange As Integer
        ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
        Element = UBound(DataX, 2) - LBound(DataX, 2) + 1
        End If
        If ArrayRange = 1 Then
        Element = 1
        End If
'*******************************************************



Kick = 0 ' 初始化为 0 16-8-30
If (Element > 7) Then
MsgBox "大于 7 暂时不支持 Exit Function"
Exit Function
End If

        Dim Num As Long
        Dim i As Long
        Dim l As Long
        'Num = UBound(DataX) - LBound(DataX) + 1
        Num = UBound(DataY) - LBound(DataY) + 1

'*********************************************************
    Rem 16-8-30  增加的元素等于 1 的情况
    Rem element 是按照次数不是元的数编写的 16 -9 - 7
If (Element = 1) Then
    Dim cDataX() As Double
    ReDim cDataX(LBound(DataX) To UBound(DataX))

        'Dim obj1 As New FileWrite001
        'Dim ArrayRange As Integer
        'ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
                For i = LBound(DataX) To UBound(DataX) '这个是否正确？ 16-9-6 一次验证正
确
                cDataX(i) = DataX(i, 1)
                Next
        Else
            If ArrayRange = 1 Then
                For i = LBound(DataX) To UBound(DataX)
```

```
                    cDataX(i) = DataX(i)
                    Next
            Else
            MsgBox "ArrayRange 不支持或有错"
            End If
        End If
    'Dim ra As Double, rb As Double
    Rem  上面是把数据 DATAX 赋给 cDATAX,一维和 2 维都考虑了。
    Rem  当次数为 1 的时候转化成线性分析
    MsgBox "请选用一元一次线性分析，或者 1 元多次多项式分析。Exit Function"
    'MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
LinearRegressionAnalysis_Equation(cDataX, DataY, ra, rb)
    Exit Function
Else
    If Element < 1 Then
    MsgBox "不支持或有错 Exit Function"
    Exit Function
    End If
End If
    '*******************************************************

Dim AR As Long
Dim obj11 As New FileWrite001
AR = obj11.ArrayRangeD(DataX())
Dim cDataX1() As Double
If (AR = 1) Then
        MsgBox "数据为一维数据  And Element<>1  无法计算！ Exit Function"
        Exit Function ' 16-9-8
        '************************
        'MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
LinearRegressionAnalysis_Equation(DataX, DataY, ra, rb)
        MsgBox "DataX 为一维数据，现在转化为二维"
        ReDim cDataX1(LBound(DataX) To UBound(DataX), 1)
        For i = LBound(DataX) To UBound(DataX)
        cDataX1(i, 1) = DataX(i)
        Next
        '********************
        AR = 2 ' 16-9-8 增加
        '********************
Else
      If (AR <> 2) Then
      MsgBox "数组维度不正确 Exit Function"
      Exit Function
      End If
```

```
End If

'******************************
Rem 16-9-8
If AR = 2 Then
'Dim cDataX1() As Double
cDataX1 = DataX
End If
'******************************


'~~~~~~~~~~~~~~if ar=2 开始~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
If AR = 2 Then
Rem 求一级数据
Dim PartX() As Double
Dim PartXSquare() As Double
ReDim PartX(1 To Element)
ReDim PartXSquare(1 To Element)
For i = 1 To Element
For l = 1 To Num
PartX(i) = PartX(i) + cDataX1(l, i)
PartXSquare(i) = PartXSquare(i) + cDataX1(l, i) ^ 2
Next l
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "X" + CStr(i) +
"项累加                              =" + CStr(PartX(i)) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "X" + CStr(i) +
"项平均值                          =" + CStr(PartX(i) / Num) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "X" + CStr(i) +
"项平方累加                      =" + CStr(PartXSquare(i)) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "X" + CStr(i) +
"项平方累加平均值  =" + CStr(PartXSquare(i) / Num) + Chr(13) + Chr(10)
Next i
  MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************
Rem 前题是 Y 项和 X 项数量是相同的
Dim PartY As Double
Dim PartYSquare As Double
For l = 1 To Num
PartY = PartY + DataY(l)
PartYSquare = PartYSquare + DataY(l) ^ 2
Next l
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "Y 项累加
=" + CStr(PartY) + Chr(13) + Chr(10)
```

```
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "Y 项平均值=" + CStr(PartY / Num) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "Y 项平方累加=" + CStr(PartYSquare) + Chr(13) + Chr(10)
  MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'************************************
Rem  求 SSi 的值
Dim SS() As Double
ReDim SS(1 To Element) As Double
For i = 1 To Element
For l = 1 To Num
SS(i) = SS(i) + (cDataX1(l, i) - PartX(i) / Num) ^ 2
Next l
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "SS" + CStr(i) + "=" + CStr(SS(i)) + Chr(13) + Chr(10)
Next i
  MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'**************************************
Rem  求 SSy 的值
Dim SSy As Double
For i = 1 To Num
SSy = SSy + (DataY(i) - PartY / Num) ^ 2
Next i
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "y 变量的总平方和 SSy" + "=" + CStr(SSy) + Chr(13) + Chr(10)
  MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*****************************************************************************
Rem  求 SPik 的值
Rem  田间统计分析书上测试已经通过（16-8-24）
Dim n As Long, m As Long
Dim SP() As Double
ReDim SP(1 To Element, 1 To Element) As Double
For i = 1 To Element
For l = 1 To Element
    If (i <> l) Then
    If (i < l) Then


        For n = 1 To Num
        SP(i, l) = SP(i, l) + (cDataX1(n, i) - PartX(i) / Num) * (cDataX1(n, l) - PartX(l) / Num)
        Next n
        MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "SP(" +
```

```
CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i, l)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(L) + ")=" + CStr(SP(i, L))
    End If
    End If
Next l
Next i
     MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)

'***************************************************************************
    Rem 求 SPi0 的值
    Dim SP0() As Double
    ReDim SP0(1 To Element) As Double
    For i = 1 To Element
        For n = 1 To Num
        SP0(i) = SP0(i) + (cDataX1(n, i) - PartX(i) / Num) * (DataY(n) - PartY / Num)
        Next n
        MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "SP0("
+ CStr(i) + ")=" + CStr(SP0(i)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(L) + ")=" + CStr(SP(i, L))
    Next i
    MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
' ***************************************************************************
    Dim a() As Double
ReDim a(1 To Element, 1 To Element)
For i = 1 To Element
For l = 1 To Element
    If (i <> l) Then
    If (i < l) Then
    SP(l, i) = SP(i, l)
    End If
    End If
Next l
Next i
For i = 1 To Element
For l = 1 To Element
   If i = l Then
   a(i, l) = SS(i)
   Else
   a(i, l) = SP(i, l)
   End If
   MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "A(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(a(i, l)) + Chr(13) + Chr(10)
```

```
Next l
Next i
Dim obj3 As New Matrix001
Dim InverseA() As Double
obj3.InverseMatrix a, InverseA
For i = 1 To Element
For l = 1 To Element
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "InverseA(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(InverseA(i, l)) + Chr(13) + Chr(10)
Next l
Next i
'*****************************************************************************
*****
'准备放系数求解
Rem 16-9-7
        Dim b1() As Double
        Dim SP01() As Double
        ReDim SP01(LBound(SP0) To UBound(SP0), 1)
        For i = LBound(SP0) To UBound(SP0)
        SP01(i, 1) = SP0(i)
        'Debug.Print "SP01=" & SP01(i, 1)
        Next i
        obj3.MatrixMultiplication InverseA, SP01, b1
        'For i = 1 To Element
        'Debug.Print "b1=" & b1(i, 1)
        'Next
        Dim b2() As Double
        ReDim b2(1 To Element + 1)
        Dim Part1 As Double
        For i = 1 To Element + 1
        If i = Element + 1 Then
              b2(1) = PartY / Num - Part1
        Else
            Part1 = Part1 + b1(i, 1) * PartX(i) / Num
            b2(i + 1) = b1(i, 1)
        End If
        Next i
        'For i = 1 To Element + 1
        'Debug.Print "b2=" & b2(i)
        'Next

        For i = LBound(b2) To UBound(b2)
        ReDim Preserve Return_Coefficient1D(i)
        Return_Coefficient1D(i) = b2(i)
```

```
        MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + CStr(i - 1) + "次项系数=" + CStr(Return_Coefficient1D(i)) + Chr(13) + Chr(10)
        Next i

        '****************************************
         MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
        Dim Return_Fitting() As Double
        ReDim Return_Fitting(1 To Num)
        For l = 1 To Num
        For i = 1 To Element
        Return_Fitting(l) = Return_Fitting(l) + Return_Coefficient1D(i + 1) * cDataX1(l, i)
        Next i
        Return_Fitting(l) = Return_Fitting(l) + Return_Coefficient1D(1)
        MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + CStr(l) + "项拟合数=" + CStr(Return_Fitting(l)) + Chr(13) + Chr(10)
        Next l
        MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*****************************************************************************
    Rem 离回归标准误
    Dim Se As Double
    Dim Uy As Double
    Dim Qy As Double
    'For i = 1 To Num    ' 等价于下面的 Uy
    ' Uy = Uy + (Return_Fitting(i) - PartY / Num) ^ 2
    'Next i
    For i = 1 To Element
    Uy = Uy + Return_Coefficient1D(i + 1) * SP0(i)
    Next i
    Qy = SSy - Uy
End If     'if ar=2 是这个的 END,放这里正确否？16-9-4!!!!!!!!!!!!!!!!!!!!
'~~~~~~~~~~~~~~~~~~~~~~~if ar=2 结束~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Se = Sqr(Qy / (Num - Element - 1))
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "估计标准误Se=" + CStr(Se) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "多元离回归平方和 Qy=SSr=" + CStr(Qy) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "M 元回归平方和 Uy=SSR=" + CStr(Uy) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
```

```
'*************************************************************************************
*********
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
"~~~~~~~~~~~~多元线性显著性检验~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim SSr As Double
Dim SS_r As Double
Dim DFy As Double
Dim DFR As Double
Dim DF_r As Double
Dim MSR As Double
Dim MS_r As Double
SSr = Uy
SS_r = Qy
DFy = Num - 1
DFR = Element
DF_r = Num - Element - 1
MSR = SSr / DFR
MS_r = SS_r / DF_r
Dim F As Double

    If SS_r = 0 Then
    MsgBox "完全重合"
    F = 999999999
    Else
    F = MSR / MS_r '16-9-6 改到 IF 内
    End If
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "变异原因
DF                          SS                          MS                          F" + Chr(13) +
Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "多元回归
" + CStr(DFR) + "          " + Chr(9) + CStr(SSr) + "          " + Chr(9) + CStr(MSR) + "          " + Chr(9) +
CStr(F) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "离回归
" + CStr(DF_r) + "          " + Chr(9) + CStr(SS_r) + "          " + Chr(9) + CStr(MS_r) + "          " + Chr(13)
+ Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "总和
" + CStr(DFy) + "          " + Chr(9) + CStr(SSy) + Chr(13) + Chr(10)
'*****************************************************************
'Dim obj As New F001
'Return_F001 = obj.PFaN1N2X2(0.01, 0.01, Return_dfR, Return_df_r)
'Return_F005 = obj.PFaN1N2X2(0.05, 0.01, Return_dfR, Return_df_r)
Rem 当多元线性回归关系经显著检验为显著时，还必须逐一对各偏回归系数进行显著性检
验，发现和剔除不显著的偏回归关系对应的自变量。
```

```
Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj2 As New F001
Return_F001 = obj2.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj2.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万
Dim Return_FReport As String

If (F > Return_F001) Then
Return_FReport = Return_FReport + "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" +
CStr(Return_F001)
ElseIf (Return_F001 > F And F > Return_F005) Then
Rem  下面没有验证
Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归
关系" + Chr(13) + Chr(10)
Return_FReport = Return_FReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_F005)
ElseIf (Return_F005 > F) Then
Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">"
+ "F=" + CStr(F)
End If
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "F 检验结果"
+ Return_FReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'**************************************************************************
***********
Rem T 检验
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "偏回归系数
的检验~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "T 检验:" +
Chr(13) + Chr(10)
'A 从这里取走 16-9-7
'****************************************
Rem 16-9-5
Dim b() As Double
Dim k() As Double
ReDim b(1 To Element, 1)
For i = 1 To Element
b(i, 1) = Return_Coefficient1D(i + 1)
Next i
'Debug.Print "b=" & b(i, 1)
obj3.MatrixMultiplication a, b, k
For i = 1 To Element
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "K(" + CStr(i)
```

```
+ ", " + CStr(1) + ") = " + "SP(" + CStr(i) + ",y)=" + CStr(k(i, 1)) + Chr(13) + Chr(10)
Next i
'*****************************************
Dim c() As Double
ReDim c(1 To Element)
For i = 1 To Element
For l = 1 To Element
   If i = l Then
   c(i) = InverseA(i, l)
   MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "C(" + CStr(i)
+ ")=" + CStr(c(i)) + Chr(13) + Chr(10)
   End If
Next l
Next i

Dim Sb() As Double
ReDim Sb(1 To Element)
Dim t() As Double
ReDim t(1 To Element)

For i = 1 To Element
Sb(i) = Se * Sqr(c(i))
    If Sb(i) = 0 Then
    Rem 这个有没有用还要验证 16-8-28
    MsgBox "完全重合"
    t(i) = 999999999
    Else
    t(i) = Return_Coefficient1D(i + 1) / Sb(i)
    MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "t(" +
CStr(i) + ")=" + CStr(t(i)) + Chr(13) + Chr(10)
    End If
Next i
Dim Return_T001 As Double, Return_T005 As Double
Dim obj4 As New T001
'*****************************************
Return_T001 = obj4.t(0.005, DF_r)
Return_T005 = obj4.t(0.025, DF_r) '16-9-7 已经修改
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
'*********************************************************************
Rem 16-9-7 参照田间统计，T 取绝度值
Dim T1() As Double
n = 0
For i = LBound(t) To UBound(t)
n = n + 1
```

```
ReDim Preserve T1(n)
T1(i) = Abs(t(i))
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "t(" + CStr(i) +
")=" + CStr(t(i)) + Chr(13) + Chr(10)
Next i
n = 0
'************************************************
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "以下 T 为绝
对值" + Chr(13) + Chr(10)
'*********************************************************************
Dim Return_TReport As String
For i = 1 To Element
   If (T1(i) > Return_T001) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "显著的回归关系" + "T" + CStr(i) + "="
+ CStr(T1(i)) + ">" + "T0.01=" + CStr(Return_T001) + Chr(13) + Chr(10)
   ElseIf (Return_T001 > T1(i) And T1(i) > Return_T005) Then
   Rem  下面没有验证
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "T" + CStr(i) + "=" + CStr(T1(i)) + ">" + "T0.05=" +
CStr(Return_T005) + Chr(13) + Chr(10)
   ElseIf (Return_T005 > T1(i)) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "无显著的回归关系" + "T0.05=" +
CStr(Return_T005) + ">" + "T" + CStr(i) + "=" + CStr(T1(i)) + Chr(13) + Chr(10)
   End If
Next i
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "T 检验结果"
+ Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
Return_TReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************************************************************
************
Rem F 检验
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "F 检验:" +
Chr(13) + Chr(10)
Dim Up() As Double
ReDim Up(1 To Element)
For i = 1 To Element
Up(i) = Return_Coefficient1D(i + 1) ^ 2 / c(i)
Next i
Dim FF() As Double
ReDim FF(1 To Element)
For i = 1 To Element
```

```
    If Qy = 0 Then
    MsgBox "完全重合"
    FF(i) = 999999999
    Else
    FF(i) = Up(i) / (Qy / DF_r)
    End If
    MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "F" +
CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
Next i

MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "变异原因
DF                    SS                    MS                    F" + Chr(13) +
Chr(10)
For i = 1 To Element
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "x" + CStr(i) +
"的偏回归" + "      " + CStr(1) + "      " + Chr(9) + CStr(Up(i)) + "      " + Chr(9) + CStr(Up(i)) +
"      " + Chr(9) + CStr(FF(i)) + "      " + Chr(13) + Chr(10)
Next i
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "离回归
" + CStr(DF_r) + "      " + Chr(9) + CStr(SS_r) + "      " + Chr(9) + CStr(MS_r) + Chr(13) + Chr(10)

Dim Return_FP001 As Double
Dim Return_FP005 As Double
Return_FP001 = obj2.PFaN1N2X2(0.01, 10000000, 1, DF_r) ' 0.01 用千万
Return_FP005 = obj2.PFaN1N2X2(0.05, 1000000, 1, DF_r)    ' 0.05 用百万
'Dim Return_FPReport As String
'For i = 1 To Element
'If (FF(i) > Return_FP001) Then
'Return_FPReport = "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" + CStr(Return_FP001)
'ElseIf (Return_FP001 > F And F > Return_FP005) Then
Rem  下面没有验证
'Return_FPReport = "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + Chr(13) +
Chr(10)
'Return_FPReport = Return_FPReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_FP005)
'ElseIf (Return_FP005 > F) Then
'Return_FPReport = "无显著的回归关系" + "F0.05=" + CStr(Return_FP005) + ">" + "F=" + CStr(F)
'End If
'Next
'MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "F 检验结果"
+ Return_FPReport + Chr(13) + Chr(10)
'MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim Remove() As Double '  为剔除 Upi 值
Dim NumRemove As Long
```

```
Dim Return_FPReport As String
For i = 1 To Element
   If (FF(i) > Return_FP001) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "显著的回归关系" + "FF" + CStr(i) +
"=" + CStr(FF(i)) + ">" + "F0.01=" + CStr(Return_FP001) + Chr(13) + Chr(10)
   ElseIf (Return_FP001 > FF(i) And FF(i) > Return_FP005) Then
   Rem 下面没有验证
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "FF" + CStr(i) + "=" + CStr(FF(i)) + ">" + "F0.05=" +
CStr(Return_FP005) + Chr(13) + Chr(10)
   ElseIf (Return_FP005 > FF(i)) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "无显著的回归关系" + "F0.05=" +
CStr(Return_FP005) + ">" + "FF" + CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
'*****************************************
   NumRemove = NumRemove + 1
   ReDim Preserve Remove(NumRemove)
   Remove(NumRemove) = i
'*****************************************
   End If
Next i
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "F 检验结果:"
+ Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
Return_FPReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************
Rem 非必要程序没有去验证，来源于田间试验与统计分析
Dim SumUp As Double
For i = 1 To Element
SumUp = SumUp + Up(i)
Next i
l = 0
If Uy = SumUp Then
MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "各自变量独
立不相关" + Chr(13) + Chr(10)
Else
      For i = 1 To Element
         If Uy > SumUp And Return_Coefficient1D(i + 1) > 0 Then
         l = l + 1
         End If
      Next i
```

```
        If l = 3 Then
        MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "变量之间正相关" + Chr(13) + Chr(10)
        End If
End If
l = 0
'*****************************
Dim UpArray() As Double
Dim rlu() As Double, rul() As Double
Dim Which() As Long
Dim obj5 As New Array1D001
n = 0

If NumRemove >= 1 Then
    For i = LBound(Remove) To UBound(Remove)
    n = n + 1
    ReDim Preserve UpArray(n)
    UpArray(n) = Up(Remove(i))
    Next i
    n = 0

    For i = LBound(UpArray) To UBound(UpArray)
    MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "UpArray(" + CStr(i) + ")=" + CStr(UpArray(i)) + Chr(13) + Chr(10)
    Next

    Dim MixNum As Double
    MixNum = obj5.Array1D_Mix(UpArray) ' 如果有两个数据一样，会给出一个 16-8-30
    For i = LBound(Up) To UBound(Up)
      If MixNum = Up(i) Then ' 如果有两个数据一样，会给出前面的 16-8-30
      Kick = i
      MultiElementLinearAnalysis_Equation2 = MultiElementLinearAnalysis_Equation2 + "需要剔除的组数是" + CStr(Kick) + Chr(13) + Chr(10)
      End If
    Next i

End If
'*********************************************************************

End Function
```

'MultiElementLinearAnalysis_Equation3 多元线性分析并返回多种值

Rem　_Equation3 的创建主要是为了返回函数中各种值 16-9-5

```
Public Function MultiElementLinearAnalysis_Equation3(ByRef DataX() As Double, _
ByRef DataY() As Double, ByVal Element As Long, ByRef Return_Coefficient1D() As Double, _
ByRef Kick As Long, ByRef Return_SS() As Double, ByRef Return_SSy As Double, ByRef Return_Uy
_
As Double, ByRef Return_SP() As Double, ByRef Return_SPy() As Double) As String


'*******************************************************
Rem 16-9-12
        Dim obj1 As New FileWrite001
        Dim ArrayRange As Integer
        ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
        Element = UBound(DataX, 2) - LBound(DataX, 2) + 1
        End If
        If ArrayRange = 1 Then
        Element = 1
        End If
'*******************************************************



  Kick = 0 ' 初始化为 0 16-8-30
If (Element > 7) Then
MsgBox "大于 7 暂时不支持"
End If


If (Element < 2) Then
    '*************************************
    Rem 16-8-30  增加的元素等于 1 的情况
    If (Element = 1) Then
    Dim cDataX() As Double
    ReDim cDataX(LBound(DataX) To UBound(DataX))
    Dim i As Long
        'Dim obj1 As New FileWrite001
        'Dim ArrayRange As Integer
        'ArrayRange = obj1.ArrayRangeD(DataX)
```

```
        If ArrayRange = 2 Then
            For i = LBound(DataX) To UBound(DataX)
            cDataX(i) = DataX(i, 1)
            Next
        Else
            If ArrayRange = 1 Then
            For i = LBound(DataX) To UBound(DataX)
            cDataX(i) = DataX(i)
            Next
            End If
        End If
    Dim ra As Double, rb As Double
    MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
LinearRegressionAnalysis_Equation(cDataX, DataY, ra, rb)
    Kick = -1 '16-9-6 用来表示 1 元线性分析
    Exit Function
    Else
    MsgBox "不支持或有错"
    Exit Function
    End If
End If
 '*************************************

Dim AR As Long
Dim obj11 As New FileWrite001
AR = obj11.ArrayRangeD(DataX())
If (AR = 1) Then
'MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation(DataX, DataY,
Element, Return_Coefficient1D())
MultiElementLinearAnalysis_Equation3 = OneVeriblePolyNomialRegression(DataX, DataY,
Element, Return_Coefficient1D())
Else
   If (AR = 2) Then
       '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       Rem 主要新程序位置
       Rem 得到行列式
       Dim DataMatrix() As Double
       Dim DataMatriy() As Double
       Dim Num As Long
       Dim l As Long
       Num = UBound(DataX) - LBound(DataX) + 1
       ReDim DataMatrix(1 To Num, 1 To (Element + 1)) As Double
       For l = 1 To (Element + 1)
       For i = 1 To Num
```

```
            If l = 1 Then
            DataMatrix(i, l) = 1
            Else
            DataMatrix(i, l) = DataX(i, l - 1)
            End If
        'DataMatrix(i, L) = DataX(i) ^ (L - 1)
        'Debug.Print "DataMatrix(" & i & "," & L & ")=" & DataX(i) ^ (L - 1)
        Next i
        Next l
        '**************************************************
        Num = UBound(DataY) - LBound(DataY) + 1
          ReDim DataMatriy(1 To Num, 1) As Double
              For i = 1 To Num
              DataMatriy(i, 1) = DataY(i)
              Next i
          '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          Dim obj As New Matrix001
          Dim DataMatrixT() As Double
          obj.TransposedMatrix DataMatrix, DataMatrixT '求 X 得转置矩阵 X^T
          Dim R3() As Double
          Dim R1() As Double
          Dim R2() As Double
          obj.MatrixMultiplication DataMatrixT, DataMatrix, R3 '求 X^T * X
          obj.InverseMatrix R3, R1 '求(X^T * X)^(-1)
          obj.MatrixMultiplication DataMatrixT, DataMatriy, R2 '求 X^T * Y
            MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"***~~~~~~~~~~~~~开始多元线性分析~~~~~~~~~~~~~~~~***" + Chr(13) + Chr(10)
          Dim b1() As Double
          obj.MatrixMultiplication R1, R2, b1 '求(X^T * X)^(-1) * (X^T * Y) =  系数矩阵 b
          For i = LBound(b1, 1) To UBound(b1, 1)
          ReDim Preserve Return_Coefficient1D(i)
          Return_Coefficient1D(i) = b1(i, 1)
          MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + CStr(i -
1) + "次项系数=" + CStr(Return_Coefficient1D(i)) + Chr(13) + Chr(10)
          Next
          '******************************
          MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
        Dim Return_Fitting() As Double
        ReDim Return_Fitting(1 To Num)
        For l = 1 To Num
        For i = 1 To Element
        Return_Fitting(l) = Return_Fitting(l) + Return_Coefficient1D(i + 1) * DataX(l, i)
        Next i
```

```
        Return_Fitting(l) = Return_Fitting(l) + Return_Coefficient1D(1)
        MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + CStr(l) +
"项拟合数=" + CStr(Return_Fitting(l)) + Chr(13) + Chr(10)
        Next l
         MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
        '********************************


   Else
   MsgBox "数组维度不等于元数"
   End If 'ar=2
End If 'ar =1
'~~~~~~~~~~~~~~if ar=2 开始~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
If AR = 2 Then
Rem  求一级数据
Dim PartX() As Double
Dim PartXSquare() As Double
ReDim PartX(1 To Element)
ReDim PartXSquare(1 To Element)
For i = 1 To Element
For l = 1 To Num
PartX(i) = PartX(i) + DataX(l, i)
PartXSquare(i) = PartXSquare(i) + DataX(l, i) ^ 2
Next l
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "X" + CStr(i) +
"项累加=" + CStr(PartX(i)) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "X" + CStr(i) +
"项平均值=" + CStr(PartX(i) / Num) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "X" + CStr(i) +
"项平方累加=" + CStr(PartXSquare(i)) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "X" + CStr(i) +
"项平方累加平均值=" + CStr(PartXSquare(i) / Num) + Chr(13) + Chr(10)
Next i
  MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************
Rem  前题是 Y 项和 X 项数量是相同的
Dim PartY As Double
Dim PartYSquare As Double
For l = 1 To Num
PartY = PartY + DataY(l)
PartYSquare = PartYSquare + DataY(l) ^ 2
Next l
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "Y 项累加=" +
```

```
CStr(PartY) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "Y 项平均值
=" + CStr(PartY / Num) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "Y 项平方累
加=" + CStr(PartYSquare) + Chr(13) + Chr(10)
  MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'**********************************
Rem  求 SSi 的值
'Dim ss() As Double
ReDim SS(1 To Element) As Double
For i = 1 To Element
For l = 1 To Num
SS(i) = SS(i) + (DataX(l, i) - PartX(i) / Num) ^ 2
ReDim Preserve Return_SS(i)
Return_SS(i) = SS(i)
Next l
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "SS" + CStr(i)
+ "=" + CStr(SS(i)) + Chr(13) + Chr(10)
Next i
  MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'**********************************
Rem  求 SSy 的值
Dim SSy As Double
For i = 1 To Num
SSy = SSy + (DataY(i) - PartY / Num) ^ 2
Next i
Return_SSy = SSy
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "y 变量的总
平方和 SSy" + "=" + CStr(SSy) + Chr(13) + Chr(10)
  MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'********************************************************************************
Rem  求 SPik 的值
Rem  田间统计分析书上测试已经通过（16-8-24）
Dim n As Long, m As Long
Dim SP() As Double
ReDim SP(1 To Element, 1 To Element) As Double
ReDim Return_SP(1 To Element, 1 To Element) As Double
For i = 1 To Element
For l = 1 To Element
     '*******************************
    Rem 16-9-5
```

```
    For m = 1 To Num
    Return_SP(i, l) = Return_SP(i, l) + (DataX(m, i) - PartX(i) / Num) * (DataX(m, l) - PartX(l) / Num)
    Next m
    '*******************************
    If (i <> l) Then
    If (i < l) Then

        For n = 1 To Num
        SP(i, l) = SP(i, l) + (DataX(n, i) - PartX(i) / Num) * (DataX(n, l) - PartX(l) / Num)

        Next n
        MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "SP(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i, l)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(L) + ")=" + CStr(SP(i, L))
    End If
    End If
Next l
Next i
      MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)

'*************************************************************************
    Rem 求 SPi0 的值
    Dim SP0() As Double
    ReDim SP0(1 To Element) As Double
    For i = 1 To Element
        For n = 1 To Num
        SP0(i) = SP0(i) + (DataX(n, i) - PartX(i) / Num) * (DataY(n) - PartY / Num)
        Next n
        MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "SP0("
+ CStr(i) + ")=" + CStr(SP0(i)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(L) + ")=" + CStr(SP(i, L))
    Next i
    MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
    '
*************************************************************************
    Rem 离回归标准误
    Dim Se As Double
    Dim Uy As Double
    Dim Qy As Double
    'For i = 1 To Num    ' 等价于下面的 Uy
    ' Uy = Uy + (Return_Fitting(i) - PartY / Num) ^ 2
    'Next i
```

```
    For i = 1 To Element
    Uy = Uy + Return_Coefficient1D(i + 1) * SP0(i)
    Next i
    Return_Uy = Uy
    Qy = SSy - Uy
End If     'if ar=2  是这个的 END,放这里正确否？16-9-4!!!!!!!!!!!!!!!!!!!!
'~~~~~~~~~~~~~~~~~~~~~~~~if ar=2  结束~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Se = Sqr(Qy / (Num - Element - 1))
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "估计标准误
Se=" + CStr(Se) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "多元离回归
平方和 Qy=SSr=" + CStr(Qy) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "M 元回归平
方和 Uy=SSR=" + CStr(Uy) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)



'*********************************************************************************
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~多元线性显著性检验~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim SSr As Double
Dim SS_r As Double
Dim DFy As Double
Dim DFR As Double
Dim DF_r As Double
Dim MSR As Double
Dim MS_r As Double
SSr = Uy
SS_r = Qy
DFy = Num - 1
DFR = Element
DF_r = Num - Element - 1
MSR = SSr / DFR
MS_r = SS_r / DF_r
Dim F As Double

    If SS_r = 0 Then
    MsgBox "完全重合"
    F = 999999999
    Else
    F = MSR / MS_r ' 16-9-6  更改到 IF 内
    End If
```

```
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "变异原因
DF                    SS                    MS                    F" + Chr(13) +
Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "多元回归
" + CStr(DFR) + "          " + Chr(9) + CStr(SSr) + "          " + Chr(9) + CStr(MSR) + "          " + Chr(9) +
CStr(F) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "离回归
" + CStr(DF_r) + "          " + Chr(9) + CStr(SS_r) + "          " + Chr(9) + CStr(MS_r) + "          " + Chr(13)
+ Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "总和
" + CStr(DFy) + "          " + Chr(9) + CStr(SSy) + Chr(13) + Chr(10)
'*********************************************************************
'Dim obj As New F001
'Return_F001 = obj.PFaN1N2X2(0.01, 0.01, Return_dfR, Return_df_r)
'Return_F005 = obj.PFaN1N2X2(0.05, 0.01, Return_dfR, Return_df_r)
Rem 当多元线性回归关系经显著检验为显著时，还必须逐一对各偏回归系数进行显著性检
验，发现和剔除不显著的偏回归关系对应的自变量。
Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj2 As New F001
Return_F001 = obj2.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj2.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万
Dim Return_FReport As String

If (F > Return_F001) Then
Return_FReport = Return_FReport + "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" +
CStr(Return_F001)
ElseIf (Return_F001 > F And F > Return_F005) Then
Rem 下面没有验证
Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归
关系" + Chr(13) + Chr(10)
Return_FReport = Return_FReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_F005)
ElseIf (Return_F005 > F) Then
Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">"
+ "F=" + CStr(F)
End If
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "F 检验结果"
+ Return_FReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************************************************************
Rem T 检验
```

```
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "偏回归系数的检验~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "T 检验:" + Chr(13) + Chr(10)
Dim a() As Double
ReDim a(1 To Element, 1 To Element)
For i = 1 To Element
For l = 1 To Element
    If (i <> l) Then
    If (i < l) Then
    SP(l, i) = SP(i, l)
    End If
    End If
Next l
Next i
For i = 1 To Element
For l = 1 To Element
  If i = l Then
  a(i, l) = SS(i)
  Else
  a(i, l) = SP(i, l)
  End If
  MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "A(" + CStr(i) + "," + CStr(l) + ")=" + CStr(a(i, l)) + Chr(13) + Chr(10)
Next l
Next i
'*****************************************
Rem 16-9-5
Dim b() As Double
Dim k() As Double
ReDim b(1 To Element, 1)
For i = 1 To Element
b(i, 1) = Return_Coefficient1D(i + 1)
Next i
obj.MatrixMultiplication a, b, k
For i = 1 To Element
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "K(" + CStr(i) + ", " + CStr(1) + ") = " + "SP(" + CStr(i) + ",y)=" + CStr(k(i, 1)) + Chr(13) + Chr(10)
Next i
Return_SPy = k
'*****************************************
Dim obj3 As New Matrix001
Dim InverseA() As Double
obj3.InverseMatrix a, InverseA
```

```
For i = 1 To Element
For l = 1 To Element
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "InverseA(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(InverseA(i, l)) + Chr(13) + Chr(10)
Next l
Next i
Dim c() As Double
ReDim c(1 To Element)
For i = 1 To Element
For l = 1 To Element
   If i = l Then
   c(i) = InverseA(i, l)
   MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "C(" + CStr(i)
+ ")=" + CStr(c(i)) + Chr(13) + Chr(10)
   End If
Next l
Next i

Dim Sb() As Double
ReDim Sb(1 To Element)
Dim t() As Double
ReDim t(1 To Element)

For i = 1 To Element
Sb(i) = Se * Sqr(c(i))
    If Sb(i) = 0 Then
    Rem 这个有没有用还要验证 16-8-28
    MsgBox "完全重合"
    t(i) = 999999999
    Else
    t(i) = Return_Coefficient1D(i + 1) / Sb(i)
    MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "t(" +
CStr(i) + ")=" + CStr(t(i)) + Chr(13) + Chr(10)
    End If
Next i
Dim Return_T001 As Double, Return_T005 As Double
Dim obj4 As New T001
Return_T001 = obj4.t(0.005, DF_r) '16-9-7 日改
Return_T005 = obj4.t(0.025, DF_r)
Dim Return_TReport As String
'************************************************
Rem 16-9-7 参照田间统计，T 取绝度值
Dim T1() As Double
n = 0
```

```
For i = LBound(t) To UBound(t)
n = n + 1
ReDim Preserve T1(n)
T1(i) = Abs(t(i))
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "t(" + CStr(i) +
")=" + CStr(t(i)) + Chr(13) + Chr(10)
Next i
'************************************************
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "以下 T 为绝
对值" + Chr(13) + Chr(10)
For i = 1 To Element
   If (T1(i) > Return_T001) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "显著的回归关系" + "T" + CStr(i) + "="
+ CStr(T1(i)) + ">" + "T0.01=" + CStr(Return_T001) + Chr(13) + Chr(10)
   ElseIf (Return_T001 > T1(i) And T1(i) > Return_T005) Then
   Rem  下面没有验证
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "T" + CStr(i) + "=" + CStr(T1(i)) + ">" + "T0.05=" +
CStr(Return_T005) + Chr(13) + Chr(10)
   ElseIf (Return_T005 > T1(i)) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "无显著的回归关系" + "T0.05=" +
CStr(Return_T005) + ">" + "T" + CStr(i) + "=" + CStr(T1(i)) + Chr(13) + Chr(10)
   End If
Next i
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "T 检验结果"
+ Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
Return_TReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'************************************************************************
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "F 检验:" +
Chr(13) + Chr(10)
Rem F 检验
Dim Up() As Double
ReDim Up(1 To Element)
For i = 1 To Element
Up(i) = Return_Coefficient1D(i + 1) ^ 2 / c(i)
Next i
Dim FF() As Double
ReDim FF(1 To Element)
For i = 1 To Element
    If Qy = 0 Then
    MsgBox "完全重合"
```

```
    FF(i) = 999999999
    Else
    FF(i) = Up(i) / (Qy / DF_r)
    End If
    MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "F" +
CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
Next i

MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "变异原因
DF                    SS                    MS                    F" + Chr(13) +
Chr(10)
For i = 1 To Element
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "x" + CStr(i) +
"的偏回归" + "          " + CStr(1) + "          " + Chr(9) + CStr(Up(i)) + "          " + Chr(9) + CStr(Up(i)) +
"          " + Chr(9) + CStr(FF(i)) + "          " + Chr(13) + Chr(10)
Next i
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "离回归
" + CStr(DF_r) + "          " + Chr(9) + CStr(SS_r) + "          " + Chr(9) + CStr(MS_r) + Chr(13) + Chr(10)

Dim Return_FP001 As Double
Dim Return_FP005 As Double
Return_FP001 = obj2.PFaN1N2X2(0.01, 10000000, 1, DF_r) ' 0.01 用千万
Return_FP005 = obj2.PFaN1N2X2(0.05, 1000000, 1, DF_r)     ' 0.05 用百万
'Dim Return_FPReport As String
'For i = 1 To Element
'If (FF(i) > Return_FP001) Then
'Return_FPReport = "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" + CStr(Return_FP001)
'ElseIf (Return_FP001 > F And F > Return_FP005) Then
Rem  下面没有验证
'Return_FPReport = "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + Chr(13) +
Chr(10)
'Return_FPReport = Return_FPReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_FP005)
'ElseIf (Return_FP005 > F) Then
'Return_FPReport = "无显著的回归关系" + "F0.05=" + CStr(Return_FP005) + ">" + "F=" + CStr(F)
'End If
'Next
'MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "F 检验结果"
+ Return_FPReport + Chr(13) + Chr(10)
'MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim Remove() As Double '  为剔除 Upi 值
Dim NumRemove As Long

Dim Return_FPReport As String
```

```
For i = 1 To Element
   If (FF(i) > Return_FP001) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "显著的回归关系" + "FF" + CStr(i) +
"=" + CStr(FF(i)) + ">" + "F0.01=" + CStr(Return_FP001) + Chr(13) + Chr(10)
   ElseIf (Return_FP001 > FF(i) And FF(i) > Return_FP005) Then
   Rem  下面没有验证
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "FF" + CStr(i) + "=" + CStr(FF(i)) + ">" + "F0.05=" +
CStr(Return_FP005) + Chr(13) + Chr(10)
   ElseIf (Return_FP005 > FF(i)) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "无显著的回归关系" + "F0.05=" +
CStr(Return_FP005) + ">" + "FF" + CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
'*****************************************
   NumRemove = NumRemove + 1
   ReDim Preserve Remove(NumRemove)
   Remove(NumRemove) = i
'*****************************************
   End If
Next i
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "F 检验结果"
+ Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
Return_FPReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)



'*************************
Rem  非必要程序没有去验证，来源于田间试验与统计分析
Dim SumUp As Double
For i = 1 To Element
SumUp = SumUp + Up(i)
Next i

l = 0

If Uy = SumUp Then
MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "各自变量独
立不相关" + Chr(13) + Chr(10)
Else
        For i = 1 To Element
           If Uy > SumUp And Return_Coefficient1D(i + 1) > 0 Then
           l = l + 1
           End If
```

```
        Next i

        If l = 3 Then
        MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "变量之间正相关" + Chr(13) + Chr(10)
        End If
End If
l = 0
'*****************************
Dim UpArray() As Double
Dim rlu() As Double, rul() As Double
Dim Which() As Long
Dim obj5 As New Array1D001
n = 0

If NumRemove >= 1 Then
    For i = LBound(Remove) To UBound(Remove)
    n = n + 1
    ReDim Preserve UpArray(n)
    UpArray(n) = Up(Remove(i))
    Next i
    n = 0

    For i = LBound(UpArray) To UBound(UpArray)
    MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "UpArray(" + CStr(i) + ")=" + CStr(UpArray(i)) + Chr(13) + Chr(10)
    Next

    Dim MixNum As Double
    MixNum = obj5.Array1D_Mix(UpArray) ' 如果有两个数据一样，会给出一个 16-8-30
    For i = LBound(Up) To UBound(Up)
        If MixNum = Up(i) Then ' 如果有两个数据一样，会给出前面的 16-8-30
        Kick = i
        MultiElementLinearAnalysis_Equation3 = MultiElementLinearAnalysis_Equation3 + "需要剔除的组数是" + CStr(Kick) + Chr(13) + Chr(10)
        End If
    Next i

End If
'*****************************************************************

End Function
```

'TheBestMultipleLinearRegressionEquation 最好的多元线性回归

Rem 最终的线性分析
Rem 当被削减程一元线性的情况没有测试。只测试了 3 元消减程 2 元有书上的数值支持 16-8-30
Rem 急需要 4 元相关的例子
Rem 16-9-8MultiElementLinearAnalysis_Equation3 改成 MultiElementLinearAnalysis_Equation4

```
Public Function TheBestMultipleLinearRegressionEquation(ByRef DataX() As Double, _
ByRef DataY() As Double, ByVal Element As Long, ByRef Return_Coefficient1D() As Double, _
ByRef Kick As Long) As String

'*******************************************************
        Rem 16-9-12
        Dim obj1 As New FileWrite001
        Dim ArrayRange As Integer
        ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
        Element = UBound(DataX, 2) - LBound(DataX, 2) + 1
        End If
        If ArrayRange = 1 Then
        Element = 1
        End If
'*******************************************************

Dim cDataX() As Double
Dim c1DataX() As Double
Dim cElement As Long
cDataX = DataX
cElement = Element
Kick = 200 ' 随便给个开始值 16-8-30
Dim LoopTimes As Long
LoopTimes = 0

'*********************
Dim i As Long
Dim Initial() As Double
ReDim Initial(1 To Element)
For i = 1 To Element
Initial(i) = i
Next
```

```
'*********************************************************************************
Dim P As Long
P = 0 ' 防止 P<>0 16-9-4
Do While Kick > 0

Dim rSS() As Double, rSSy As Double, rUy As Double, rSP() As Double, rSPy() As Double
Dim n As Long
n = 0
'TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation +
MultiElementLinearAnalysis_Equation2(cDataX, DataY, cElement, Return_Coefficient1D, Kick) +
Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation +
MultiElementLinearAnalysis_Equation4(cDataX, DataY, cElement, Return_Coefficient1D, Kick, rSS,
rSSy, rUy, rSP, rSPy) + Chr(13) + Chr(10)
'*******************************
Rem 16-9-6 一元线性分析，就不用下面的偏相关分析内容了，直接跳出
If Kick = -1 Then
Exit Function
End If
'*********************************
Rem 16-9-4 读取第一次的 Return_Coefficient1D
P = P + 1
Dim Old_Coefficient1D() As Double
Dim Old_SS() As Double
Dim Old_SSy As Double
Dim Old_Uy As Double
If (P = 1) Then
Old_Coefficient1D = Return_Coefficient1D
Old_SS = rSS
Old_SSy = rSSy
Old_Uy = rUy
End If

'*********************************
'For i = LBound(Old_SS) To UBound(Old_SS)
'Debug.Print "Old_SS=    " + CStr(Old_SS(i))
'Next



If Kick > 0 Then '从 K<>0 改成 K>0
LoopTimes = LoopTimes + 1 ' 记录循环次数，对后面更改数组数有帮助 16-8-30
c1DataX = cDataX 'cDataX 使用后就不能保存原来数据了，所以启用 c1DataX16-8-30
cElement = cElement - 1 '每次减少一列数
```

```
        '*************************
        Dim obj2 As New Array1D001
        Dim R3() As Double
        obj2.Array1D_RemoveOneNum Initial, Kick, R3
        'ReDim Preserve Initial(LBound(r) To UBound(r))
        Initial = R3

        'For i = LBound(r) To UBound(r)
        'Debug.Print Initial(i)
        'Next

        'Debug.Print "Kick=" & Kick
        'For i = LBound(Initial) To UBound(Initial)
        'Debug.Print "Initial(" & i & ")=" & Initial(i)
        'Next i

        'Dim CKick As Long
        'CKick = Kick
        'Debug.Print "kick=" & Kick
        'Do
        'If Initial(CKick) <> 0 Then
        'Initial(CKick) = 0
        'Debug.Print "Initial(" & CKick & ")=" & 0
        'Else
        'CKick = CKick + 1
        'End If
        'Loop While Initial(CKick) <> 0
        '*************************
        Rem 记录下去除的项,但是由于每次去除都是改变后的数组，所以不行 16-8-31
        'Dim KickNum() As Long
        'ReDim Preserve KickNum(LoopTimes)
        'KickNum(LoopTimes) = Kick
        '*************************

       Dim l As Long
       Dim Num As Long
       Num = UBound(DataX) - LBound(DataX) + 1
       ReDim cDataX(1 To Num, 1 To cElement) As Double

       n = 0 'n 的初始化
       TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "
剔除的数据是" + CStr(Kick) + "组数据" + Chr(13) + Chr(10)
       TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "
新的 X 数据如下" + Chr(13) + Chr(10)
```

```
        For l = 1 To (Element - LoopTimes + 1)
                If Kick <> l Then
                n = n + 1
                End If
                For i = 1 To Num
                   If Kick <> l Then
                   cDataX(i, n) = c1DataX(i, l)
                   'Debug.Print "cDatax(" & i & "," & n & ")=" & "c1datax(" & i & "," & L & ")" & "="
& c1DataX(i, L)

                   TheBestMultipleLinearRegressionEquation =
TheBestMultipleLinearRegressionEquation + "CDataX(" + CStr(i) + "," + CStr(n) + ")=" +
CStr(cDataX(i, n)) + Chr(13) + Chr(10)
                   End If
               Next i
        Next l
        n = 0
End If

Loop

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'If Kick = 0 Then
'TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation +
MultiElementLinearAnalysis_Equation3(cDataX, DataY, cElement, Return_Coefficient1D, Kick)
'Exit Function
'End if
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 偏回归系数



Dim PartialRegressionB() As Double
For i = LBound(Initial) To UBound(Initial)
'Debug.Print Initial(i)
ReDim Preserve PartialRegressionB(Initial(i))
PartialRegressionB(Initial(i)) = Old_Coefficient1D(Initial(i) + 1) * Sqr(Old_SS(Initial(i)) / rSSy)
'PartialRegressionB(Initial(i)) = Old_Coefficient1D(Initial(i) + 1) * Sqr(SS(Initial(i)) / SSy)
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + _
"标准偏回归系数 PartialRegressionB(" + CStr(Initial(i)) + ")=" + CStr(PartialRegressionB(Initial(i)))
+ Chr(13) + Chr(10)
Next i
'*************************************************************************
Rem 16-9-4
'第二节 复相关分析
```

```
'SSR = Uy
Dim R As Double
Dim SquareR As Double
SquareR = rUy / Old_SSy
R = Sqr(rUy / Old_SSy) ' 发现前面程序有使用 R 的情况，已经更改为 R3（），16-9-4
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "复相关分析~~~~~~~~~~~~~~~" + CStr(SquareR) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "相关指数（correlation index）R^2=" + CStr(SquareR) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "复相关系数（multiple  correlation coefficient）R=" + CStr(R) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "FR 和 F 是等价的，和数据的 F 检验相同" + Chr(13) + Chr(10)
'*第三节  偏相关分析
'*********************************************************************************
*********
'相关矩阵
Dim l As Long '16-9-19
'For i = 1 To Element - LoopTimes + 1
'Debug.Print " rSPy(" & i & ", 1)=" & rSPy(i, 1)
'Next i
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + CStr(SquareR) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "偏相关分析:" + Chr(13) + Chr(10)
Dim NewSP() As Double
ReDim NewSP(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1)
For i = 1 To Element - LoopTimes + 1
For l = 1 To Element - LoopTimes + 1
If (i <> (Element - LoopTimes + 1) And l <> (Element - LoopTimes + 1)) Then
NewSP(i, l) = rSP(i, l)
Else
'NewSP(i, l) = rSPy(1, 1)
If (i = Element - LoopTimes + 1 And l = Element - LoopTimes + 1) Then
NewSP(i, l) = rSSy
Exit For
End If
If (l = Element - LoopTimes + 1) Then
NewSP(i, l) = rSPy(i, 1)
End If
If (i = Element - LoopTimes + 1) Then
NewSP(i, l) = rSPy(l, 1)
End If
```

```
End If
Next l
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'For i = 1 To Element - LoopTimes + 1
'For l = 1 To Element - LoopTimes + 1
'Debug.Print "NewSP(" & i & ", " & l & ")=" & NewSP(i, l)
  'Debug.Print "rSP(" & i & "," & l; ")=" & rSP(i, l)
'Next l
'Next i

'For i = 1 To element - LoopTimes + 1
'For l = 1 To element - LoopTimes + 1
'Debug.Print "NewSP(" & i & ", " & l & ")=" & NewSP(i, l)
'Next l
'Next i

Dim NewSS() As Double
ReDim NewSS(1 To Element - LoopTimes + 1)
For i = 1 To Element - LoopTimes + 1
If (i <> Element - LoopTimes + 1) Then
NewSS(i) = rSS(i)
Else
NewSS(i) = rSSy '  是不是 SS4=SSY 这个还需要确认下
End If
Next i

Dim CorrelationMatrixR() As Double
ReDim CorrelationMatrixR(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1) '16-9-10
由 1 TO ELEMENT 改成 Element - LoopTimes + 1
For i = 1 To Element - LoopTimes + 1
For l = 1 To Element - LoopTimes + 1
CorrelationMatrixR(i, l) = NewSP(i, l) / Sqr(NewSS(i) * NewSS(l))
'Debug.Print "R(" + CStr(i) + "," + CStr(l) + ")=" + CStr(CorrelationMatrixR(i, l))
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "简单
相关系数 R(" + CStr(i) + "," + CStr(l) + ")=" + CStr(CorrelationMatrixR(i, l)) + Chr(13) + Chr(10)
Next l
Next i
Dim obj As New Matrix001
Dim InverseCorrelationMatrixR() As Double
obj.InverseMatrix CorrelationMatrixR, InverseCorrelationMatrixR

For i = 1 To Element - LoopTimes + 1
For l = 1 To Element - LoopTimes + 1
```

```
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "R^-1("
+ CStr(i) + "," + CStr(l) + ")=" + CStr(InverseCorrelationMatrixR(i, l)) + Chr(13) + Chr(10)
Next l
Next i

Dim PartialCorrelation() As Double
ReDim PartialCorrelation(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1) As Double
Dim Sr() As Double
ReDim Sr(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1) As Double
Dim t() As Double, T1() As Double
n = 0
ReDim t(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1)

'****************************************
Rem 16-9-19
Dim Num As Long
Num = UBound(DataX) - LBound(DataX) + 1
'****************************************

For i = 1 To Element - LoopTimes + 1
For l = 1 To Element - LoopTimes + 1
   If (i < l) Then
   If (l <> i) Then
    PartialCorrelation(i, l) = -InverseCorrelationMatrixR(l, i) / Sqr(InverseCorrelationMatrixR(i, i) *
InverseCorrelationMatrixR(l, l))
    TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "偏
相关系数 r(" + CStr(i) + "," + CStr(l) + ")=" + CStr(PartialCorrelation(i, l)) + Chr(13) + Chr(10)
    Sr(i, l) = Sqr((1 - PartialCorrelation(i, l) ^ 2) / (Num - (Element - LoopTimes + 1)))
    n = n + 1
    t(i, l) = PartialCorrelation(i, l) / Sr(i, l)
    ReDim Preserve T1(n)
    T1(n) = PartialCorrelation(i, l) / Sr(i, l)
    TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "t("
+ CStr(i) + "," + CStr(l) + ")=" + CStr(t(i, l)) + Chr(9) + "     对应 T" + CStr(n) + Chr(13) + Chr(10)
   End If
   End If
Next l
Next i

'****************************************
Rem  偏相关假设检验
Dim Return_T001 As Double, Return_T005 As Double
Dim obj4 As New T001
Rem 16-9-7 日改
```

```
Return_T001 = obj4.t(0.005, Num - (Element - LoopTimes + 1))
Return_T005 = obj4.t(0.025, Num - (Element - LoopTimes + 1))
'Return_T001 = obj4.t(0.01, Num - (Element - LoopTimes + 1))
'Return_T005 = obj4.t(0.05, Num - (Element - LoopTimes + 1))
'************************************************
Rem 16-9-7 参照田间统计，T 取绝度值
Dim T2() As Double
n = 0
For i = LBound(T1) To UBound(T1)
n = n + 1
ReDim Preserve T2(n)
T2(i) = Abs(T1(i))
Next i
'************************************************
Dim Return_TReport As String
For i = 1 To Element - LoopTimes + 1
   If (T2(i) > Return_T001) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "显著的回归关系" + "T" + CStr(i) + "="
+ CStr(T2(i)) + ">" + "T0.01=" + CStr(Return_T001) + Chr(13) + Chr(10)
   ElseIf (Return_T001 > T2(i) And T2(i) > Return_T005) Then
   Rem 下面没有验证
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "T" + CStr(i) + "=" + CStr(T2(i)) + ">" + "T0.05=" +
CStr(Return_T005) + Chr(13) + Chr(10)
   ElseIf (Return_T005 > T2(i)) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "无显著的回归关系" + "T0.05=" +
CStr(Return_T005) + ">" + "T" + CStr(i) + "=" + CStr(T2(i)) + Chr(13) + Chr(10)
   End If
Next i
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation + "T 检验
结果" + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation +
Return_TReport + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation = TheBestMultipleLinearRegressionEquation +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
End Function
```

Rem 16-9-7 废弃

Rem 三、一元二次多项式回归分析

```
'Public Function QuadraticPolyNomialWithOneVariableRegressionAnalysis(ByRef DataX() As
Double, ByRef DataY() As Double) As String
'Dim cDataX() As Double
'cDataX = DataX
'Dim Return_Coefficient1D() As Double
```

```
'Dim Kick As Long
'Kick = 200 ' 随便给个开始值 16-8-30
'Dim rSS() As Double, rSSy As Double, rUy As Double, rSP() As Double, rSPy() As Double
'QuadraticPolyNomialWithOneVariableRegressionAnalysis =
QuadraticPolyNomialWithOneVariableRegressionAnalysis +
MultiElementLinearAnalysis_Equation4(cDataX, DataY, 1, Return_Coefficient1D, Kick, rSS, rSSy,
rUy, rSP, rSPy) + Chr(13) + Chr(10)
'End Function
```

'MultiElementLinearAnalysis_Equation4 多元线性回归分析 4 号

Rem 16 -9 - 8

Rem MultiElementLinearAnalysis_Equation4 是在 MultiElementLinearAnalysis_Equation2 更改过后而再次更改的

Rem 没有更改 MultiElementLinearAnalysis_Equation3

Rem 16-9-10 N 阶行列式为 0 的时候，矩阵不可逆。也就是说有的数据会导致 N 阶行列式为 0，除数为 0 而程序出错

```
Public Function MultiElementLinearAnalysis_Equation4(ByRef DataX() As Double, _
ByRef DataY() As Double, ByVal Element As Long, ByRef Return_Coefficient1D() As Double, _
ByRef Kick As Long, ByRef Return_SS() As Double, ByRef Return_SSy As Double, ByRef Return_Uy
_
As Double, ByRef Return_SP() As Double, ByRef Return_SPy() As Double, ByRef R() As Double) As
String

'*******************************************************
Rem 16-9-12
        Dim obj1 As New FileWrite001
        Dim ArrayRange As Integer
        ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
        Element = UBound(DataX, 2) - LBound(DataX, 2) + 1
        End If
        If ArrayRange = 1 Then
        Element = 1
        End If
'*******************************************************
  Kick = 0 ' 初始化为 0 16-8-30
If (Element > 7) Then
MsgBox "大于 7 暂时不支持"
End If

       Dim Num As Long
```

```
        Dim i As Long
        Dim l As Long

        'For i = 1 To 24
        'Debug.Print DataY(i)
        'Next
        Num = UBound(DataY) - LBound(DataY) + 1


If (Element < 2) Then
    '**************************************
    Rem 16-8-30 增加的元素等于 1 的情况
    If (Element = 1) Then
    Dim cDataX() As Double
    ReDim cDataX(LBound(DataX) To UBound(DataX))
    'Dim i As Long
        'Dim obj1 As New FileWrite001
        'Dim ArrayRange As Integer
        'ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
              For i = LBound(DataX) To UBound(DataX)
              cDataX(i) = DataX(i, 1)
              Next
        Else
              If ArrayRange = 1 Then
              For i = LBound(DataX) To UBound(DataX)
              cDataX(i) = DataX(i)
              Next
              End If
        End If
    Dim ra As Double, rb As Double
    'MsgBox "请选用一元一次线性分析，或者 1 元多次多项式分析。"
    Rem             16-9-8 这个专门用来完成偏相关
    MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
LinearRegressionAnalysis_Equation(cDataX, DataY, ra, rb)
    Kick = -1 '16-9-6 用来表示 1 元线性分析
    Exit Function
    Else
    MsgBox "不支持或有错"
    Exit Function
    End If
End If
 '**************************************
```

```
Dim AR As Long
Dim obj11 As New FileWrite001
AR = obj11.ArrayRangeD(DataX())

If (AR = 1) Then
          MsgBox "数据为一维数据  And Element<>1  无法计算！ "
          Exit Function ' 16-9-8
Else
       If (AR <> 2) Then
       MsgBox "数组维度不正确"
       Exit Function
       End If
End If
'*****************************
Rem 16-9-8
If AR = 2 Then
Dim cDataX1() As Double
cDataX1 = DataX
End If
'*****************************
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"***~~~~~~~~~~~~~~开始多元线性分析~~~~~~~~~~~~~~~~***" + Chr(13) + Chr(10)
'~~~~~~~~~~~~~~~if ar=2 开始~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
If AR = 2 Then
Rem  求一级数据
Dim PartX() As Double
Dim PartXSquare() As Double
ReDim PartX(1 To Element)
ReDim PartXSquare(1 To Element)
For i = 1 To Element
For l = 1 To Num
PartX(i) = PartX(i) + DataX(l, i)
PartXSquare(i) = PartXSquare(i) + DataX(l, i) ^ 2
Next l
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "X" + CStr(i) +
"项累加                              =" + CStr(PartX(i)) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "X" + CStr(i) +
"项平均值                           =" + CStr(PartX(i) / Num) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "X" + CStr(i) +
"项平方累加                    =" + CStr(PartXSquare(i)) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "X" + CStr(i) +
"项平方累加平均值=" + CStr(PartXSquare(i) / Num) + Chr(13) + Chr(10)
Next i
  MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
```

```
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************
Rem 前题是 Y 项和 X 项数量是相同的
Dim PartY As Double
Dim PartYSquare As Double
For l = 1 To Num
PartY = PartY + DataY(l)
PartYSquare = PartYSquare + DataY(l) ^ 2
Next l
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "Y 项累加
=" + CStr(PartY) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "Y 项平均值
=" + CStr(PartY / Num) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "Y 项平方累
加=" + CStr(PartYSquare) + Chr(13) + Chr(10)
  MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************************
Rem 求 SSi 的值
'Dim ss() As Double
ReDim SS(1 To Element) As Double
For i = 1 To Element
For l = 1 To Num
SS(i) = SS(i) + (DataX(l, i) - PartX(i) / Num) ^ 2
ReDim Preserve Return_SS(i)
Return_SS(i) = SS(i)
Next l
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "SS" + CStr(i)
+ "=" + CStr(SS(i)) + Chr(13) + Chr(10)
Next i
  MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'***************************************
Rem 求 SSy 的值
Dim SSy As Double
For i = 1 To Num
SSy = SSy + (DataY(i) - PartY / Num) ^ 2
Next i
Return_SSy = SSy
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "y 变量的总
平方和 SSy" + "=" + CStr(SSy) + Chr(13) + Chr(10)
  MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'********************************************************************************
```

```
Rem 16-9-19
Dim SS1() As Double
ReDim SS1(1 To Element + 1)
For i = 1 To Element
SS1(i) = SS(i)
Next
SS1(Element + 1) = SSy

'******************************************************************************
Rem  求 SPik 的值
Rem  田间统计分析书上测试已经通过（16-8-24）
Dim n As Long, m As Long
Dim SP() As Double
ReDim SP(1 To Element, 1 To Element) As Double
ReDim Return_SP(1 To Element, 1 To Element) As Double
For i = 1 To Element
For l = 1 To Element
    '*********************************
    Rem 16-9-5
    For m = 1 To Num
    Return_SP(i, l) = Return_SP(i, l) + (DataX(m, i) - PartX(i) / Num) * (DataX(m, l) - PartX(l) / Num)
    Next m
    MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"Return_SP(" + CStr(i) + "," + CStr(l) + ")=" + CStr(Return_SP(i, l)) + Chr(13) + Chr(10)
    '*********************************
    If (i <> l) Then
    If (i < l) Then

        For n = 1 To Num
        SP(i, l) = SP(i, l) + (DataX(n, i) - PartX(i) / Num) * (DataX(n, l) - PartX(l) / Num)
        Next n
        MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "SP(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i, l)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i, l))
    End If
    End If
Next l
Next i
     MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)

'*****************************************************************************
    Rem  求 SPi0 的值
    Dim SP0() As Double
```

```
    ReDim SP0(1 To Element) As Double
    For i = 1 To Element
        For n = 1 To Num
        SP0(i) = SP0(i) + (DataX(n, i) - PartX(i) / Num) * (DataY(n) - PartY / Num)
        Next n
        MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "SP0("
+ CStr(i) + ")=" + CStr(SP0(i)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(L) + ")=" + CStr(SP(i, L))
    Next i
    MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
    '
***************************************************************************
Dim a() As Double
ReDim a(1 To Element, 1 To Element)
For i = 1 To Element
For l = 1 To Element
    If (i <> l) Then
    If (i < l) Then
    SP(l, i) = SP(i, l)
    End If
    End If
Next l
Next i
For i = 1 To Element
For l = 1 To Element
   If i = l Then
   a(i, l) = SS(i)
   Else
   a(i, l) = SP(i, l)
   End If
   MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "A(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(a(i, l)) + Chr(13) + Chr(10)
  'Debug.Print "A(" + CStr(i) + "," + CStr(l) + ")=" + CStr(A(i, l))
Next l
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim obj3 As New Matrix001
Dim InverseA() As Double
obj3.InverseMatrix a, InverseA
For i = 1 To Element
For l = 1 To Element
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "InverseA(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(InverseA(i, l)) + Chr(13) + Chr(10)
```

```
Next l
Next i
'*******************************************************************************
'准备放系数求解
Rem 16-9-7
        Dim b1() As Double
        Dim SP01() As Double
        ReDim SP01(LBound(SP0) To UBound(SP0), 1)
        For i = LBound(SP0) To UBound(SP0)
        SP01(i, 1) = SP0(i)
        'Debug.Print "SP01=" & SP01(i, 1)
        Next i
        obj3.MatrixMultiplication InverseA, SP01, b1
        'For i = 1 To Element
        'Debug.Print "b1=" & b1(i, 1)
        'Next
        Dim b2() As Double
        ReDim b2(1 To Element + 1)
        Dim Part1 As Double
        For i = 1 To Element + 1
        If i = Element + 1 Then
               b2(1) = PartY / Num - Part1
        Else
             Part1 = Part1 + b1(i, 1) * PartX(i) / Num
             b2(i + 1) = b1(i, 1)
        End If
        Next i
        'For i = 1 To Element + 1
        'Debug.Print "b2=" & b2(i)
        'Next

        For i = LBound(b2) To UBound(b2)
        ReDim Preserve Return_Coefficient1D(i)
        Return_Coefficient1D(i) = b2(i)
        MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + CStr(i -
1) + "次项系数=" + CStr(Return_Coefficient1D(i)) + Chr(13) + Chr(10)
        Next i

        '****************************************
          MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
        Dim Return_Fitting() As Double
        ReDim Return_Fitting(1 To Num)
        For l = 1 To Num
```

```
        For i = 1 To Element
        Return_Fitting(l) = Return_Fitting(l) + Return_Coefficient1D(i + 1) * cDataX1(l, i)
        Next i
        Return_Fitting(l) = Return_Fitting(l) + Return_Coefficient1D(1)
        MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + CStr(l) +
"项拟合数=" + CStr(Return_Fitting(l)) + Chr(13) + Chr(10)
        Next l
        MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
        '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Rem  离回归标准误
    Dim Se As Double
    Dim Uy As Double
    Dim Qy As Double
    'For i = 1 To Num    '  等价于下面的 Uy
    ' Uy = Uy + (Return_Fitting(i) - PartY / Num) ^ 2
    'Next i
    For i = 1 To Element
    Uy = Uy + Return_Coefficient1D(i + 1) * SP0(i)
    Next i
    Return_Uy = Uy
    Qy = SSy - Uy
End If     'if ar=2  是这个的 END,放这里正确否？16-9-4!!!!!!!!!!!!!!!!!!!!
'~~~~~~~~~~~~~~~~~~~~~~~~~~~if ar=2  结束~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Se = Sqr(Qy / (Num - Element - 1))
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "估计标准误
Se=" + CStr(Se) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "多元离回归
平方和 Qy=SSr=" + CStr(Qy) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "M 元回归平
方和 Uy=SSR=" + CStr(Uy) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)


'*******************************************************************************
*********
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~多元线性显著性检验~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim SSr As Double
Dim SS_r As Double
Dim DFy As Double
Dim DFR As Double
Dim DF_r As Double
```

```
Dim MSR As Double
Dim MS_r As Double
SSr = Uy
SS_r = Qy
DFy = Num - 1
DFR = Element
DF_r = Num - Element - 1
MSR = SSr / DFR
MS_r = SS_r / DF_r
Dim F As Double

    If SS_r = 0 Then
    MsgBox "完全重合"
    F = 999999999
    Else
    F = MSR / MS_r ' 16-9-6  更改到 IF 内
    End If

MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "变异原因
DF                              SS                              MS                              F" + Chr(13) +
Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "多元回归
" + CStr(DFR) + "          " + Chr(9) + CStr(SSr) + "          " + Chr(9) + CStr(MSR) + "          " + Chr(9) +
CStr(F) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "离回归
" + CStr(DF_r) + "          " + Chr(9) + CStr(SS_r) + "          " + Chr(9) + CStr(MS_r) + "          " + Chr(13)
+ Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "总和
" + CStr(DFy) + "          " + Chr(9) + CStr(SSy) + Chr(13) + Chr(10)
'*****************************************************************
'Dim obj As New F001
'Return_F001 = obj.PFaN1N2X2(0.01, 0.01, Return_dfR, Return_df_r)
'Return_F005 = obj.PFaN1N2X2(0.05, 0.01, Return_dfR, Return_df_r)
Rem  当多元线性回归关系经显著检验为显著时，还必须逐一对各偏回归系数进行显著性检
验，发现和剔除不显著的偏回归关系对应的自变量。
Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj2 As New F001
Return_F001 = obj2.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj2.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万
Dim Return_FReport As String

If (F > Return_F001) Then
```

```
Return_FReport = Return_FReport + "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" + CStr(Return_F001)
ElseIf (Return_F001 > F And F > Return_F005) Then
Rem 下面没有验证
Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + Chr(13) + Chr(10)
Return_FReport = Return_FReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_F005)
ElseIf (Return_F005 > F) Then
Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">" + "F=" + CStr(F)
End If
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "F 检验结果" + Return_FReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************************************************************
Rem T 检验

MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "偏回归系数的检验~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "T 检验:" + Chr(13) + Chr(10)

'*****************************************
Rem 16-9-5
Dim b() As Double
Dim k() As Double
ReDim b(1 To Element, 1)
For i = 1 To Element
b(i, 1) = Return_Coefficient1D(i + 1)
Next i
obj3.MatrixMultiplication a, b, k
For i = 1 To Element
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "K(" + CStr(i) + ", " + CStr(1) + ") = " + "SP(" + CStr(i) + ",y)=" + CStr(k(i, 1)) + Chr(13) + Chr(10)
Next i
Return_SPy = k
'*****************************************
Dim c() As Double
ReDim c(1 To Element)
For i = 1 To Element
For l = 1 To Element
  If i = l Then
  c(i) = InverseA(i, l)
```

```
   MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "C(" + CStr(i)
+ ")=" + CStr(c(i)) + Chr(13) + Chr(10)
   End If
Next l
Next i

Dim Sb() As Double
ReDim Sb(1 To Element)
Dim t() As Double
ReDim t(1 To Element)

For i = 1 To Element
Sb(i) = Se * Sqr(c(i))
    If Sb(i) = 0 Then
    Rem 这个有没有用还要验证 16-8-28
    MsgBox "完全重合"
    t(i) = 999999999
    Else
    t(i) = Return_Coefficient1D(i + 1) / Sb(i)
    MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "t(" +
CStr(i) + ")=" + CStr(t(i)) + Chr(13) + Chr(10)
    End If
Next i
Dim Return_T001 As Double, Return_T005 As Double
Dim obj4 As New T001
Return_T001 = obj4.t(0.005, DF_r) '16-9-7 日改
Return_T005 = obj4.t(0.025, DF_r)
Dim Return_TReport As String
'************************************************
Rem 16-9-7 参照田间统计，T 取绝度值
Dim T1() As Double
n = 0
For i = LBound(t) To UBound(t)
n = n + 1
ReDim Preserve T1(n)
T1(i) = Abs(t(i))
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "t(" + CStr(i) +
")=" + CStr(t(i)) + Chr(13) + Chr(10)
Next i
n = 0
'************************************************
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "以下 T 为绝
对值" + Chr(13) + Chr(10)
For i = 1 To Element
```

```
   If (T1(i) > Return_T001) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "显著的回归关系" + "T" + CStr(i) + "="
+ CStr(T1(i)) + ">" + "T0.01=" + CStr(Return_T001) + Chr(13) + Chr(10)
   ElseIf (Return_T001 > T1(i) And T1(i) > Return_T005) Then
   Rem 下面没有验证
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "T" + CStr(i) + "=" + CStr(T1(i)) + ">" + "T0.05=" +
CStr(Return_T005) + Chr(13) + Chr(10)
   ElseIf (Return_T005 > T1(i)) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "无显著的回归关系" + "T0.05=" +
CStr(Return_T005) + ">" + "T" + CStr(i) + "=" + CStr(T1(i)) + Chr(13) + Chr(10)
   End If
Next i
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "T 检验结果"
+ Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
Return_TReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************************************************************
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "F 检验:" +
Chr(13) + Chr(10)
Rem F 检验
Dim Up() As Double
ReDim Up(1 To Element)
For i = 1 To Element
Up(i) = Return_Coefficient1D(i + 1) ^ 2 / c(i)
Next i
Dim FF() As Double
ReDim FF(1 To Element)
For i = 1 To Element
    If Qy = 0 Then
    MsgBox "完全重合"
    FF(i) = 999999999
    Else
    FF(i) = Up(i) / (Qy / DF_r)
    End If
    MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "F" +
CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
Next i

MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "变异原因
DF                    SS                    MS                    F" + Chr(13) +
Chr(10)
```

```
For i = 1 To Element
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "x" + CStr(i) + "的偏回归" + "          " + CStr(1) + "          " + Chr(9) + CStr(Up(i)) + "          " + Chr(9) + CStr(Up(i)) + "          " + Chr(9) + CStr(FF(i)) + "          " + Chr(13) + Chr(10)
Next i
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "离回归" + CStr(DF_r) + "          " + Chr(9) + CStr(SS_r) + "          " + Chr(9) + CStr(MS_r) + Chr(13) + Chr(10)

Dim Return_FP001 As Double
Dim Return_FP005 As Double
Return_FP001 = obj2.PFaN1N2X2(0.01, 10000000, 1, DF_r) ' 0.01 用千万
Return_FP005 = obj2.PFaN1N2X2(0.05, 1000000, 1, DF_r)     ' 0.05 用百万
'Dim Return_FPReport As String
'For i = 1 To Element
'If (FF(i) > Return_FP001) Then
'Return_FPReport = "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" + CStr(Return_FP001)
'ElseIf (Return_FP001 > F And F > Return_FP005) Then
Rem  下面没有验证
'Return_FPReport = "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + Chr(13) + Chr(10)
'Return_FPReport = Return_FPReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_FP005)
'ElseIf (Return_FP005 > F) Then
'Return_FPReport = "无显著的回归关系" + "F0.05=" + CStr(Return_FP005) + ">" + "F=" + CStr(F)
'End If
'Next
'MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "F 检验结果" + Return_FPReport + Chr(13) + Chr(10)
'MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim Remove() As Double '  为剔除 Upi 值
Dim NumRemove As Long

Dim Return_FPReport As String
For i = 1 To Element
   If (FF(i) > Return_FP001) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "显著的回归关系" + "FF" + CStr(i) + "=" + CStr(FF(i)) + ">" + "F0.01=" + CStr(Return_FP001) + Chr(13) + Chr(10)
   ElseIf (Return_FP001 > FF(i) And FF(i) > Return_FP005) Then
   Rem  下面没有验证
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + "FF" + CStr(i) + "=" + CStr(FF(i)) + ">" + "F0.05=" + CStr(Return_FP005) + Chr(13) + Chr(10)
   ElseIf (Return_FP005 > FF(i)) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "无显著的回归关系" + "F0.05=" +
```

```
CStr(Return_FP005) + ">" + "FF" + CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
'****************************************


   NumRemove = NumRemove + 1
   ReDim Preserve Remove(NumRemove)
   Remove(NumRemove) = i
'****************************************
   End If
Next i
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "F 检验结果"
+ Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
Return_FPReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)



'*************************
Rem 非必要程序没有去验证，来源于田间试验与统计分析
Dim SumUp As Double
For i = 1 To Element
SumUp = SumUp + Up(i)
Next i

l = 0

If Uy = SumUp Then
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "各自变量独
立不相关" + Chr(13) + Chr(10)
Else
        For i = 1 To Element
           If Uy > SumUp And Return_Coefficient1D(i + 1) > 0 Then
           l = l + 1
           End If
        Next i

        If l = 3 Then
        MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "变量
之间正相关" + Chr(13) + Chr(10)
        End If
End If
l = 0
'*************************************************************************
Dim UpArray() As Double
```

```
Dim rlu() As Double, rul() As Double
Dim Which() As Long
Dim obj5 As New Array1D001
n = 0

If NumRemove >= 1 Then
    For i = LBound(Remove) To UBound(Remove)
    n = n + 1
    ReDim Preserve UpArray(n)
    UpArray(n) = Up(Remove(i))
    Next i
    n = 0

    For i = LBound(UpArray) To UBound(UpArray)
    MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 +
"UpArray(" + CStr(i) + ")=" + CStr(UpArray(i)) + Chr(13) + Chr(10)
    Next

    Dim MixNum As Double
    MixNum = obj5.Array1D_Mix(UpArray) ' 如果有两个数据一样，会给出一个 16-8-30
    For i = LBound(Up) To UBound(Up)
      If MixNum = Up(i) Then ' 如果有两个数据一样，会给出前面的 16-8-30
      Kick = i
      MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "需要
剔除的组数是" + CStr(Kick) + Chr(13) + Chr(10)
      End If
    Next i

End If
'*****************************************************************
ReDim R(1 To Element + 1, 1 To Element + 1)
For i = 1 To Element + 1
For l = 1 To Element + 1
If l = Element + 1 Or i = Element + 1 Then
    If l = Element + 1 And i = Element + 1 Then
    R(i, l) = 1
    Else
    If l = Element + 1 Then
    R(i, l) = SP0(i) / Sqr(SS1(i) * SS1(l))
    End If
    If i = Element + 1 Then
    R(i, l) = SP0(l) / Sqr(SS1(i) * SS1(l))
    End If
    End If
```

```
Else
R(i, l) = Return_SP(i, l) / Sqr(SS1(i) * SS1(l))
End If
MultiElementLinearAnalysis_Equation4 = MultiElementLinearAnalysis_Equation4 + "简单相关系
数 r(" + CStr(i) + "," + CStr(l) + ")=" + CStr(R(i, l)) + Chr(13) + Chr(10)
Next l
Next i
End Function
```

'MultiElementLinearAnalysis_Equation5 多元线性回归分析 5 号

Rem 16-9-14 用于通径分析，暂时没有用

```
Public Function MultiElementLinearAnalysis_Equation5(ByRef DataX() As Double, _
ByRef DataY() As Double, ByVal Element As Long, ByRef Return_Coefficient1D() As Double, _
ByRef Kick As Long, ByRef Return_SS() As Double, ByRef Return_SSy As Double, ByRef Return_Uy
_
As Double, ByRef Return_SP() As Double, ByRef Return_SPy() As Double, ByRef
Return_MeanPartX() As Double, _
ByRef Return_MeanPartY As Double) As String

'*********************************************************
Rem 16-9-12
        Dim obj1 As New FileWrite001
        Dim ArrayRange As Integer
        ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
        Element = UBound(DataX, 2) - LBound(DataX, 2) + 1
        End If
        If ArrayRange = 1 Then
        Element = 1
        End If
'*********************************************************
  Kick = 0 ' 初始化为 0 16-8-30
If (Element > 7) Then
MsgBox "大于 7 暂时不支持"
End If

      Dim Num As Long
      Dim i As Long
      Dim l As Long

      'For i = 1 To 24
      'Debug.Print DataY(i)
```

```
        'Next
        Num = UBound(DataY) - LBound(DataY) + 1


If (Element < 2) Then
    '*************************************
    Rem 16-8-30  增加的元素等于 1 的情况
    If (Element = 1) Then
    Dim cDataX() As Double
    ReDim cDataX(LBound(DataX) To UBound(DataX))
    'Dim i As Long
        'Dim obj1 As New FileWrite001
        'Dim ArrayRange As Integer
        'ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
            For i = LBound(DataX) To UBound(DataX)
            cDataX(i) = DataX(i, 1)
            Next
        Else
            If ArrayRange = 1 Then
            For i = LBound(DataX) To UBound(DataX)
            cDataX(i) = DataX(i)
            Next
            End If
        End If
    Dim ra As Double, rb As Double
    'MsgBox "请选用一元一次线性分析，或者 1 元多次多项式分析。"
    Rem             16-9-8  这个专门用来完成偏相关
    MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
LinearRegressionAnalysis_Equation(cDataX, DataY, ra, rb)
    Kick = -1 '16-9-6  用来表示 1 元线性分析
    Exit Function
    Else
    MsgBox "不支持或有错"
    Exit Function
    End If
End If
 '*************************************


Dim AR As Long
Dim obj11 As New FileWrite001
AR = obj11.ArrayRangeD(DataX())

If (AR = 1) Then
```

```
            MsgBox "数据为一维数据  And Element<>1  无法计算！"
            Exit Function ' 16-9-8
Else
        If (AR <> 2) Then
        MsgBox "数组维度不正确"
        Exit Function
        End If
End If
'******************************
Rem 16-9-8
If AR = 2 Then
Dim cDataX1() As Double
cDataX1 = DataX
End If
'******************************
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"***~~~~~~~~~~~~~开始多元线性分析~~~~~~~~~~~~~~~***" + Chr(13) + Chr(10)
'~~~~~~~~~~~~~~if ar=2 开始~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
If AR = 2 Then
Rem  求一级数据
Dim PartX() As Double
Dim PartXSquare() As Double
Dim Return_MeanPartX() As Double

ReDim PartX(1 To Element)
ReDim PartXSquare(1 To Element)
For i = 1 To Element
For l = 1 To Num
PartX(i) = PartX(i) + DataX(l, i)
PartXSquare(i) = PartXSquare(i) + DataX(l, i) ^ 2
Next l
ReDim Return_MeanPartX(i)
Return_MeanPartX(i) = PartX(i) / Num
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "X" + CStr(i) +
"项累加                    =" + CStr(PartX(i)) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "X" + CStr(i) +
"项平均值                  =" + CStr(PartX(i) / Num) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "X" + CStr(i) +
"项平方累加              =" + CStr(PartXSquare(i)) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "X" + CStr(i) +
"项平方累加平均值=" + CStr(PartXSquare(i) / Num) + Chr(13) + Chr(10)
Next i
  MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
```

```
'*******************************
Rem 前题是 Y 项和 X 项数量是相同的
Dim PartY As Double
Dim PartYSquare As Double
Dim Return_MeanPartY As Double
For l = 1 To Num
PartY = PartY + DataY(l)
PartYSquare = PartYSquare + DataY(l) ^ 2
Next l
Return_MeanPartY = PartY / Num
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "Y 项累加
=" + CStr(PartY) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "Y 项平均值
=" + CStr(PartY / Num) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "Y 项平方累
加=" + CStr(PartYSquare) + Chr(13) + Chr(10)
  MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************
Rem 求 SSi 的值
'Dim ss() As Double
ReDim SS(1 To Element) As Double
For i = 1 To Element
For l = 1 To Num
SS(i) = SS(i) + (DataX(l, i) - PartX(i) / Num) ^ 2
ReDim Preserve Return_SS(i)
Return_SS(i) = SS(i)
Next l
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "SS" + CStr(i)
+ "=" + CStr(SS(i)) + Chr(13) + Chr(10)
Next i
  MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************
Rem 求 SSy 的值
Dim SSy As Double
For i = 1 To Num
SSy = SSy + (DataY(i) - PartY / Num) ^ 2
Next i
Return_SSy = SSy
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "y 变量的总
平方和 SSy" + "=" + CStr(SSy) + Chr(13) + Chr(10)
  MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
```

```
'*************************************************************************
Rem 16-9-19
Dim SS1() As Double
ReDim SS1(1 To Element + 1)
For i = 1 To Element
SS1(i) = SS(i)
Next
SS1(Element + 1) = SSy

'*****************************

Rem  求 SPik 的值
Rem  田间统计分析书上测试已经通过（16-8-24）
Dim n As Long, m As Long
Dim SP() As Double
ReDim SP(1 To Element, 1 To Element) As Double
ReDim Return_SP(1 To Element, 1 To Element) As Double
For i = 1 To Element
For l = 1 To Element
      '*********************************
    Rem 16-9-5
    Rem 16-9-19 Return_SP 看是否包含 SPy
    For m = 1 To Num
    Return_SP(i, l) = Return_SP(i, l) + (DataX(m, i) - PartX(i) / Num) * (DataX(m, l) - PartX(l) / Num)
    MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"Return_SP(" + CStr(i) + "," + CStr(l) + ")=" + CStr(Return_SP(i, l)) + Chr(13) + Chr(10)
    Next m
    '*********************************
    If (i <> l) Then
    If (i < l) Then

        For n = 1 To Num
        SP(i, l) = SP(i, l) + (DataX(n, i) - PartX(i) / Num) * (DataX(n, l) - PartX(l) / Num)
        Next n
        MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "SP(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i, l)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i, l))
    End If
    End If
Next l
Next i
      MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
```

```
'*****************************************************************************
    Rem  求 SPi0 的值
    Dim SP0() As Double
    ReDim SP0(1 To Element) As Double
    For i = 1 To Element
        For n = 1 To Num
        SP0(i) = SP0(i) + (DataX(n, i) - PartX(i) / Num) * (DataY(n) - PartY / Num)
        Next n
        MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "SP0("
+ CStr(i) + ")=" + CStr(SP0(i)) + Chr(13) + Chr(10)
        'Debug.Print "SP(" + CStr(i) + "," + CStr(L) + ")=" + CStr(SP(i, L))
    Next i
    MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
    '
*****************************************************************************
Dim a() As Double
ReDim a(1 To Element, 1 To Element)
For i = 1 To Element
For l = 1 To Element
    If (i <> l) Then
    If (i < l) Then
    SP(l, i) = SP(i, l)
    End If
    End If
Next l
Next i
For i = 1 To Element
For l = 1 To Element
   If i = l Then
   a(i, l) = SS(i)
   Else
   a(i, l) = SP(i, l)
   End If
   MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "A(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(a(i, l)) + Chr(13) + Chr(10)
  'Debug.Print "A(" + CStr(i) + "," + CStr(l) + ")=" + CStr(A(i, l))
Next l
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim obj3 As New Matrix001
Dim InverseA() As Double
obj3.InverseMatrix a, InverseA
For i = 1 To Element
```

```
For l = 1 To Element
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "InverseA(" +
CStr(i) + "," + CStr(l) + ")=" + CStr(InverseA(i, l)) + Chr(13) + Chr(10)
Next l
Next i
'*****************************************************************************
'准备放系数求解
Rem 16-9-7
        Dim b1() As Double
        Dim SP01() As Double
        ReDim SP01(LBound(SP0) To UBound(SP0), 1)
        For i = LBound(SP0) To UBound(SP0)
        SP01(i, 1) = SP0(i)
        'Debug.Print "SP01=" & SP01(i, 1)
        Next i
        obj3.MatrixMultiplication InverseA, SP01, b1
        'For i = 1 To Element
        'Debug.Print "b1=" & b1(i, 1)
        'Next
        Dim b2() As Double
        ReDim b2(1 To Element + 1)
        Dim Part1 As Double
        For i = 1 To Element + 1
        If i = Element + 1 Then
               b2(1) = PartY / Num - Part1
        Else
             Part1 = Part1 + b1(i, 1) * PartX(i) / Num
             b2(i + 1) = b1(i, 1)
        End If
        Next i
        'For i = 1 To Element + 1
        'Debug.Print "b2=" & b2(i)
        'Next

        For i = LBound(b2) To UBound(b2)
        ReDim Preserve Return_Coefficient1D(i)
        Return_Coefficient1D(i) = b2(i)
        MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + CStr(i -
1) + "次项系数=" + CStr(Return_Coefficient1D(i)) + Chr(13) + Chr(10)
        Next i

        '****************************************
          MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
```

```
        Dim Return_Fitting() As Double
        ReDim Return_Fitting(1 To Num)
        For l = 1 To Num
        For i = 1 To Element
        Return_Fitting(l) = Return_Fitting(l) + Return_Coefficient1D(i + 1) * cDataX1(l, i)
        Next i
        Return_Fitting(l) = Return_Fitting(l) + Return_Coefficient1D(1)
        MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + CStr(l) +
"项拟合数=" + CStr(Return_Fitting(l)) + Chr(13) + Chr(10)
        Next l
        MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
        '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Rem  离回归标准误
    Dim Se As Double
    Dim Uy As Double
    Dim Qy As Double
    'For i = 1 To Num    '  等价于下面的 Uy
    ' Uy = Uy + (Return_Fitting(i) - PartY / Num) ^ 2
    'Next i
    For i = 1 To Element
    Uy = Uy + Return_Coefficient1D(i + 1) * SP0(i)
    Next i
    Return_Uy = Uy
    Qy = SSy - Uy
End If     'if ar=2  是这个的 END,放这里正确否？16-9-4!!!!!!!!!!!!!!!!!!!!
'~~~~~~~~~~~~~~~~~~~~~if ar=2  结束~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Se = Sqr(Qy / (Num - Element - 1))
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "估计标准误
Se=" + CStr(Se) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "多元离回归
平方和 Qy=SSr=" + CStr(Qy) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "M 元回归平
方和 Uy=SSR=" + CStr(Uy) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)


'*****************************************************************************
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~多元线性显著性检验~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim SSr As Double
Dim SS_r As Double
Dim DFy As Double
```

```
Dim DFR As Double
Dim DF_r As Double
Dim MSR As Double
Dim MS_r As Double
SSr = Uy
SS_r = Qy
DFy = Num - 1
DFR = Element
DF_r = Num - Element - 1
MSR = SSr / DFR
MS_r = SS_r / DF_r
Dim F As Double

    If SS_r = 0 Then
    MsgBox "完全重合"
    F = 999999999
    Else
    F = MSR / MS_r ' 16-9-6  更改到 IF 内
    End If

MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "变异原因
DF                              SS                              MS                              F" + Chr(13) +
Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "多元回归
" + CStr(DFR) + "          " + Chr(9) + CStr(SSr) + "          " + Chr(9) + CStr(MSR) + "          " + Chr(9) +
CStr(F) + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "离回归
" + CStr(DF_r) + "          " + Chr(9) + CStr(SS_r) + "          " + Chr(9) + CStr(MS_r) + "          " + Chr(13)
+ Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "总和
" + CStr(DFy) + "          " + Chr(9) + CStr(SSy) + Chr(13) + Chr(10)
'*****************************************************************
'Dim obj As New F001
'Return_F001 = obj.PFaN1N2X2(0.01, 0.01, Return_dfR, Return_df_r)
'Return_F005 = obj.PFaN1N2X2(0.05, 0.01, Return_dfR, Return_df_r)
Rem  当多元线性回归关系经显著检验为显著时，还必须逐一对各偏回归系数进行显著性检
验，发现和剔除不显著的偏回归关系对应的自变量。
Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj2 As New F001
Return_F001 = obj2.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj2.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万
Dim Return_FReport As String
```

```
If (F > Return_F001) Then
Return_FReport = Return_FReport + "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" +
CStr(Return_F001)
ElseIf (Return_F001 > F And F > Return_F005) Then
Rem 下面没有验证
Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归
关系" + Chr(13) + Chr(10)
Return_FReport = Return_FReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_F005)
ElseIf (Return_F005 > F) Then
Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">"
+ "F=" + CStr(F)
End If
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "F 检验结果"
+ Return_FReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************************************************************
Rem T 检验

MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "偏回归系数
的检验~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "T 检验:" +
Chr(13) + Chr(10)

'*****************************************
Rem 16-9-5
Dim b() As Double
Dim k() As Double
ReDim b(1 To Element, 1)
For i = 1 To Element
b(i, 1) = Return_Coefficient1D(i + 1)
Next i
obj3.MatrixMultiplication a, b, k
For i = 1 To Element
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "K(" + CStr(i)
+ ", " + CStr(1) + ") = " + "SP(" + CStr(i) + ",y)=" + CStr(k(i, 1)) + Chr(13) + Chr(10)
Next i
Return_SPy = k
'*****************************************
Dim c() As Double
ReDim c(1 To Element)
For i = 1 To Element
For l = 1 To Element
```

```
    If i = l Then
    c(i) = InverseA(i, l)
    MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "C(" + CStr(i)
+ ")=" + CStr(c(i)) + Chr(13) + Chr(10)
    End If
Next l
Next i

Dim Sb() As Double
ReDim Sb(1 To Element)
Dim t() As Double
ReDim t(1 To Element)

For i = 1 To Element
Sb(i) = Se * Sqr(c(i))
    If Sb(i) = 0 Then
    Rem 这个有没有用还要验证 16-8-28
    MsgBox "完全重合"
    t(i) = 999999999
    Else
    t(i) = Return_Coefficient1D(i + 1) / Sb(i)
    MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "t(" +
CStr(i) + ")=" + CStr(t(i)) + Chr(13) + Chr(10)
    End If
Next i
Dim Return_T001 As Double, Return_T005 As Double
Dim obj4 As New T001
Return_T001 = obj4.t(0.005, DF_r) '16-9-7 日改
Return_T005 = obj4.t(0.025, DF_r)
Dim Return_TReport As String
'************************************************
Rem 16-9-7 参照田间统计，T 取绝度值
Dim T1() As Double
n = 0
For i = LBound(t) To UBound(t)
n = n + 1
ReDim Preserve T1(n)
T1(i) = Abs(t(i))
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "t(" + CStr(i) +
")=" + CStr(t(i)) + Chr(13) + Chr(10)
Next i
n = 0
'************************************************
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "以下 T 为绝
```

```
对值" + Chr(13) + Chr(10)
For i = 1 To Element
   If (T1(i) > Return_T001) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "显著的回归关系" + "T" + CStr(i) + "="
+ CStr(T1(i)) + ">" + "T0.01=" + CStr(Return_T001) + Chr(13) + Chr(10)
   ElseIf (Return_T001 > T1(i) And T1(i) > Return_T005) Then
   Rem 下面没有验证
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "T" + CStr(i) + "=" + CStr(T1(i)) + ">" + "T0.05=" +
CStr(Return_T005) + Chr(13) + Chr(10)
   ElseIf (Return_T005 > T1(i)) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "无显著的回归关系" + "T0.05=" +
CStr(Return_T005) + ">" + "T" + CStr(i) + "=" + CStr(T1(i)) + Chr(13) + Chr(10)
   End If
Next i
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "T 检验结果"
+ Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
Return_TReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'********************************************************************************
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "F 检验:" +
Chr(13) + Chr(10)
Rem F 检验
Dim Up() As Double
ReDim Up(1 To Element)
For i = 1 To Element
Up(i) = Return_Coefficient1D(i + 1) ^ 2 / c(i)
Next i
Dim FF() As Double
ReDim FF(1 To Element)
For i = 1 To Element
    If Qy = 0 Then
    MsgBox "完全重合"
    FF(i) = 999999999
    Else
    FF(i) = Up(i) / (Qy / DF_r)
    End If
    MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "F" +
CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
Next i

MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "变异原因
```

```
DF                    SS                    MS                    F" + Chr(13) +
Chr(10)
For i = 1 To Element
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "x" + CStr(i) +
"的偏回归" + "          " + CStr(1) + "          " + Chr(9) + CStr(Up(i)) + "          " + Chr(9) + CStr(Up(i)) +
"          " + Chr(9) + CStr(FF(i)) + "          " + Chr(13) + Chr(10)
Next i
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "离回归
" + CStr(DF_r) + "          " + Chr(9) + CStr(SS_r) + "          " + Chr(9) + CStr(MS_r) + Chr(13) + Chr(10)

Dim Return_FP001 As Double
Dim Return_FP005 As Double
Return_FP001 = obj2.PFaN1N2X2(0.01, 10000000, 1, DF_r) ' 0.01 用千万
Return_FP005 = obj2.PFaN1N2X2(0.05, 1000000, 1, DF_r)     ' 0.05 用百万
'Dim Return_FPReport As String
'For i = 1 To Element
'If (FF(i) > Return_FP001) Then
'Return_FPReport = "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" + CStr(Return_FP001)
'ElseIf (Return_FP001 > F And F > Return_FP005) Then
Rem  下面没有验证
'Return_FPReport = "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + Chr(13) +
Chr(10)
'Return_FPReport = Return_FPReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_FP005)
'ElseIf (Return_FP005 > F) Then
'Return_FPReport = "无显著的回归关系" + "F0.05=" + CStr(Return_FP005) + ">" + "F=" + CStr(F)
'End If
'Next
'MultiElementLinearAnalysis_Equation５  = MultiElementLinearAnalysis_Equation５  + "F 检验结
果" + Return_FPReport + Chr(13) + Chr(10)
'MultiElementLinearAnalysis_Equation５  = MultiElementLinearAnalysis_Equation５  +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
Dim Remove() As Double '  为剔除 Upi 值
Dim NumRemove As Long

Dim Return_FPReport As String
For i = 1 To Element
   If (FF(i) > Return_FP001) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "显著的回归关系" + "FF" + CStr(i) +
"=" + CStr(FF(i)) + ">" + "F0.01=" + CStr(Return_FP001) + Chr(13) + Chr(10)
   ElseIf (Return_FP001 > FF(i) And FF(i) > Return_FP005) Then
   Rem  下面没有验证
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "FF" + CStr(i) + "=" + CStr(FF(i)) + ">" + "F0.05=" +
CStr(Return_FP005) + Chr(13) + Chr(10)
```

```
   ElseIf (Return_FP005 > FF(i)) Then
   Return_FPReport = Return_FPReport + "F" + CStr(i) + ":" + "无显著的回归关系" + "F0.05=" +
CStr(Return_FP005) + ">" + "FF" + CStr(i) + "=" + CStr(FF(i)) + Chr(13) + Chr(10)
'*****************************************
   NumRemove = NumRemove + 1
   ReDim Preserve Remove(NumRemove)
   Remove(NumRemove) = i
'*****************************************
   End If
Next i
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "F 检验结果"
+ Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
Return_FPReport + Chr(13) + Chr(10)
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)



'*************************
Rem 非必要程序没有去验证，来源于田间试验与统计分析
Dim SumUp As Double
For i = 1 To Element
SumUp = SumUp + Up(i)
Next i

l = 0

If Uy = SumUp Then
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "各自变量独
立不相关" + Chr(13) + Chr(10)
Else
       For i = 1 To Element
          If Uy > SumUp And Return_Coefficient1D(i + 1) > 0 Then
          l = l + 1
          End If
       Next i

       If l = 3 Then
       MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "变量
之间正相关" + Chr(13) + Chr(10)
       End If
End If
l = 0
'*************************************************************************
```

```
********
Dim UpArray() As Double
Dim rlu() As Double, rul() As Double
Dim Which() As Long
Dim obj5 As New Array1D001
n = 0

If NumRemove >= 1 Then
    For i = LBound(Remove) To UBound(Remove)
    n = n + 1
    ReDim Preserve UpArray(n)
    UpArray(n) = Up(Remove(i))
    Next i
    n = 0

    For i = LBound(UpArray) To UBound(UpArray)
    MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 +
"UpArray(" + CStr(i) + ")=" + CStr(UpArray(i)) + Chr(13) + Chr(10)
    Next

    Dim MixNum As Double
    MixNum = obj5.Array1D_Mix(UpArray) ' 如果有两个数据一样，会给出一个 16-8-30
    For i = LBound(Up) To UBound(Up)
      If MixNum = Up(i) Then ' 如果有两个数据一样，会给出前面的 16-8-30
      Kick = i
      MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "需要
剔除的组数是" + CStr(Kick) + Chr(13) + Chr(10)
      End If
    Next i

End If
'**************************************
Dim R() As Double
ReDim R(1 To Element + 1, 1 To Element + 1)
For i = 1 To Element + 1
For l = 1 To Element + 1
R(i, l) = Return_SP(i, l) / Sqr(SS1(i) * SS1(l))
MultiElementLinearAnalysis_Equation5 = MultiElementLinearAnalysis_Equation5 + "r(" + CStr(i) +
"," + CStr(l) + ")=" + CStr(R(i, l)) + Chr(13) + Chr(10)
Next l
Next i
End Function
```

'TheBestMultipleLinearRegressionEquation_R 最好的多元线性回归返回 r 值

Rem 16-9-12 用于返回数据 Return_Element，通径分析用

```
Public Function TheBestMultipleLinearRegressionEquation_R(ByRef DataX() As Double, _
ByRef DataY() As Double, ByVal Element As Long, ByRef Return_Coefficient1D() As Double, _
ByRef Kick As Long, ByRef Return_Element As Long, ByRef PartialCorrelation() As Double, _
ByRef Return_PartialCorrelation() As Double, ByRef Return_CorrelationMatrixR() As Double,
ByRef Return_r() As Double) As String

'*********************************************************
        Rem 16-9-12
        Dim obj1 As New FileWrite001
        Dim ArrayRange As Integer
        ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
        Element = UBound(DataX, 2) - LBound(DataX, 2) + 1
        End If
        If ArrayRange = 1 Then
        Element = 1
        End If
'*********************************************************
Dim cDataX() As Double
Dim c1DataX() As Double
Dim cElement As Long
cDataX = DataX
cElement = Element
Kick = 200 ' 随便给个开始值 16-8-30
Dim LoopTimes As Long
LoopTimes = 0
'*********************************************************
Dim i As Long
Dim Initial() As Double
ReDim Initial(1 To Element)
For i = 1 To Element
Initial(i) = i
Next
'*********************************************************
Dim P As Long
P = 0 ' 防止 P<>0 16-9-4
Do While Kick > 0
```

```
Dim rSS() As Double, rSSy As Double, rUy As Double, rSP() As Double, rSPy() As Double, R() As Double
Dim n As Long
n = 0
'TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + MultiElementLinearAnalysis_Equation2(cDataX, DataY, cElement, Return_Coefficient1D, Kick) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + MultiElementLinearAnalysis_Equation4(cDataX, DataY, cElement, Return_Coefficient1D, Kick, rSS, rSSy, rUy, rSP, rSPy, R) + Chr(13) + Chr(10)
Return_r = R
'*********************************
Rem 16-9-6 一元线性分析，就不用下面的偏相关分析内容了，直接跳出
If Kick = -1 Then
Exit Function
End If
'*********************************
Rem 16-9-4 读取第一次的 Return_Coefficient1D
P = P + 1
Dim Old_Coefficient1D() As Double
Dim Old_SS() As Double
Dim Old_SSy As Double
Dim Old_Uy As Double
If (P = 1) Then
Old_Coefficient1D = Return_Coefficient1D
Old_SS = rSS
Old_SSy = rSSy
Old_Uy = rUy
End If

If Kick > 0 Then '从 K<>0 改成 K>0
LoopTimes = LoopTimes + 1 ' 记录循环次数，对后面更改数组数有帮助 16-8-30
c1DataX = cDataX 'cDataX 使用后就不能保存原来数据了，所以启用 c1DataX16-8-30
cElement = cElement - 1 '每次减少一列数
        '*************************
        Dim obj2 As New Array1D001
        Dim R3() As Double
        obj2.Array1D_RemoveOneNum Initial, Kick, R3
        'ReDim Preserve Initial(LBound(r) To UBound(r))
        Initial = R3

      Dim l As Long
      Dim Num As Long
      Num = UBound(DataX) - LBound(DataX) + 1
```

```
        ReDim cDataX(1 To Num, 1 To cElement) As Double

        n = 0 'n 的初始化
        TheBestMultipleLinearRegressionEquation_R =
TheBestMultipleLinearRegressionEquation_R + "剔除的数据是" + CStr(Kick) + "组数据" + Chr(13)
+ Chr(10)
        TheBestMultipleLinearRegressionEquation_R =
TheBestMultipleLinearRegressionEquation_R + "新的 X 数据如下" + Chr(13) + Chr(10)
        For l = 1 To (Element - LoopTimes + 1)
                If Kick <> l Then
                n = n + 1
                End If
                For i = 1 To Num
                   If Kick <> l Then
                   cDataX(i, n) = c1DataX(i, l)
                   'Debug.Print "cDatax(" & i & "," & n & ")=" & "c1datax(" & i & "," & L & ")" & "="
& c1DataX(i, L)

                   TheBestMultipleLinearRegressionEquation_R =
TheBestMultipleLinearRegressionEquation_R + "CDataX(" + CStr(i) + "," + CStr(n) + ")=" +
CStr(cDataX(i, n)) + Chr(13) + Chr(10)
                   End If
               Next i
        Next l
        n = 0
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Loop
'If Kick = 0 Then
'TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R +
MultiElementLinearAnalysis_Equation3(cDataX, DataY, cElement, Return_Coefficient1D, Kick)
'Exit Function
'End if
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  偏回归系数


Dim PartialRegressionB() As Double
For i = LBound(Initial) To UBound(Initial)
'Debug.Print Initial(i)
ReDim Preserve PartialRegressionB(Initial(i))
PartialRegressionB(Initial(i)) = Old_Coefficient1D(Initial(i) + 1) * Sqr(Old_SS(Initial(i)) / rSSy)
'PartialRegressionB(Initial(i)) = Old_Coefficient1D(Initial(i) + 1) * Sqr(SS(Initial(i)) / SSy)
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + _
```

```
"标准偏回归系数 PartialRegressionB(" + CStr(Initial(i)) + ")=" + CStr(PartialRegressionB(Initial(i))) + Chr(13) + Chr(10)
Next i
'*************************************************************************
Rem 16-9-4
'第二节 复相关分析
'SSR = Uy
Dim R1 As Double
Dim SquareR As Double
SquareR = rUy / Old_SSy
R1 = Sqr(rUy / Old_SSy) ' 发现前面程序有使用 R 的情况，已经更改为 R3（），16-9-4
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + "复相关分析~~~~~~~~~~~~~~~" + CStr(SquareR) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + "相关指数（correlation index）R^2=" + CStr(SquareR) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + "复相关系数（multiple  correlation coefficient）R=" + CStr(R1) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + "FR 和 F 是等价的，和数据的 F 检验相同" + Chr(13) + Chr(10)
'*第三节  偏相关分析
'*************************************************************************
'相关矩阵

'For i = 1 To Element - LoopTimes + 1
'Debug.Print " rSPy(" & i & ", 1)=" & rSPy(i, 1)
'Next i
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + CStr(SquareR) + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + "偏相关分析:" + Chr(13) + Chr(10)
Dim NewSP() As Double
ReDim NewSP(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1)
For i = 1 To Element - LoopTimes + 1
For l = 1 To Element - LoopTimes + 1
If (i <> (Element - LoopTimes + 1) And l <> (Element - LoopTimes + 1)) Then
NewSP(i, l) = rSP(i, l)
Else
'NewSP(i, l) = rSPy(1, 1)
If (i = Element - LoopTimes + 1 And l = Element - LoopTimes + 1) Then
NewSP(i, l) = rSSy
Exit For
End If
If (l = Element - LoopTimes + 1) Then
NewSP(i, l) = rSPy(i, 1)
```

```
End If
If (i = Element - LoopTimes + 1) Then
NewSP(i, l) = rSPy(l, 1)
End If

End If
Next l
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Dim NewSS() As Double
ReDim NewSS(1 To Element - LoopTimes + 1)
For i = 1 To Element - LoopTimes + 1
If (i <> Element - LoopTimes + 1) Then
NewSS(i) = rSS(i)
Else
NewSS(i) = rSSy '  是不是 SS4=SSY 这个还需要确认下
End If
Next i

Dim CorrelationMatrixR() As Double
ReDim CorrelationMatrixR(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1) '16-9-10
由 1 TO ELEMENT 改成 Element - LoopTimes + 1
For i = 1 To Element - LoopTimes + 1
For l = 1 To Element - LoopTimes + 1
CorrelationMatrixR(i, l) = NewSP(i, l) / Sqr(NewSS(i) * NewSS(l))

'Debug.Print "R(" + CStr(i) + "," + CStr(l) + ")=" + CStr(CorrelationMatrixR(i, l))
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + "
简单相关系数 R(" + CStr(i) + "," + CStr(l) + ")=" + CStr(CorrelationMatrixR(i, l)) + Chr(13) + Chr(10)
Next l
Next i

Return_CorrelationMatrixR = CorrelationMatrixR '19-9-18

Dim obj As New Matrix001
Dim InverseCorrelationMatrixR() As Double
obj.InverseMatrix CorrelationMatrixR, InverseCorrelationMatrixR

For i = 1 To Element - LoopTimes + 1
For l = 1 To Element - LoopTimes + 1
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R +
"(InverseCorrelationMatrixR)R^-1(" + CStr(i) + "," + CStr(l) + ")=" +
CStr(InverseCorrelationMatrixR(i, l)) + Chr(13) + Chr(10)
```

```
Next l
Next i

'Dim PartialCorrelation() As Double
ReDim PartialCorrelation(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1) As Double
ReDim Return_PartialCorrelation(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1) As Double
Dim Sr() As Double
ReDim Sr(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1) As Double
Dim t() As Double, T1() As Double
n = 0
ReDim t(1 To Element - LoopTimes + 1, 1 To Element - LoopTimes + 1)
'*****************************************
Rem 16-9-19

Num = UBound(DataX) - LBound(DataX) + 1
'*****************************************
For i = 1 To Element - LoopTimes + 1
For l = 1 To Element - LoopTimes + 1

   If (l <> i) Then
   Return_PartialCorrelation(i, l) = -InverseCorrelationMatrixR(l, i) /
Sqr(InverseCorrelationMatrixR(i, i) * InverseCorrelationMatrixR(l, l))
   If (i < l) Then
     PartialCorrelation(i, l) = -InverseCorrelationMatrixR(l, i) / Sqr(InverseCorrelationMatrixR(i, i) *
InverseCorrelationMatrixR(l, l))
     TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R
+ "偏相关系数 r(" + CStr(i) + "," + CStr(l) + ")=" + CStr(PartialCorrelation(i, l)) + Chr(13) + Chr(10)
     Sr(i, l) = Sqr((1 - PartialCorrelation(i, l) ^ 2) / (Num - (Element - LoopTimes + 1)))
     n = n + 1
     t(i, l) = PartialCorrelation(i, l) / Sr(i, l)
     ReDim Preserve T1(n)
     T1(n) = PartialCorrelation(i, l) / Sr(i, l)
     TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R
+ "t(" + CStr(i) + "," + CStr(l) + ")=" + CStr(t(i, l)) + Chr(9) + "      对应 T" + CStr(n) + Chr(13) +
Chr(10)
   End If
   End If
Next l
Next i

'*****************************************
Rem  偏相关假设检验
Dim Return_T001 As Double, Return_T005 As Double
```

```
Dim obj4 As New T001
Rem 16-9-7 日改
Return_T001 = obj4.t(0.005, Num - (Element - LoopTimes + 1))
Return_T005 = obj4.t(0.025, Num - (Element - LoopTimes + 1))
'*********************************************
Rem 16-9-7 参照田间统计，T 取绝度值
Dim T2() As Double
n = 0
For i = LBound(T1) To UBound(T1)
n = n + 1
ReDim Preserve T2(n)
T2(i) = Abs(T1(i))
Next i
'*********************************************
Dim Return_TReport As String
For i = 1 To Element - LoopTimes + 1
   If (T2(i) > Return_T001) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "显著的回归关系" + "T" + CStr(i) + "="
+ CStr(T2(i)) + ">" + "T0.01=" + CStr(Return_T001) + Chr(13) + Chr(10)
   ElseIf (Return_T001 > T2(i) And T2(i) > Return_T005) Then
   Rem 下面没有验证
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "T" + CStr(i) + "=" + CStr(T2(i)) + ">" + "T0.05=" +
CStr(Return_T005) + Chr(13) + Chr(10)
   ElseIf (Return_T005 > T2(i)) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "无显著的回归关系" + "T0.05=" +
CStr(Return_T005) + ">" + "T" + CStr(i) + "=" + CStr(T2(i)) + Chr(13) + Chr(10)
   End If
Next i
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R + "T
检验结果" + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R +
Return_TReport + Chr(13) + Chr(10)
TheBestMultipleLinearRegressionEquation_R = TheBestMultipleLinearRegressionEquation_R +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Return_Element = cElement
End Function
```

'PathAnalysis_Test1 通径分析测试 1

```
Public Function PathAnalysis_Test1(ByRef a() As Double, ByRef c() As Double, ByRef R() As Double)
As String
```

```
ReDim a(1 To 3, 1 To 3)
ReDim c(1 To 3, 1)
ReDim R(1 To 4, 1 To 4)
a(1, 1) = 1
a(1, 2) = 0.855
a(1, 3) = -0.468
a(2, 1) = 0.855
a(2, 2) = 1
a(2, 3) = -0.675
a(3, 1) = -0.468
a(3, 2) = -0.675
a(3, 3) = 1
c(1, 1) = 0.748
c(2, 1) = 0.895
c(3, 1) = -0.487
R(1, 1) = 1
R(1, 2) = 0.855
R(1, 3) = -0.468
R(1, 4) = 0.748
R(2, 1) = 0.855
R(2, 2) = 1
R(2, 3) = -0.675
R(2, 4) = 0.895
R(3, 1) = -0.468
R(3, 2) = -0.675
R(3, 3) = 1
R(3, 4) = -0.487
R(4, 1) = 0.748
R(4, 2) = 0.895
R(4, 3) = -0.487
R(4, 4) = 1
PathAnalysis_Test1 = "程序截流"
End Function
```

'PathAnalysis_Test2 通径分析测试 2

```
Public Function PathAnalysis_Test2(ByRef a() As Double, ByRef c() As Double, ByRef R() As Double)
As String
ReDim a(1 To 4, 1 To 4)
ReDim c(1 To 4, 1)
ReDim R(1 To 5, 1 To 5)
a(1, 1) = 1
a(1, 2) = 0.132
```

```
a(1, 3) = 0.0903
a(1, 4) = 0.0864
a(2, 1) = 0.132
a(2, 2) = 1
a(2, 3) = 0.9573
a(2, 4) = 0.9274
a(3, 1) = 0.0903
a(3, 2) = 0.9573
a(3, 3) = 1
a(3, 4) = 0.9239
a(4, 1) = 0.0864
a(4, 2) = 0.9274
a(4, 3) = 0.9239
a(4, 4) = 1

c(1, 1) = 0.2026
c(2, 1) = 0.7644
c(3, 1) = 0.7981
c(4, 1) = 0.7561

R(1, 1) = 1
R(1, 2) = 0.132
R(1, 3) = 0.0903
R(1, 4) = 0.0864
R(1, 5) = 0.2026
R(2, 1) = 0.132
R(2, 2) = 1
R(2, 3) = 0.9573
R(2, 4) = 0.9274
R(2, 5) = 0.7644
R(3, 1) = 0.0903
R(3, 2) = 0.9573
R(3, 3) = 1
R(3, 4) = 0.9239
R(3, 5) = 0.7981
R(4, 1) = 0.0864
R(4, 2) = 0.9274
R(4, 3) = 0.9239
R(4, 4) = 1
R(4, 5) = 0.7561

PathAnalysis_Test2 = "程序截流"
End Function
```

31. StringOpration

详细说明：

函数或方法名称	作用
AddAtAddPosition	字符串在 AddPosition 位置添加内容
EraseMark	字符串删除特定字符
EraseMarkReplace	字符串删除特定字符（未测试）
BetweenBracket	判断是否在两个指定括号之间，括号是成对出现的是前题
BetweenTowMark	判断是否在两个指定符号之间
FindMark	找到指定字符
RemoveEmpty	去掉字符串中的所有空格
ReCreateTest	修改字符串
FindNumberDbl	找到字符串中 Double 数字串
FindNumLng	找到字符串中 Long 数字串
FindNumbeFSRVar	找到字符串中 Variant 数字串

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*************************************************************************
'名称 = S4_StringOpration.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 12 月 31 日
'简介 = 操作字符串类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=None
'操作历史：
'修改名称和部分程序                                    2020-4-12
'添加 FindNumberDbl                                    2020-4-20
'添加 FindNumLng                                       2021-12-31
'添加 FindNumbeFSRVar                                  2022-1-4
'*************************************************************************
```

'AddAtAddPosition 字符串在 AddPosition 位置添加内容

```
Rem 字符串在 AddPosition 位置添加内容
Rem Add=添加的字符串内容
```

Rem OldString=添加前的内容
Rem AddPosition=添加的位置
Rem AddAtAddPosition=返回新的字符串

```
Public Function AddAtAddPosition(ByVal Add As String, ByVal OldString As String, _
ByVal AddPosition As Long) As String
Dim PartLeft As String, PartRight As String, PartAdd As String
If AddPosition <> -1 Then
PartLeft = Left(OldString, AddPosition)
PartRight = Right(OldString, Len(OldString) - AddPosition)
PartAdd = Add
AddAtAddPosition = PartLeft + Add + PartRight
End If
End Function
```

'EraseMark 字符串删除特定字符

Rem 字符串删除特定字符
Rem Test=需要操作的字符串
Rem EraseMark=需要删除的字符
Rem EraseStartPosition=开始删除的位置
Rem EraseEndPosition=结束的位置
Rem EraseNumber=删除的数量，有可能有多个这种字符串

```
Public Sub EraseMark(ByVal Test As String, ByRef ReturnTest As String, ByVal EraseMark As String,
_
Optional EraseStartPosition As Long, Optional EraseEndPosition As Long, _
Optional EraseNumber As Long)
Dim i As Long, Count As Long
Dim PartL As String, PartR As String
If EraseEndPosition = 0 Then
EraseEndPosition = Len(Test)
End If
If EraseStartPosition = 0 Then
EraseStartPosition = 1
End If
For i = EraseStartPosition To EraseEndPosition
        If Mid(Test, i, 1) = EraseMark Then
            PartL = Left(Test, i - 1)
            PartR = Right(Test, Len(Test) - i)
            Count = Count + 1
            ReturnTest = PartL + PartR
            If Count = EraseNumber Then
            Exit Sub
            End If
```

```
            End If
Next i
End Sub
```

'EraseMarkReplace 字符串删除特定字符（未测试）

Rem 未测试
Rem 字符串删除特定字符
Rem CalculateText=目标字符串
Rem EraseMark=删除目标
Rem EraseStartPosition=删除开始位置
Rem EraseNumber=删除数量

```
Public Sub EraseMarkReplace(ByRef CalculateText As String, ByVal EraseMark As String, _
Optional EraseStartPosition As Long, _
Optional EraseNumber As Long)
CalculateText = Replace(CalculateText, EraseMark, "", EraseStartPosition, EraseNumber,
vbTextCompare)
End Sub
```

'BetweenBracket 判断是否在两个指定括号之间，括号是成对出现的是前题

Rem 判断是否在两个指定括号之间，括号是成对出现的是前题
Rem Text=源字符串
Rem Postion=判断的字符串的位置
Rem LeftStr=左边括号
Rem RightStr=右边括号

```
Public Function BetweenBracket(ByRef Text As String, ByVal Postion As Long, _
ByVal LeftStr As String, ByVal RightStr As String) As Boolean
Dim i As Long
For i = Postion To Len(Text) Step 1
    If Mid(Text, i, 1) = RightStr Then '找到右括号
    BetweenBracket = True
    Exit Function
    End If
    If Mid(Text, i, 1) = LeftStr Then '找到左括号
    BetweenBracket = False
    Exit Function
    End If
Next i
End Function
```

'BetweenTowMark 判断是否在两个指定符号之间

Rem 判断是否在两个指定符号之间
Rem Text=源字符串
Rem Postion=判断的字符的位置
Rem LeftMark=左边字符
Rem RightMark=右边字符

```
Public Function BetweenTowMark(ByRef Text As String, ByVal Postion As Long, _
ByVal LeftMark As String, ByVal RightMark As String) As Boolean
Dim i As Long
Dim Part1 As Boolean, Part2 As Boolean
For i = Postion To Len(Text) Step 1
    If Mid(Text, i, 1) = RightMark Then '找到右括号
    Part1 = True
    End If
Next i
For i = Postion To 1 Step -1
    If Mid(Text, i, 1) = LeftMark Then '找到左括号
    Part2 = True
    End If
Next i
If Part1 = True And Part2 = True Then
    BetweenTowMark = True
End If
End Function
```

'FindMark 找到指定字符

Rem 找到指定字符
Rem Text=源字符串
Rem Start=开始位置
Rem Mark=寻找的字符
Rem ReturnPosition=返回找到字符的位置

```
Public Sub FindMark(ByVal Text As String, ByVal Start As Long, ByVal Mark As String, _
ByRef ReturnPosition As Long)
Dim i As Long
For i = Start To Len(Text) Step 1
    If Mid(Text, i, 1) = Mark Then
    ReturnPosition = i
    Exit For
    End If
Next i
```

```
End Sub
```

'RemoveEmpty 去掉字符串中的所有空格

Rem 去掉字符串中的所有空格
Rem Test=源字符串

```
Public Function RemoveEmpty(ByVal Test As String) As String
RemoveEmpty = Replace(Test, " ", "") '去掉算式中的空格
End Function
```

'ReCreateTest 修改字符串

Rem 修改字符串
Rem Test=源字符串
Rem ReCreateStartPosition=替换开始的位置
Rem PartTest=要被替换的字符段
Rem ReplacePartTest=替换成的字符段
Rem ReturnNewTest=返回替换后的字符串

```
Public Sub ReCreateTest(ByVal Test As String, ByVal ReCreateStartPosition As Long, _
ByVal PartTest As String, ByVal ReplacePartTest As String, ByRef ReturnNewTest As String)
Dim RightBracketPosition As Long
Dim LeftPartString As String
If ReCreateStartPosition - 1 > 0 Then
LeftPartString = Left(Test, ReCreateStartPosition - 1)
Else
LeftPartString = ""
ReCreateStartPosition = 1
End If
ReturnNewTest = Replace(Test, PartTest, ReplacePartTest, ReCreateStartPosition, _
1, vbTextCompare)
ReturnNewTest = LeftPartString + ReturnNewTest
Debug.Print ReturnNewTest
End Sub
```

'FindNumberDbl 找到字符串中 Double 数字串

Rem 作者:xy-wolf
Rem 主要部分来自：https://bbs.csdn.net/wap/topics/110056538

```
Public Function FindNumberDbl(ByVal TargetString As String, ByRef ReturnNumber() As Double)
Erase ReturnNumber
Dim Acount As Integer
```

```
Dim i As Integer
Dim ReturnNumberNum As Boolean
Acount = 1
ReDim Preserve ReturnNumber(1 To Acount)
For i = 1 To Len(TargetString)
    If (Asc(Mid(TargetString, i, 1)) >= 48 And Asc(Mid(TargetString, i, 1)) <= 57) _
        Or Asc(Mid(TargetString, i, 1)) = 46 Then
        ReturnNumber(Acount) = ReturnNumber(Acount) & Mid(TargetString, i, 1)
        If ReturnNumberNum = False Then
            ReturnNumberNum = True
        End If
    Else
        If ReturnNumberNum = True Then
            Acount = Acount + 1
            ReDim Preserve ReturnNumber(1 To Acount)
            ReturnNumberNum = False
        End If
    End If
Next i
End Function
```

'FindNumLng 找到字符串中 Long 数字串

Rem 在字符串中找到数字

```
Sub FindNumLng(ByVal Data As Variant, ByRef R_Data() As Long)
Dim A() As Variant
Dim f As Boolean
Dim i As Long, j As Long
Dim n As Long
For i = 1 To Len(Data)
    f = False
    If IsNumeric(Mid(Data, i, 1)) Then
        If i > 1 Then
            If IsNumeric(Mid(Data, i - 1, 1)) Then
            f = True
            A(n) = A(n) & Mid(Data, i, 1)
            'Debug.Print A(n)
            End If
        End If
        If f = False Then
            n = n + 1
            ReDim Preserve A(n)
            ReDim Preserve R_Data(n)
```

```
                            A(n) = Mid(Data, i, 1)
                        End If
        End If
  Next i
If UBound(A) > 0 Then
            For i = 1 To UBound(A)
               R_Data(i) = CLng(A(i))
               'Debug.Print "R_Data(" & i; ")=" & R_Data(i)
            Next i
End If
End Sub
```

'FindNumbeFSRVar 找到字符串中 Variant 数字串

Rem 作者:xy-wolf

Rem 主要部分来自：https://bbs.csdn.net/wap/topics/110056538

```
Public Function FindNumbeFSRVar(ByVal TargetString As String, ByRef ReturnNumber() As
Variant)
Erase ReturnNumber
Dim Acount As Integer
Dim i As Integer
Dim ReturnNumberNum As Boolean
Acount = 1
ReDim Preserve ReturnNumber(1 To Acount)
For i = 1 To Len(TargetString)
      If (Asc(Mid(TargetString, i, 1)) >= 48 And Asc(Mid(TargetString, i, 1)) <= 57) _
           Or Asc(Mid(TargetString, i, 1)) = 46 Then
          ReturnNumber(Acount) = ReturnNumber(Acount) & Mid(TargetString, i, 1)
          If ReturnNumberNum = False Then
              ReturnNumberNum = True
          End If
      Else
          If ReturnNumberNum = True Then
              Acount = Acount + 1
              ReDim Preserve ReturnNumber(1 To Acount)
              ReturnNumberNum = False
          End If
      End If
Next i
End Function
```

32. VBScript

详细说明：

函数或方法名称	作用
VBScriptSet	脚本设置
VBScriptRunWithOutParameter	脚本运行显示错误报告
Example	带有参数的脚本例子
JS	脚本算算数

代码：

```
Option Explicit
Option Base 1
'*************************************************************************
'名称 = VBScript.Cls
'作者 = 张宏
'最后修改时间 = 2022 年 1 月 2 日
'简介 = VB 脚本操作
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=
'操作历史：
'*************************************************************************
```

'VBScriptSet 脚本设置

```
Rem 脚本设置
Rem Str=脚本文本
Rem ScriptControl=脚本名称
Public Sub VBScriptSet(ByVal Str As String, ByRef ScriptControl)
Set ScriptControl = CreateObject("ScriptControl")
ScriptControl.language = "VBScript"
ScriptControl.AddCode Str
End Sub
```

'VBScriptRunWithOutParameter 脚本运行显示错误报告

```
Rem 脚本运行显示错误报告
Rem RunProgram=脚本函数名称
```

Rem ScriptControl=脚本名称

```
Public Sub VBScriptRunWithOutParameter(ByRef ScriptControl, ByVal RunProgram As String)
On Error GoTo scError
ScriptControl.Run RunProgram
Exit Sub
scError:
Debug.Print ScriptControl.Error.Number & _
":" & ScriptControl.Error.Description & _
" in line " & ScriptControl.Error.Line
Exit Sub
End Sub
```

'Example 带有参数的脚本例子

Rem 带有参数的脚本例子

```
Private Sub Example()
' 创建函数。
Dim ScriptControl1
Dim strFunction As String
strFunction = _
"Function ReturnThis(x, y)" & vbCrLf & _
"ReturnThis = x * y" & vbCrLf & _
"End Function"
' 添加代码，然后运行该函数。
Set ScriptControl1 = CreateObject("ScriptControl")
ScriptControl1.language = "VBScript"
ScriptControl1.AddCode strFunction
MsgBox ScriptControl1.Run("ReturnThis", 3, 25)
End Function
```

'JS 脚本算算数

Rem https://zhidao.baidu.com/question/1445906878644459780.html

Rem qq15495835 2013-10-08/2020-4-3

Rem 用脚本算算数

```
Function JS(ByVal Expressions As String, Optional Wrong As Boolean) As String
   Dim i As Long
    Dim Mssc As Object
    Set Mssc = CreateObject("MSScriptControl.ScriptControl")
    Mssc.language = "vbscript"
    On Error GoTo EvalErr
     For i = 1 To Len(Expressions)
```

```
        If Mid(Expressions, i, 1) = "[" Then
        Mid(Expressions, i, 1) = "("
        End If
        If Mid(Expressions, i, 1) = "]" Then
        Mid(Expressions, i, 1) = ")"
        End If
        If Mid(Expressions, i, 1) = "X" Then
        Wrong = True
        Exit Function
        End If
        Next i
    JS = Mssc.Eval(Expressions)
    JS = CDec(JS)
    Exit Function
EvalErr:
    Report = Report + "出错计算式=" + Expressions
    'Debug.Print Expressions
    Wrong = True
    Exit Function
End Function
```

33. FindSubWindow【模块】

详细说明：

函数或方法名称	作用
EnumWinProc	枚举 Windows 程序
EnumChildProc	枚举子程序
EnumThreadProc	枚举线程程序
StripNulls	移除多余的 0 让字符串对比工作
PathAnalysis_Part1	通径分析 Part1
PathAnalysis_Part3	通径分析 Part3
Times	资料次数的计算
NormalDistributionPosibilityRectangle	正态分布概率新矩形法
Rectangle	NormalDistributionPosibilityRectangle 子程序
Rectangle1	NormalDistributionPosibilityRectangle 子程序
ArithmeticAverage	算数平均数
PopulationVariance	总体方差
SampleVariance	样本方差

代码：

Rem 使用 EnumWindows 枚举顶级窗口,使用 EnumChildWindows 枚举子窗口,
Rem 或者使用 EnumThreadWindows 枚举与某个线程关联的所有非子窗口是首选方法.

```
Option Explicit
Type RECT
Left As Long
Top As Long
Right As Long
Bottom As Long
End Type
Declare Function GetWindowRect Lib "user32" (ByVal hwnd As Long, _
lpRect As RECT) As Long
Declare Function MoveWindow Lib "user32" (ByVal hwnd As Long, _
ByVal X As Long, ByVal Y As Long, ByVal nWidth As Long, _
ByVal nHeight As Long, ByVal bRepaint As Long) As Long
Declare Function GetDesktopWindow Lib "user32" () As Long
Declare Function EnumWindows Lib "user32" _
(ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long
Declare Function EnumChildWindows Lib "user32" (ByVal hWndParent _
As Long, ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long
```

```
Declare Function EnumThreadWindows Lib "user32" (ByVal dwThreadId _
As Long, ByVal lpfn As Long, ByVal lParam As Long) As Long
Declare Function GetWindowThreadProcessId Lib "user32" _
(ByVal hwnd As Long, lpdwProcessId As Long) As Long
Declare Function GetClassName Lib "user32" Alias "GetClassNameA" _
(ByVal hwnd As Long, ByVal lpClassName As String, _
ByVal nMaxCount As Long) As Long
Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" _
(ByVal hwnd As Long, ByVal lpString As String, _
ByVal cch As Long) As Long
Public TopCount As Integer ' Number of Top level Windows
Public ChildCount As Integer ' Number of Child Windows
Public ThreadCount As Integer ' Number of Thread Windows
```

'EnumWinProc 枚举 Windows 程序

```
Function EnumWinProc(ByVal lhWnd As Long, ByVal lParam As Long) _
As Long
Dim RetVal As Long, ProcessID As Long, ThreadID As Long
Dim WinClassBuf As String * 255, WinTitleBuf As String * 255
Dim WinClass As String, WinTitle As String
RetVal = GetClassName(lhWnd, WinClassBuf, 255)
WinClass = StripNulls(WinClassBuf) ' remove extra Nulls & spaces
RetVal = GetWindowText(lhWnd, WinTitleBuf, 255)
WinTitle = StripNulls(WinTitleBuf)
TopCount = TopCount + 1
' see the Windows Class and Title for each top level Window
Debug.Print "Top level Class = "; WinClass; ", Title = "; WinTitle
' Usually either enumerate Child or Thread Windows, not both.
' In this example, EnumThreadWindows may produce a very long list!
RetVal = EnumChildWindows(lhWnd, AddressOf EnumChildProc, lParam)
ThreadID = GetWindowThreadProcessId(lhWnd, ProcessID)
RetVal = EnumThreadWindows(ThreadID, AddressOf EnumThreadProc, _
lParam)
EnumWinProc = True
End Function
```

'EnumChildProc 枚举子程序

```
Function EnumChildProc(ByVal lhWnd As Long, ByVal lParam As Long) _
As Long
Dim RetVal As Long
```

```
Dim WinClassBuf As String * 255, WinTitleBuf As String * 255
Dim WinClass As String, WinTitle As String
Dim WinRect As RECT
Dim WinWidth As Long, WinHeight As Long
RetVal = GetClassName(lhWnd, WinClassBuf, 255)
WinClass = StripNulls(WinClassBuf) ' remove extra Nulls & spaces
RetVal = GetWindowText(lhWnd, WinTitleBuf, 255)
WinTitle = StripNulls(WinTitleBuf)
ChildCount = ChildCount + 1
' see the Windows Class and Title for each Child Window enumerated
Debug.Print " Child Class = "; WinClass; ", Title = "; WinTitle
' You can find any type of Window by searching for its WinClass
If WinClass = "ThunderTextBox" Then ' TextBox Window
RetVal = GetWindowRect(lhWnd, WinRect) ' get current size
WinWidth = WinRect.Right - WinRect.Left ' keep current width
WinHeight = (WinRect.Bottom - WinRect.Top) * 2 ' double height
'RetVal = MoveWindow(lhWnd, 0, 0, WinWidth, WinHeight, True)'移动窗体
EnumChildProc = False
Else
EnumChildProc = True
End If
End Function
```

'EnumThreadProc 枚举线程程序

```
Function EnumThreadProc(ByVal lhWnd As Long, ByVal lParam As Long) _
As Long
Dim RetVal As Long
Dim WinClassBuf As String * 255, WinTitleBuf As String * 255
Dim WinClass As String, WinTitle As String
RetVal = GetClassName(lhWnd, WinClassBuf, 255)
WinClass = StripNulls(WinClassBuf) ' remove extra Nulls & spaces
RetVal = GetWindowText(lhWnd, WinTitleBuf, 255)
WinTitle = StripNulls(WinTitleBuf)
ThreadCount = ThreadCount + 1
' see the Windows Class and Title for top level Window
Debug.Print "Thread Window Class = "; WinClass; ", Title = "; _
WinTitle
EnumThreadProc = True
End Function
```

'StripNulls 移除多余的 0 让字符串对比工作

```
Public Function StripNulls(OriginalStr As String) As String
' This removes the extra Nulls so String comparisons will work
If (InStr(OriginalStr, Chr(0)) > 0) Then
OriginalStr = Left(OriginalStr, InStr(OriginalStr, Chr(0)) - 1)
End If
StripNulls = OriginalStr
End Function
```

'PathAnalysis_Part1 通径分析 Part1

Rem 通径分析

Rem 至少是 2 元？

```
Public Function PathAnalysis_Part1(ByRef DataX() As Double, ByRef DataY() As Double, _
ByRef KickPy As Long, ByRef Return_A() As Double, ByRef Return_C() As Double, ByRef Return_B()
As Double, _
ByRef Return_SSr As Double) As String
PathAnalysis_Part1 = PathAnalysis_Part1 + "PathAnalysis_Part1 程序开始" + Chr(13) + Chr(10)
        Dim Element As Long
        Dim i As Long, l As Long
        Dim Num As Long
        Num = UBound(DataY) - LBound(DataY) + 1

        KickPy = 0 '16-9-22 初始化
        '********************
        Rem 16-9-22 防止程序重复运行不必要的部分
        'If Recall > 0 Then
        'GoTo Lab1:
        '******************
'*********************************************************
Rem 16-9-12

        Dim obj1 As New FileWrite001
        Dim ArrayRange As Integer
        ArrayRange = obj1.ArrayRangeD(DataX)
        If ArrayRange = 2 Then
        Element = UBound(DataX, 2) - LBound(DataX, 2) + 1
        End If
        If ArrayRange = 1 Then
        Element = 1
```

```
        KickPy = -1
        End If

'*********************************************************************
         Rem 判断数据是否可以通径分析
         Dim b() As Double
         Dim Return_Ele As Long, Kick As Long, PartialCorrelation() As Double,
Return_PartialCorrelation() As Double, Return_CorrelationMatrixR() As Double, R() As Double
         Dim StringTheBest As String
         StringTheBest = TheBestMultipleLinearRegressionEquation_R(DataX, DataY, Element, b(),
Kick, Return_Ele, PartialCorrelation, Return_PartialCorrelation, Return_CorrelationMatrixR, R)
         PathAnalysis_Part1 = PathAnalysis_Part1 + StringTheBest + Chr(13) + Chr(10)
         PathAnalysis_Part1 = PathAnalysis_Part1 + "~~~~~~~~~~~~~~~多元线性分析完毕
~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
         If (Return_Ele < 2) Then
         MsgBox "自变量数目小于 2. 无法通径分析  Exit Function PathAnalysis_Part1"
         Exit Function
         End If
'*********************************************************************

        Dim cDataX() As Double
        'ReDim cDataX(LBound(DataX) To UBound(DataX))
        If ArrayRange = 2 Then
             'For i = LBound(DataX) To UBound(DataX)
             cDataX = DataX
             'Next
        Else
             If ArrayRange = 1 Then
             ReDim cDataX(LBound(DataX) To UBound(DataX), 1)
             For i = LBound(DataX) To UBound(DataX)
             cDataX(i, 1) = DataX(i)
             Next
             Else
             MsgBox "数据维度有错，只支持 1 维和 2 维 。Exit Function PathAnalysis_Part1"
             Exit Function
             End If
        End If
'*********************************************************************

Dim PartX() As Double
Dim PartXSquare() As Double
ReDim PartX(1 To Element)
ReDim PartXSquare(1 To Element)
For i = 1 To Element
```

```
For l = 1 To Num
PartX(i) = PartX(i) + cDataX(l, i)
PartXSquare(i) = PartXSquare(i) + cDataX(l, i) ^ 2
Next l
Next i
'*******************************
Rem 前题是 Y 项和 X 项数量是相同的
Dim PartY As Double
Dim PartYSquare As Double
For l = 1 To Num
PartY = PartY + DataY(l)
PartYSquare = PartYSquare + DataY(l) ^ 2
Next l
'***********************************
ReDim SS(1 To Element) As Double
For i = 1 To Element
For l = 1 To Num
SS(i) = SS(i) + (cDataX(l, i) - PartX(i) / Num) ^ 2
'ReDim Preserve Return_SS(i)
'Return_SS(i) = SS(i)
Next l
'PathAnalysis_Part1 = PathAnalysis_Part1 + "SS" + CStr(i) + "=" + CStr(SS(i)) + Chr(13) + Chr(10)
Next i
  'PathAnalysis_Part1 = PathAnalysis_Part1 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" +
Chr(13) + Chr(10)
'*************************************
Rem 求 SSy 的值
Dim SSy As Double
For i = 1 To Num
SSy = SSy + (DataY(i) - PartY / Num) ^ 2
Next i
'Return_SSy = SSy
'PathAnalysis_Part1 = PathAnalysis_Part1 + "y 变量的总平方和 SSy" + "=" + CStr(SSy) + Chr(13) +
Chr(10)
' PathAnalysis_Part1 = PathAnalysis_Part1 + "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" +
Chr(13) + Chr(10)
'*****************************************************************************
Rem 求 SPik 的值
Rem 田间统计分析书上测试已经通过（16-8-24）
Dim n As Long, m As Long
Dim SP() As Double
ReDim SP(1 To Element, 1 To Element) As Double
ReDim Return_SP(1 To Element, 1 To Element) As Double
For i = 1 To Element
```

```
For l = 1 To Element
    '*********************************
   Rem 16-9-5
   For m = 1 To Num
   Return_SP(i, l) = Return_SP(i, l) + (cDataX(m, i) - PartX(i) / Num) * (cDataX(m, l) - PartX(l) /
Num)
   Next m
   '*********************************
   If (i <> l) Then
   If (i < l) Then

      For n = 1 To Num
      SP(i, l) = SP(i, l) + (cDataX(n, i) - PartX(i) / Num) * (cDataX(n, l) - PartX(l) / Num)

      Next n
      'PathAnalysis_Part1 = PathAnalysis_Part1 + "SP(" + CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i,
l)) + Chr(13) + Chr(10)
      'Debug.Print "SP(" + CStr(i) + "," + CStr(l) + ")=" + CStr(SP(i, l))
   End If
   End If
Next l
Next i

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PathAnalysis_Part1 = PathAnalysis_Part1 + "~~~~~~~~~~~通径分析开始~~~~~~~~~~~~~~"
+ Chr(13) + Chr(10)

'*******************************************************************************
    Rem 16-9-22
    Rem 剔除不显著的 Py 只需要用到下面的程序
'Lab1:

'*******************************************************************************

Dim a() As Double
Dim c() As Double
ReDim c(1 To Return_Ele, 1) As Double
ReDim a(1 To Return_Ele, 1 To Return_Ele) As Double
ReDim Return_A(1 To Return_Ele, 1 To Return_Ele) As Double
ReDim Return_C(1 To Return_Ele, 1) As Double
For i = 1 To Return_Ele + 1
For l = 1 To Return_Ele + 1
If (i <> Return_Ele + 1 And l <> Return_Ele + 1) Then
a(i, l) = R(i, l)
```

```
Return_A(i, l) = a(i, l)
PathAnalysis_Part1 = PathAnalysis_Part1 + "通径 A" + CStr(i) + "," + CStr(l) + "=" + CStr(a(i, l)) +
Chr(13) + Chr(10)
Else
    If (i = Return_Ele + 1) And (l = Return_Ele + 1) Then
    Else
    If (i = Return_Ele + 1) Then
    c(l, 1) = R(i, l)
    Return_C(l, 1) = c(l, 1)
    PathAnalysis_Part1 = PathAnalysis_Part1 + "通径 C" + CStr(i) + "=" + CStr(c(l, 1)) + Chr(13) +
Chr(10)
    End If
    End If
End If

Next l
Next i
'********************
Rem 程序截断
'Num = 273
'PathAnalysis_Part1 = PathAnalysis_Part1 + PathAnalysis_Test2(a, c, r) + Chr(13) + Chr(10)
'Return_Ele = 4
'**********************
Dim b1() As Double
'Dim Py() As Double
Dim InverseA() As Double
Dim obj As New Matrix001
obj.InverseMatrix a, InverseA
For i = 1 To Return_Ele
For l = 1 To Return_Ele
PathAnalysis_Part1 = PathAnalysis_Part1 + "InverseA" + CStr(i) + "," + CStr(l) + "=" +
CStr(InverseA(i, l)) + Chr(13) + Chr(10)
Next l
Next i
obj.MatrixMultiplication InverseA, c, b1
Return_B = b1
Rem  b1()= c()间接通径系数都不存在
PathAnalysis_Part1 = PathAnalysis_Part1 + "直接作用： " + Chr(13) + Chr(10)
For i = 1 To Return_Ele
PathAnalysis_Part1 = PathAnalysis_Part1 + "B" + CStr(i) + "=" + CStr(b1(i, 1)) + Chr(13) + Chr(10)
'Debug.Print b1(i, 1) '不知道是否正确
Next

'**************************************
```

```
Rem 间接通径系数计算 16-9-20
Dim P() As Double
ReDim P(1 To Return_Ele, 1 To Return_Ele)
PathAnalysis_Part1 = PathAnalysis_Part1 + "间接作用：" + Chr(13) + Chr(10)
For i = 1 To Return_Ele
For l = 1 To Return_Ele
If (i <> l) Then
P(i, l) = a(i, l) * b1(l, 1)
PathAnalysis_Part1 = PathAnalysis_Part1 + "P(" + CStr(i) + "," + CStr(l) + ",y)=" + CStr(P(i, l)) +
Chr(13) + Chr(10)
End If
Next l
Next i
'**************************************
Rem 决定系数计算 16-9-20
Dim Dy1() As Double
Dim Dy2() As Double
Dim Dy3() As Double
ReDim Dy1(1 To Return_Ele)
ReDim Dy2(1 To Return_Ele, 1 To Return_Ele)



n = 0
For i = 1 To Return_Ele
n = n + 1
ReDim Preserve Dy3(n)
Dy1(i) = b1(i, 1) ^ 2
Dy3(n) = Abs(Dy1(i))
PathAnalysis_Part1 = PathAnalysis_Part1 + "Dy(" + CStr(i) + ")=" + CStr(Dy1(i)) + Chr(13) + Chr(10)
For l = 1 To Return_Ele
If (i <> l) Then
If (i < l) Then
n = n + 1
ReDim Preserve Dy3(n)
Dy2(i, l) = 2 * a(i, l) * b1(i, 1) * b1(l, 1)
Dy3(n) = Abs(Dy2(i, l))
PathAnalysis_Part1 = PathAnalysis_Part1 + "Dy(" + CStr(i) + "," + CStr(l) + ")=" + CStr(Dy2(i, l)) +
Chr(13) + Chr(10)
End If
End If
Next l
Next i
n = 0
```

```
Dim Dye As Double
Dim Pye As Double
Dim obj2 As New Array1D001
Dim obj3 As New Array2D001
Dye = 1 - (obj2.Array1D_Sum(Dy1) + obj3.Array2D_Sum(Dy2))
Pye = Sqr(Dye)

Return_SSr = Dye

PathAnalysis_Part1 = PathAnalysis_Part1 + "Dye=" + CStr(Dye) + Chr(13) + Chr(10)
PathAnalysis_Part1 = PathAnalysis_Part1 + "Pye=" + CStr(Pye) + Chr(13) + Chr(10)
'**************************************
Rem 相关指数计算 16-9-20 Correlation index
Dim CorrelationIndex As Double
CorrelationIndex = obj2.Array1D_Sum(Dy1) + obj3.Array2D_Sum(Dy2)
'**************************************



Dim Record() As String
Dim RSLU1() As Double, RSUL1() As Double
obj2.Array1D_Sort Dy3, RSLU1, RSUL1
PathAnalysis_Part1 = PathAnalysis_Part1 + "影响 Y 的重要因素：" + Chr(13) + Chr(10)
Dim i1 As Long, l1 As Long
n = 0
For i = LBound(RSUL1) To UBound(RSUL1)

   For l = LBound(Dy1) To UBound(Dy1)
   If Abs(Dy1(l)) = RSUL1(i) Then
        If i <> UBound(RSUL1) Then
        n = n + 1
        ReDim Preserve Record(n)
        Record(n) = "|dy" + CStr(l) + "|=" + CStr(RSUL1(i))
        PathAnalysis_Part1 = PathAnalysis_Part1 + "|dy" + CStr(l) + "|=" + CStr(RSUL1(i)) + ">"
        Else
        n = n + 1
        ReDim Preserve Record(n)
        Record(n) = "|dy" + CStr(l) + "|=" + CStr(RSUL1(i))
        PathAnalysis_Part1 = PathAnalysis_Part1 + "|dy" + CStr(l) + "|=" + CStr(RSUL1(i)) + Chr(13)
+ Chr(10)
        End If
   End If
   Next l

   For i1 = LBound(Dy1, 1) To UBound(Dy1, 1)
```

```
   For l1 = LBound(Dy1, 1) To UBound(Dy1, 1)
   If (i1 <> l1) Then
   If (l1 > i1) Then
      If Abs(Dy2(i1, l1)) = RSUL1(i) Then
          If i <> UBound(RSUL1) Then
          n = n + 1
          ReDim Preserve Record(n)
          Record(n) = "|dy(" + CStr(i1) + "," + CStr(l1) + ")|=" + CStr(RSUL1(i))
          PathAnalysis_Part1 = PathAnalysis_Part1 + "|dy(" + CStr(i1) + "," + CStr(l1) + ")|=" +
CStr(RSUL1(i)) + ">"
          Else
          n = n + 1
          ReDim Preserve Record(n)
          Record(n) = "|dy(" + CStr(i1) + "," + CStr(l1) + ")|=" + CStr(RSUL1(i))
          PathAnalysis_Part1 = PathAnalysis_Part1 + "|dy(" + CStr(i1) + "," + CStr(l1) + ")|=" +
CStr(RSUL1(i)) + Chr(13) + Chr(10)
          End If
      End If
   End If
   End If
   Next l1
   Next i1


Next i
n = 0
'******************************************************
Rem 结论解释
Rem 通径系数重要性排列
PathAnalysis_Part1 = PathAnalysis_Part1 + "~~~~~~~~~~~~~~~结论解释~~~~~~~~~~~~~~~" +
Chr(13) + Chr(10)
PathAnalysis_Part1 = PathAnalysis_Part1 + "通径系数相对重要性排列：" + Chr(13) + Chr(10)
Dim RSLU() As Variant, RSUL() As Variant, RSLU2d() As Variant, RSUL2d() As Variant
Dim r_b1() As Variant
obj3.Array2D_CDataTypeDtoV b1, r_b1
obj3.Array2D_SortIncludeEmpty r_b1, RSLU, RSUL, RSLU2d, RSUL2d
For i = LBound(RSUL) To UBound(RSUL)
   'n = n + 1
   For l = LBound(RSUL) To UBound(RSUL)
   If r_b1(l, 1) = RSUL(i) Then
       If i <> UBound(RSUL) Then
          If (RSUL(i) > 0) Then
          PathAnalysis_Part1 = PathAnalysis_Part1 + "正效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
">"
```

```
          ElseIf (RSUL(i) = 0) Then
          PathAnalysis_Part1 = PathAnalysis_Part1 + "无效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
">"
          ElseIf (RSUL(i) < 0) Then
          PathAnalysis_Part1 = PathAnalysis_Part1 + "负效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
">"
          End If
        Else
          If (RSUL(i) > 0) Then
          PathAnalysis_Part1 = PathAnalysis_Part1 + "正效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
Chr(13) + Chr(10)
          ElseIf (RSUL(i) = 0) Then
          PathAnalysis_Part1 = PathAnalysis_Part1 + "无效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
Chr(13) + Chr(10)
          ElseIf (RSUL(i) < 0) Then
          PathAnalysis_Part1 = PathAnalysis_Part1 + "负效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
Chr(13) + Chr(10)
          End If
        End If
   End If
   Next l
Next i
PathAnalysis_Part1 = PathAnalysis_Part1 + "决定系数最大值为:" + Record(1) + "," + "说明" +
Record(1) + "是影响 Y 值第一因素" + Chr(13) + Chr(10)

 '*************************************************************************

 For i = 1 To Return_Ele


 If R(i, UBound(R, 2)) < 0 And b1(i, 1) > 0 Then
 PathAnalysis_Part1 = PathAnalysis_Part1 + "X" + CStr(i) + "与 Y 的相关系数为负数" + CStr(R(i,
UBound(R, 2))) + "； " + "直接作用为 Py" + CStr(i) + "=" + CStr(b1(i, 1)) + "两者符号相反： " +
Chr(13) + Chr(10)
        For l = 1 To Return_Ele
        If (i <> l) Then
        PathAnalysis_Part1 = PathAnalysis_Part1 + "因为 P(" + CStr(i) + "," + CStr(l) + ",y)=" +
CStr(P(i, l)) + Chr(13) + Chr(10)
        End If
        Next l
        PathAnalysis_Part1 = PathAnalysis_Part1 + "这些间接作用混杂在与 Y 的相关系数中，其
中负间接通径系数掩盖了对 Y 的正效应将其夸大为负值" + Chr(13) + Chr(10)
 End If
```

```
   If R(i, UBound(R, 2)) > 0 And b1(i, 1) < 0 Then
   PathAnalysis_Part1 = PathAnalysis_Part1 + "X" + CStr(i) + "与 Y 的相关系数为正数" + CStr(R(i, UBound(R, 2))) + "；" + "直接作用为 Py" + CStr(i) + "=" + CStr(b1(i, 1)) + "两者符号相反：" + Chr(13) + Chr(10)
        For l = 1 To Return_Ele
        If (i <> l) Then
        PathAnalysis_Part1 = PathAnalysis_Part1 + "因为 P(" + CStr(i) + "," + CStr(l) + ",y)=" + CStr(P(i, l)) + Chr(13) + Chr(10)
        End If
        Next l
        PathAnalysis_Part1 = PathAnalysis_Part1 + "这些间接作用混杂在与 Y 的相关系数中，其中正间接通径系数掩盖了对 Y 的负效应将其夸大为正值" + Chr(13) + Chr(10)
        PathAnalysis_Part1 = PathAnalysis_Part1 + Chr(13) + Chr(10)
  End If



  Next i
PathAnalysis_Part1 = PathAnalysis_Part1 + "相关指数 R^2=" + CStr(CorrelationIndex) + "," + "说明" + CStr(Return_Ele) + "因素决定 Y   " + CStr(CorrelationIndex * 100) + "%" + Chr(13) + Chr(10)

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  线性关系显著性检验
Dim Return_FReport As String
Dim DFR As Long, DF_r As Long
DFR = Return_Ele
DF_r = Num - Return_Ele - 1
PathAnalysis_Part1 = PathAnalysis_Part1 + "线性关系显著性检验 F 检验：" + Return_FReport + Chr(13) + Chr(10)
Dim F As Double
F = (CorrelationIndex / Return_Ele) / (Dye / (Num - Return_Ele - 1))
Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj4 As New F001
Return_F001 = obj4.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj4.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万


If (F > Return_F001) Then
Return_FReport = Return_FReport + "显著的回归关系" + "F=" + CStr(F) + ">" + "F0.01=" + CStr(Return_F001)
ElseIf (Return_F001 > F And F > Return_F005) Then
```

```
Rem 下面没有验证
Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + Chr(13) + Chr(10)
Return_FReport = Return_FReport + "F=" + CStr(F) + ">" + "F0.05=" + CStr(Return_F005)
ElseIf (Return_F005 > F) Then
Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">" + "F=" + CStr(F)
End If
PathAnalysis_Part1 = PathAnalysis_Part1 + "F 检验结果" + Return_FReport + Chr(13) + Chr(10)
PathAnalysis_Part1 = PathAnalysis_Part1 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************************************************************
Rem 通径系数显著性检验
PathAnalysis_Part1 = PathAnalysis_Part1 + "通径系数显著性检验：" + Chr(13) + Chr(10)
Dim Se As Double
Se = Sqr(Dye / (Num - Return_Ele - 1))
Dim S() As Double, t() As Double
ReDim S(1 To Return_Ele)
ReDim t(1 To Return_Ele)
For i = 1 To Return_Ele
S(i) = Se * Sqr(InverseA(i, i))
t(i) = b1(i, 1) / S(i)
'PathAnalysis_Part1 = PathAnalysis_Part1 + "t(" + CStr(i) + ")=" + CStr(t(i)) + Chr(13) + Chr(10)
Next
Dim Return_T001 As Double, Return_T005 As Double
Dim obj5 As New T001
Return_T001 = obj5.t(0.005, DF_r) '16-9-7 日改
Return_T005 = obj5.t(0.025, DF_r)
Dim Return_TReport As String

Dim T1() As Double
n = 0
For i = LBound(t) To UBound(t)
n = n + 1
ReDim Preserve T1(n)
T1(i) = Abs(t(i))
PathAnalysis_Part1 = PathAnalysis_Part1 + "t(" + CStr(i) + ")=" + CStr(t(i)) + Chr(13) + Chr(10)
Next i
n = 0
'*********************************************
Dim Remove() As Double ' 为剔除 Upi 值
Dim NumRemove As Long
n = 0
PathAnalysis_Part1 = PathAnalysis_Part1 + "以下 T 为绝对值" + Chr(13) + Chr(10)
```

```
For i = 1 To Return_Ele
   If (T1(i) > Return_T001) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "显著的回归关系" + "T" + CStr(i) + "="
+ CStr(T1(i)) + ">" + "T0.01=" + CStr(Return_T001) + Chr(13) + Chr(10)
   ElseIf (Return_T001 > T1(i) And T1(i) > Return_T005) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "T" + CStr(i) + "=" + CStr(T1(i)) + ">" + "T0.05=" +
CStr(Return_T005) + Chr(13) + Chr(10)
   ElseIf (Return_T005 > T1(i)) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "无显著的回归关系" + "T0.05=" +
CStr(Return_T005) + ">" + "T" + CStr(i) + "=" + CStr(T1(i)) + Chr(13) + Chr(10)

   '****************************************
   Rem 16-9-22 剔除检验不合格的
   NumRemove = NumRemove + 1
   ReDim Preserve Remove(NumRemove)
   Remove(NumRemove) = i
'****************************************
   End If
Next i
PathAnalysis_Part1 = PathAnalysis_Part1 + "T 检验结果" + Chr(13) + Chr(10)
PathAnalysis_Part1 = PathAnalysis_Part1 + Return_TReport + Chr(13) + Chr(10)
PathAnalysis_Part1 = PathAnalysis_Part1 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'********************************************************************************
Dim UpArray() As Double
Dim rlu() As Double, rul() As Double
Dim Which() As Long

n = 0

If NumRemove >= 1 Then
    For i = LBound(Remove) To UBound(Remove)
    n = n + 1
    ReDim Preserve UpArray(n)
    UpArray(n) = T1(Remove(i))
    Next i
    n = 0

    For i = LBound(UpArray) To UBound(UpArray)
    PathAnalysis_Part1 = PathAnalysis_Part1 + "UpArray(" + CStr(i) + ")=" + CStr(UpArray(i)) +
Chr(13) + Chr(10)
    Next
Dim MixNum As Double
```

```
    MixNum = obj2.Array1D_Mix(UpArray) ' 如果有两个数据一样，会给出一个 16-8-30
    For i = LBound(T1) To UBound(T1)
       If MixNum = T1(i) Then ' 如果有两个数据一样，会给出前面的 16-8-30
       KickPy = i
       PathAnalysis_Part1 = PathAnalysis_Part1 + "需要剔除的组数是" + CStr(KickPy) + Chr(13) +
Chr(10)
       End If
    Next i
Else
KickPy = 0
End If
PathAnalysis_Part1 = PathAnalysis_Part1 + "PathAnalysis_Part1 程序结束" + Chr(13) + Chr(10)
End Function
```

Rem 未使用通径分析

```
Public Function PathAnalysis_Unuse()
Rem 剔除数据版本
a1:
Dim Num1 As Long
Num1 = UBound(DataY) - LBound(DataY) + 1

Dim Return_FReport As String
Dim DFR As Long, DF_r As Long
DFR = Element
DF_r = Num1 - Element - 1
PathAnalysis_Main = PathAnalysis_Main + "通径系数差异性检验 F 检验：" + Return_FReport +
Chr(13) + Chr(10)

Dim F() As Double
n = 0

For i = LBound(cB) To UBound(cB)
For l = LBound(cB) To UBound(cB)
If (i <> l) Then
If (i < l) Then
n = n + 1
ReDim Preserve F(n)
F(n) = (cB(i, 1) - cB(l, 1)) ^ 2 / (InversecA(i, i) + InversecA(l, l) - 2 * InversecA(i, l)) / (SSr / (Num1 -
Element - 1))
PathAnalysis_Main = PathAnalysis_Main + "F(" + CStr(n) + " )=" + CStr(F(n)) + Chr(13) + Chr(10)
End If
End If
Next l
Next i
n = 0
```

```
Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj4 As New F001
Return_F001 = obj4.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj4.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万

For i = LBound(F) To UBound(F)
If (F(i) > Return_F001) Then
Return_FReport = Return_FReport + "显著的回归关系" + "F(" + CStr(i) + ")=" + CStr(F(i)) + ">" +
"F0.01=" + CStr(Return_F001)
ElseIf (Return_F001 > F(i) And F(i) > Return_F005) Then
Rem  下面没有验证
Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归
关系" + Chr(13) + Chr(10)
Return_FReport = Return_FReport + "F(" + CStr(i) + ")=" + CStr(F(i)) + ">" + "F0.05=" +
CStr(Return_F005)
ElseIf (Return_F005 > F(i)) Then
Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">"
+ "F(" + CStr(i) + ")=" + CStr(F(i))
End If
Next
PathAnalysis_Main = PathAnalysis_Main + "通径系数差异性检验  F 检验结果" +
Return_FReport + Chr(13) + Chr(10)
PathAnalysis_Main = PathAnalysis_Main +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************************************************************



Rem  不剔除数据版本
a1:
Dim Num1 As Long
Num1 = UBound(DataY) - LBound(DataY) + 1

'Dim Element As Long '16-9-25  测试完成需要修改回来

Element = UBound(a, 1) - LBound(a, 1) + 1 '  应该是多元分析完成后
Dim Return_FReport As String
Dim DFR As Long, DF_r As Long
DFR = Element
DF_r = Num1 - Element - 1
PathAnalysis_Main = PathAnalysis_Main + "通径系数差异性检验  F 检验：" + Return_FReport +
Chr(13) + Chr(10)
```

```
Dim n As Long
Dim F() As Double
n = 0
Dim InverseA() As Double
Dim obj As New Matrix001
obj.InverseMatrix a, InverseA
For i = LBound(a) To UBound(a) ' 改之前是 B ， 16-9-25
For l = LBound(a) To UBound(a)
If (i <> l) Then
If (i < l) Then
n = n + 1
ReDim Preserve F(n)
F(n) = (b(i, 1) - b(l, 1)) ^ 2 / (InverseA(i, i) + InverseA(l, l) - 2 * InverseA(i, l)) / (SSr / (Num1 -
Element - 1))
PathAnalysis_Main = PathAnalysis_Main + "F(" + CStr(n) + " )=" + CStr(F(n)) + Chr(13) + Chr(10)
End If
End If
Next l
Next i
n = 0

Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj4 As New F001
Return_F001 = obj4.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj4.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万

For i = LBound(F) To UBound(F)
If (F(i) > Return_F001) Then
Return_FReport = Return_FReport + "显著的回归关系" + "F(" + CStr(i) + ")=" + CStr(F(i)) + ">" +
"F0.01=" + CStr(Return_F001)
ElseIf (Return_F001 > F(i) And F(i) > Return_F005) Then
Rem 下面没有验证
Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归
关系" + Chr(13) + Chr(10)
Return_FReport = Return_FReport + "F(" + CStr(i) + ")=" + CStr(F(i)) + ">" + "F0.05=" +
CStr(Return_F005)
ElseIf (Return_F005 > F(i)) Then
Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">"
+ "F(" + CStr(i) + ")=" + CStr(F(i))
End If
Next
PathAnalysis_Main = PathAnalysis_Main + "通径系数差异性检验 F 检验结果" +
```

```
Return_FReport + Chr(13) + Chr(10)
PathAnalysis_Main = PathAnalysis_Main +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*********************************************************************************
End Function
```

'PathAnalysis_Main_K 通径分析

Rem 这个版本是通径系数不显著要剔除后的，但是只能到剔除一步。

```
Public Function PathAnalysis_Main_K(ByRef DataX() As Double, ByRef DataY() As Double, ByRef
KickPy As Long) As String
'*******************************************************
Rem 16-9-23 得到 X 有多少组
Dim Num As Long
Num = UBound(DataX, 2) - LBound(DataX, 2) + 1
'*******************************************************
Dim cDataX() As Double
cDataX = DataX
Dim a() As Double, c() As Double, b() As Double, SSr As Double
KickPy = 200 ' 16-9-23 放在这里不知道起作用没有
PathAnalysis_Main_K = PathAnalysis_Main_K + PathAnalysis_Part1(cDataX, DataY, KickPy, a, c, b,
SSr) + Chr(13) + Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Element As Long
'*******************
Rem 程序截断
'Num = 273
'Num = 30
'Return_Ele = 3
'Element = 3
'Dim r() As Double
'PathAnalysis_Main_K = PathAnalysis_Main_K + PathAnalysis_Test1(a, c, r) + Chr(13) + Chr(10)
'KickPy = 1
'*********************
Dim i As Long, l As Long
'*****************************************************
Rem 剔除不显著通径程序 16-9-25
'16-9-25 Debug 检验程序
'For i = LBound(a, 1) To UBound(a, 1)
'For l = LBound(a, 1) To UBound(a, 1)
'Debug.Print "A(" & i & " , " & l & ")=" & a(i, l)
'Next l
'Next i
```

```
'For i = LBound(c, 1) To UBound(c, 1)
'Debug.Print "C(" & i & " , " & 1 & ")=" & c(i, 1)
'Next i
'Debug.Print "KickPy=" & KickPy
'************************************************
Dim cA() As Double, cC() As Double, cB() As Double, InversecA() As Double
Do While KickPy > 0
Dim n As Long
n = 0
PathAnalysis_Main_K = PathAnalysis_Main_K + PathAnalysis_Part3(a, c, DataY, KickPy, cA, cC, _
cB, InversecA, SSr, Element) + Chr(13) + Chr(10)
'*******************************
'MsgBox "调试推出，记住去除"
'GoTo a1
Rem 16-9-6 一元线性分析，就不用下面的偏相关分析内容了，直接跳出
If KickPy = -1 Then
MsgBox "只剩下一个 X，不用通径分析，直接跳出 EXIT FUNCTION PathAnalysis_Main_K"
Exit Function
End If
DoEvents
Loop
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Num1 As Long
'********************
Rem 程序截断
'Num1 = 30
'Dim Return_Ele    As Long
'Return_Ele = 3
'Dim Element As Long
'Element = 3
'Dim r() As Double '16-9-25 使用完改回
'PathAnalysis_Main_K = PathAnalysis_Main_K + PathAnalysis_Test1(a, c, r) + Chr(13) + Chr(10)
'KickPy = 1
'********************
Rem 通径系数差异显著性检验
'*******************************
Rem 剔除数据版本
a1:

Num1 = UBound(DataY) - LBound(DataY) + 1

Dim Return_FReport As String
Dim DFR As Long, DF_r As Long
DFR = Element
```

```
DF_r = Num1 - Element - 1
PathAnalysis_Main_K = PathAnalysis_Main_K + "通径系数差异性检验 F 检验：" +
Return_FReport + Chr(13) + Chr(10)
Dim F() As Double
n = 0

'For i = LBound(cB) To UBound(cB)
'For l = LBound(cB) To UBound(cB)
If (i <> l) Then
If (i < l) Then
n = n + 1
ReDim Preserve F(n)
'F(n) = (cB(i, 1) - cB(l, 1)) ^ 2 / (InversecA(i, i) + InversecA(l, l) - 2 * InversecA(i, l)) / (SSr / (Num1 -
Element - 1))
'PathAnalysis_Main_K = PathAnalysis_Main_K + "F(" + CStr(n) + " )=" + CStr(F(n)) + Chr(13) +
Chr(10)
End If
End If
'Next l
'Next i
n = 0

Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj4 As New F001
'Return_F001 = obj4.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
'Return_F005 = obj4.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万

'For i = LBound(F) To UBound(F)
'If (F(i) > Return_F001) Then
'Return_FReport = Return_FReport + "显著的回归关系" + "F(" + CStr(i) + ")=" + CStr(F(i)) + ">" +
"F0.01=" + CStr(Return_F001)
'ElseIf (Return_F001 > F(i) And F(i) > Return_F005) Then
Rem 下面没有验证
'Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归
关系" + Chr(13) + Chr(10)
'Return_FReport = Return_FReport + "F(" + CStr(i) + ")=" + CStr(F(i)) + ">" + "F0.05=" +
CStr(Return_F005)
'ElseIf (Return_F005 > F(i)) Then
'Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">"
+ "F(" + CStr(i) + ")=" + CStr(F(i))
'End If
'Next
```

```
'PathAnalysis_Main_K = PathAnalysis_Main_K + "通径系数差异性检验 F 检验结果" +
Return_FReport + Chr(13) + Chr(10)
'PathAnalysis_Main_K = PathAnalysis_Main_K +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************************************************************
End Function
```

'PathAnalysis_Main 通径分析主程序

Rem 通径分析主程序

```
Public Function PathAnalysis_Main(ByRef DataX() As Double, ByRef DataY() As Double, ByRef
KickPy As Long) As String
'*******************************************************
Rem 16-9-23 得到 X 有多少组
Dim Num As Long
Num = UBound(DataX, 2) - LBound(DataX, 2) + 1
'*******************************************************
Dim cDataX() As Double
cDataX = DataX
Dim a() As Double, c() As Double, b() As Double, SSr As Double
KickPy = 200 ' 16-9-23 放在这里不知道起作用没有
PathAnalysis_Main = PathAnalysis_Main + PathAnalysis_Part1(cDataX, DataY, KickPy, a, c, b, SSr) +
Chr(13) + Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'*******************
Rem 程序截断
'Num = 273
'Return_Ele = 3
'Dim Element As Long
'Element = 4
'Dim r() As Double
'PathAnalysis_Main = PathAnalysis_Main + PathAnalysis_Test2(a, c, r) + Chr(13) + Chr(10)
'KickPy = 1
'*********************
Dim i As Long, l As Long
'*************************************************
Rem 剔除不显著通径程序 16-9-25
'16-9-25 Debug 检验程序
'For i = LBound(a, 1) To UBound(a, 1)
'For l = LBound(a, 1) To UBound(a, 1)
'Debug.Print "A(" & i & " , " & l & ")=" & a(i, l)
'Next l
'Next i
```

```
'For i = LBound(c, 1) To UBound(c, 1)
'Debug.Print "C(" & i & " , " & 1 & ")=" & c(i, 1)
'Next i
'Debug.Print "KickPy=" & KickPy
'************************************************
'Dim cA() As Double, cC() As Double, cB() As Double, InversecA() As Double, SSr As Double,
Element As Long
'Do While KickPy > 0
'Dim n As Long
'n = 0
'PathAnalysis_Main = PathAnalysis_Main + PathAnalysis_Part3(a, c, DataY, KickPy, cA, cC, _
cB, InversecA, SSr, Element) + Chr(13) + Chr(10)
'*******************************
'MsgBox "调试推出，记住去除"
'GoTo a1


Rem 16-9-6 一元线性分析，就不用下面的偏相关分析内容了，直接跳出
'If KickPy = -1 Then
'MsgBox "只剩下一个 X，不用通径分析，直接跳出 EXIT FUNCTION PathAnalysis_Main"
'Exit Function
'End If
'DoEvents
'Loop
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'********************
Rem 程序截断
'Num1 = 30
'Return_Ele = 3
'Dim element As Long
'Element = 3
'Dim r() As Double '16-9-25 使用完改回
'PathAnalysis_Main = PathAnalysis_Main + PathAnalysis_Test1(a, c, r) + Chr(13) + Chr(10)
'KickPy = 1
'**********************
Rem 通径系数差异显著性检验
'*******************************
Rem 不剔除数据版本
a1:
Dim Num1 As Long
Num1 = UBound(DataY) - LBound(DataY) + 1
'*************
'Num1 = 273 '16-9-26 测试完成需要修改回来
'************
```

```
Dim Element As Long '16-9-25 测试完成需要修改回来
Element = UBound(a, 1) - LBound(a, 1) + 1 ' 应该是多元分析完成后
Dim Return_FReport As String
Dim DFR As Long, DF_r As Long
DFR = Element
DF_r = Num1 - Element - 1
PathAnalysis_Main = PathAnalysis_Main + "通径系数差异性检验 F,T 检验：" + Return_FReport + Chr(13) + Chr(10)
Dim n As Long
Dim F() As Double, t() As Double
n = 0
Dim InverseA() As Double
Dim obj As New Matrix001
obj.InverseMatrix a, InverseA
ReDim Preserve F(LBound(a) To UBound(a), LBound(a) To UBound(a))
ReDim Preserve t(LBound(a) To UBound(a), LBound(a) To UBound(a))
For i = LBound(a) To UBound(a) ' 改之前是 B ， 16-9-25
For l = LBound(a) To UBound(a)
If (i <> l) Then
If (i < l) Then
'n = n + 1
'ReDim Preserve F(n)
'ReDim Preserve T(n)
t(i, l) = (b(i, 1) - b(l, 1)) / (Sqr(SSr / (Num1 - Element - 1)) * Sqr(InverseA(i, i) + InverseA(l, l) - 2 * InverseA(i, l)))
F(i, l) = (b(i, 1) - b(l, 1)) ^ 2 / (InverseA(i, i) + InverseA(l, l) - 2 * InverseA(i, l)) / (SSr / (Num1 - Element - 1))
PathAnalysis_Main = PathAnalysis_Main + "T(" + CStr(i) + "," + CStr(l) + " )=" + CStr(t(i, l)) + Chr(13) + Chr(10)
PathAnalysis_Main = PathAnalysis_Main + "F(" + CStr(i) + "," + CStr(l) + " )=" + CStr(F(i, l)) + Chr(13) + Chr(10)
End If
End If
Next l
Next i
n = 0

Dim Return_F001 As Double
Dim Return_F005 As Double
'Dim df As Long
Dim obj4 As New F001
Return_F001 = obj4.PFaN1N2X2(0.01, 10000000, DFR, DF_r) ' 0.01 用千万
Return_F005 = obj4.PFaN1N2X2(0.05, 1000000, DFR, DF_r)     ' 0.05 用百万
PathAnalysis_Main = PathAnalysis_Main + "F 检验结果:" + Chr(13) + Chr(10)
```

```
For i = LBound(a) To UBound(a) ' 改之前是 B ， 16-9-25
For l = LBound(a) To UBound(a)
If (i <> l) Then
If (i < l) Then
    If (F(i, l) > Return_F001) Then
    Return_FReport = Return_FReport + "显著的回归关系" + "F(" + CStr(i) + "," + CStr(l) + ")=" + CStr(F(i, l)) + ">" + "F0.01=" + CStr(Return_F001)
    ElseIf (Return_F001 > F(i, l) And F(i, l) > Return_F005) Then
    Rem 下面没有验证
    Return_FReport = Return_FReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + Chr(13) + Chr(10)
    Return_FReport = Return_FReport + "F(" + CStr(i) + "," + CStr(l) + ")=" + CStr(F(i, l)) + ">" + "F0.05=" + CStr(Return_F005)
    ElseIf (Return_F005 > F(i, l)) Then
    Return_FReport = Return_FReport + "无显著的回归关系" + "F0.05=" + CStr(Return_F005) + ">" + "F(" + CStr(i) + "," + CStr(l) + ")=" + CStr(F(i, l))
    End If
End If
End If
Next l
Next i
PathAnalysis_Main = PathAnalysis_Main + Return_FReport + Chr(13) + Chr(10)

Dim Return_T001 As Double, Return_T005 As Double
Dim obj5 As New T001
'Dim DF_r As Long
'Dim s() As Double, t() As Double
Return_T001 = obj5.t(0.005, DF_r) '16-9-7 日改
Return_T005 = obj5.t(0.025, DF_r)
Dim Return_TReport As String
PathAnalysis_Main = PathAnalysis_Main + "T 检验结果:" + Chr(13) + Chr(10)
For i = LBound(a) To UBound(a) ' 改之前是 B ， 16-9-25
For l = LBound(a) To UBound(a)
If (i <> l) Then
If (i < l) Then
    If (Abs(t(i, l)) > Return_T001) Then
    Return_TReport = Return_TReport + "显著的回归关系" + "|t(" + CStr(i) + "," + CStr(l) + ")|=" + CStr(t(i, l)) + ">" + "T0.01=" + CStr(Return_T001)
    ElseIf (Return_T001 > Abs(t(i, l)) And Abs(t(i, l)) > Return_T005) Then
    Rem 下面没有验证
    Return_TReport = Return_TReport + "当 AERF=0.05 有显著的回归关系，AERF=0.01 无显著回归关系" + Chr(13) + Chr(10)
    Return_TReport = Return_TReport + "|t(" + CStr(i) + "," + CStr(l) + ")|=" + CStr(t(i, l)) + ">" + "T0.05=" + CStr(Return_T005)
```

```
    ElseIf (Return_T005 > Abs(t(i, l))) Then
    Return_TReport = Return_TReport + "无显著的回归关系" + "T0.05=" + CStr(Return_T005) +
">" + "|t(" + CStr(i) + "," + CStr(l) + ")|=" + CStr(t(i, l))
    End If
End If
End If
Next l
Next i
PathAnalysis_Main = PathAnalysis_Main + Return_TReport + Chr(13) + Chr(10)
PathAnalysis_Main = PathAnalysis_Main +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*************************************************************************
End Function
```

'PathAnalysis_Part3 通径分析 Part3

Rem Part3 给入 A,C 矩阵返回显性检验更改后 A，C 矩阵

```
Public Function PathAnalysis_Part3(ByRef a() As Double, ByRef c() As Double, ByRef DataY() As
Double, _
ByRef KickPy As Long, ByRef Return_A() As Double, ByRef Return_C() As Double, ByRef Return_B()
As Double, _
ByRef Return_InverseA() As Double, ByRef Return_SSr As Double, ByRef Return_Element As Long)
As String
PathAnalysis_Part3 = PathAnalysis_Part3 + "PathAnalysis_Part3 程序开始" + Chr(13) + Chr(10)
        Dim i As Long, l As Long, n As Long
        Dim Element As Long
        Element = UBound(a) - LBound(a) + 1
        'KickPy = 0 '16-9-23 现在应该放在哪？
'*******************************************************
Dim cA() As Double, cC() As Double, cB() As Double
Dim c1A() As Double, c1C() As Double
Dim cNum As Long
'cA = a
ReDim cA(1 To Element - 1, 1 To Element - 1)
ReDim cC(1 To Element - 1, 1)
Dim cElement As Long
cElement = Element
'KickPy = 200 ' 随便给个开始值 16-8-30
Dim LoopTimes As Long
LoopTimes = 0
'*******************************************************
Dim Initial() As Double
ReDim Initial(1 To Element)
```

```
For i = 1 To Element
Initial(i) = i
'Debug.Print "Initial(" & i & ")=" & Initial(i)
Next
'*******************************************************
Dim p1 As Long
p1 = 0 ' 防止 P<>0 16-9-4
p1 = p1 + 1
If KickPy > 0 Then '从 K<>0 改成 K>0
LoopTimes = LoopTimes + 1 ' 记录循环次数，对后面更改数组数有帮助 16-8-30
'cA = A 'cDataX 使用后就不能保存原来数据了，所以启用 c1DataX16-8-30
'cC = C
'cNum = cNum - 1 '每次减少一列数
        '*************************
        Dim obj2 As New Array1D001
        Dim R3() As Double
        obj2.Array1D_RemoveOneNum Initial, KickPy, R3
        Initial = R3

      Dim Num As Long
      Num = UBound(DataY) - LBound(DataY) + 1
      '************
      Rem    16-9-23
      'MsgBox "程序截流"
      'Num = 30
      'PathAnalysis_Part3 = PathAnalysis_Part3 + "截流程序 Num=30" + Chr(13) + Chr(10)
      '************
      'ReDim cA(1 To Num, 1 To Num) As Double
      n = 0 'n 的初始化
      Dim m As Long
      m = 0
      PathAnalysis_Part3 = PathAnalysis_Part3 + "剔除的数据是" + CStr(KickPy) + "组数据" +
Chr(13) + Chr(10)
      PathAnalysis_Part3 = PathAnalysis_Part3 + "新的数据如下" + Chr(13) + Chr(10)
      For l = 1 To (Element - LoopTimes + 1)
            If KickPy <> l Then
            n = n + 1
            End If
            m = 0
            For i = 1 To (Element - LoopTimes + 1)
               If (KickPy <> l And KickPy <> i) Then
               If KickPy <> i Then
               m = m + 1
               End If
```

```
                    cA(m, n) = a(i, l)
                    'Debug.Print "cA(" & m & "," & n & ")=" & "A(" & i & "," & l & ")" & "=" & a(i, l)
                    PathAnalysis_Part3 = PathAnalysis_Part3 + "cA(" + CStr(m) + "," + CStr(n) + ")=" +
CStr(cA(m, n)) + Chr(13) + Chr(10)
                    End If
                Next i

        Next l
        Return_A = cA
        n = 0
        m = 0
        For l = 1 To (Element - LoopTimes + 1)
                If KickPy <> l Then
                n = n + 1
                cC(n, 1) = c(l, 1)
                'Debug.Print "cC(" & n & "," & 1 & ")=" & "c(" & l & "," & 1 & ")" & "=" & c(l, 1)
                PathAnalysis_Part3 = PathAnalysis_Part3 + "Cc(" + CStr(n) + "," + CStr(1) + ")=" +
CStr(cC(n, 1)) + Chr(13) + Chr(10)
                End If



        Next l
        Return_C = cC
        n = 0
End If
'*********************************************************************
          Rem  判断数据是否可以通径分析
          If Element = 1 Then
          MsgBox "自变量数目小于 2.  无法通径分析  Exit Function PathAnalysis_Part3"
          Exit Function
          End If
'*********************
Dim Return_Ele    As Long
Return_Ele = Element - 1
Dim b1() As Double
Dim InverseA() As Double
Dim obj As New Matrix001
obj.InverseMatrix Return_A, InverseA
Return_InverseA = InverseA ' 16-9-25
For i = 1 To Return_Ele
For l = 1 To Return_Ele
PathAnalysis_Part3 = PathAnalysis_Part3 + "InverseA" + CStr(i) + "," + CStr(l) + "=" +
CStr(InverseA(i, l)) + Chr(13) + Chr(10)
Next l
```

```
Next i
obj.MatrixMultiplication InverseA, Return_C, b1
'******************
Rem 16-9-25
Return_B = b1
'******************
Rem    b1()= c()间接通径系数都不存在
PathAnalysis_Part3 = PathAnalysis_Part3 + "直接作用：" + Chr(13) + Chr(10)
For i = 1 To Return_Ele
PathAnalysis_Part3 = PathAnalysis_Part3 + "B" + CStr(i) + "=" + CStr(b1(i, 1)) + Chr(13) + Chr(10)
'Debug.Print b1(i, 1) '不知道是否正确
Next
'**************************************
Rem  间接通径系数计算 16-9-20
Dim P() As Double
ReDim P(1 To Return_Ele, 1 To Return_Ele)
PathAnalysis_Part3 = PathAnalysis_Part3 + "间接作用：" + Chr(13) + Chr(10)
For i = 1 To Return_Ele
For l = 1 To Return_Ele
If (i <> l) Then
P(i, l) = Return_A(i, l) * b1(l, 1)
PathAnalysis_Part3 = PathAnalysis_Part3 + "P(" + CStr(i) + "," + CStr(l) + ",y)=" + CStr(P(i, l)) +
Chr(13) + Chr(10)
End If
Next l
Next i
'**************************************
Rem  决定系数计算 16-9-20
Dim Dy1() As Double
Dim Dy2() As Double
Dim Dy3() As Double
ReDim Dy1(1 To Return_Ele)
ReDim Dy2(1 To Return_Ele, 1 To Return_Ele)



n = 0
For i = 1 To Return_Ele
n = n + 1
ReDim Preserve Dy3(n)
Dy1(i) = b1(i, 1) ^ 2
Dy3(n) = Abs(Dy1(i))
PathAnalysis_Part3 = PathAnalysis_Part3 + "Dy(" + CStr(i) + ")=" + CStr(Dy1(i)) + Chr(13) + Chr(10)
For l = 1 To Return_Ele
If (i <> l) Then
```

```
If (i < l) Then
n = n + 1
ReDim Preserve Dy3(n)
Dy2(i, l) = 2 * Return_A(i, l) * b1(i, 1) * b1(l, 1)
'Dy2(i, l) = 2 * Return_A(i, l) * p(i, l) * p(l, i)
Dy3(n) = Abs(Dy2(i, l))
PathAnalysis_Part3 = PathAnalysis_Part3 + "Dy(" + CStr(i) + "," + CStr(l) + ")=" + CStr(Dy2(i, l)) +
Chr(13) + Chr(10)
End If
End If
Next l
Next i
n = 0

Dim Dye As Double
Dim Pye As Double
'Dim obj2 As New Array1D001
Dim obj3 As New Array2D001
Dye = 1 - (obj2.Array1D_Sum(Dy1) + obj3.Array2D_Sum(Dy2)) ' 16-9-26 Dye=SSr 这个数据是有
问题的
Pye = Sqr(Dye)
PathAnalysis_Part3 = PathAnalysis_Part3 + "Dye=" + CStr(Dye) + Chr(13) + Chr(10)
PathAnalysis_Part3 = PathAnalysis_Part3 + "Pye=" + CStr(Pye) + Chr(13) + Chr(10)
'****************************************
Rem 相关指数计算 16-9-20 Correlation index
Dim CorrelationIndex As Double
CorrelationIndex = obj2.Array1D_Sum(Dy1) + obj3.Array2D_Sum(Dy2) ' 16-9-26 Dye=SSr 这个数
据是有问题的
'**********************************
CorrelationIndex = 0
For i = 1 To Return_Ele
CorrelationIndex = CorrelationIndex + b1(i, 1) * Return_C(i, 1)
Next i
PathAnalysis_Part3 = PathAnalysis_Part3 + "R^2=" + CStr(CorrelationIndex) + Chr(13) + Chr(10)
'**************************************
Rem 16-9-25
Return_SSr = 1 - CorrelationIndex
Return_Element = Return_Ele
'**************************************
Dim Record() As String
Dim RSLU1() As Double, RSUL1() As Double
obj2.Array1D_Sort Dy3, RSLU1, RSUL1
PathAnalysis_Part3 = PathAnalysis_Part3 + "影响 Y 的重要因素：" + Chr(13) + Chr(10)
Dim i1 As Long, l1 As Long
```

```
n = 0
For i = LBound(RSUL1) To UBound(RSUL1)

   For l = LBound(Dy1) To UBound(Dy1)
   If Abs(Dy1(l)) = RSUL1(i) Then
        If i <> UBound(RSUL1) Then
        n = n + 1
        ReDim Preserve Record(n)
        Record(n) = "|dy" + CStr(l) + "|=" + CStr(RSUL1(i))
        PathAnalysis_Part3 = PathAnalysis_Part3 + "|dy" + CStr(l) + "|=" + CStr(RSUL1(i)) + ">"
        Else
        n = n + 1
        ReDim Preserve Record(n)
        Record(n) = "|dy" + CStr(l) + "|=" + CStr(RSUL1(i))
        PathAnalysis_Part3 = PathAnalysis_Part3 + "|dy" + CStr(l) + "|=" + CStr(RSUL1(i)) + Chr(13)
+ Chr(10)
        End If
   End If
   Next l

   For i1 = LBound(Dy1, 1) To UBound(Dy1, 1)
   For l1 = LBound(Dy1, 1) To UBound(Dy1, 1)
   If (i1 <> l1) Then
   If (l1 > i1) Then
      If Abs(Dy2(i1, l1)) = RSUL1(i) Then
          If i <> UBound(RSUL1) Then
          n = n + 1
          ReDim Preserve Record(n)
          Record(n) = "|dy(" + CStr(i1) + "," + CStr(l1) + ")|=" + CStr(RSUL1(i))
          PathAnalysis_Part3 = PathAnalysis_Part3 + "|dy(" + CStr(i1) + "," + CStr(l1) + ")|=" +
CStr(RSUL1(i)) + ">"
          Else
          n = n + 1
          ReDim Preserve Record(n)
          Record(n) = "|dy(" + CStr(i1) + "," + CStr(l1) + ")|=" + CStr(RSUL1(i))
          PathAnalysis_Part3 = PathAnalysis_Part3 + "|dy(" + CStr(i1) + "," + CStr(l1) + ")|=" +
CStr(RSUL1(i)) + Chr(13) + Chr(10)
          End If
      End If
   End If
   End If
   Next l1
   Next i1
```

```
Next i
n = 0
'*****************************************************
Rem 结论解释
Rem 通径系数重要性排列
'PathAnalysis_Part3 = PathAnalysis_Part3 + "~~~~~~~~~~~~~~~结论解释~~~~~~~~~~~~~~~" +
Chr(13) + Chr(10)
'PathAnalysis_Part3 = PathAnalysis_Part3 + "通径系数相对重要性排列： " + Chr(13) + Chr(10)
'Dim RSLU() As Variant, RSUL() As Variant, RSLU2d() As Variant, RSUL2d() As Variant
'Dim r_b1() As Variant
'obj3.Array2D_CDataTypeDtoV b1, r_b1
'obj3.Array2D_SortIncludeEmpty r_b1, RSLU, RSUL, RSLU2d, RSUL2d
'For i = LBound(RSUL) To UBound(RSUL)
   'For l = LBound(RSUL) To UBound(RSUL)
   'If r_b1(l, 1) = RSUL(i) Then
        'If i <> UBound(RSUL) Then
           'If (RSUL(i) > 0) Then
           'PathAnalysis_Part3 = PathAnalysis_Part3 + "正效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
">"
           'ElseIf (RSUL(i) = 0) Then
           'PathAnalysis_Part3 = PathAnalysis_Part3 + "无效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
">"
           'ElseIf (RSUL(i) < 0) Then
           'PathAnalysis_Part3 = PathAnalysis_Part3 + "负效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
">"
           'End If
        'Else
           'If (RSUL(i) > 0) Then
           'PathAnalysis_Part3 = PathAnalysis_Part3 + "正效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
Chr(13) + Chr(10)
           'ElseIf (RSUL(i) = 0) Then
           'PathAnalysis_Part3 = PathAnalysis_Part3 + "无效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
Chr(13) + Chr(10)
           'ElseIf (RSUL(i) < 0) Then
           'PathAnalysis_Part3 = PathAnalysis_Part3 + "负效应 Py" + CStr(l) + "=" + CStr(RSUL(i)) +
Chr(13) + Chr(10)
           'End If
        'End If
   'End If
   'Next l
'Next i
'PathAnalysis_Part3 = PathAnalysis_Part3 + "决定系数最大值为:" + Record(1) + "," + "说明" +
Record(1) + "是影响 Y 值第一因素" + Chr(13) + Chr(10)
```

```
'*************************************************************************
'For i = 1 To Return_Ele

'If r(i, UBound(r, 2)) < 0 And b1(i, 1) > 0 Then
'PathAnalysis_Part3 = PathAnalysis_Part3 + "X" + CStr(i) + "与 Y 的相关系数为负数" + CStr(r(i, UBound(r, 2))) + "；" + "直接作用为 Py" + CStr(i) + "=" + CStr(b1(i, 1)) + "两者符号相反：" + Chr(13) + Chr(10)
'       For l = 1 To Return_Ele
'       If (i <> l) Then
'       PathAnalysis_Part3 = PathAnalysis_Part3 + "因为 P(" + CStr(i) + "," + CStr(l) + ",y)=" + CStr(p(i, l)) + Chr(13) + Chr(10)
'       End If
'      Next l
'       PathAnalysis_Part3 = PathAnalysis_Part3 + "这些间接作用混杂在与 Y 的相关系数中，其中负间接通径系数掩盖了对 Y 的正效应将其夸大为负值" + Chr(13) + Chr(10)
'End If


  'If r(i, UBound(r, 2)) > 0 And b1(i, 1) < 0 Then
  'PathAnalysis_Part3 = PathAnalysis_Part3 + "X" + CStr(i) + "与 Y 的相关系数为正数" + CStr(r(i, UBound(r, 2))) + "；" + "直接作用为 Py" + CStr(i) + "=" + CStr(b1(i, 1)) + "两者符号相反：" + Chr(13) + Chr(10)
  '     For l = 1 To Return_Ele
  '     If (i <> l) Then
  '     PathAnalysis_Part3 = PathAnalysis_Part3 + "因为 P(" + CStr(i) + "," + CStr(l) + ",y)=" + CStr(p(i, l)) + Chr(13) + Chr(10)
  '     End If
  '     Next l
  '     PathAnalysis_Part3 = PathAnalysis_Part3 + "这些间接作用混杂在与 Y 的相关系数中，其中正间接通径系数掩盖了对 Y 的负效应将其夸大为正值" + Chr(13) + Chr(10)
  '     PathAnalysis_Part3 = PathAnalysis_Part3 + Chr(13) + Chr(10)
' End If
'Next i
'PathAnalysis_Part3 = PathAnalysis_Part3 + "相关指数 R^2=" + CStr(CorrelationIndex) + "," + "说明" + CStr(Return_Ele) + "因素决定 Y   " + CStr(CorrelationIndex * 100) + "%" + Chr(13) + Chr(10)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem  通径系数显著性检验
PathAnalysis_Part3 = PathAnalysis_Part3 + "通径系数显著性检验：" + Chr(13) + Chr(10)
Dim Se As Double
Se = Sqr(Dye / (Num - Return_Ele - 1))
PathAnalysis_Part3 = PathAnalysis_Part3 + "Se=" + CStr(Se) + Chr(13) + Chr(10)
Dim S() As Double, t() As Double
ReDim S(1 To Return_Ele)
```

```
ReDim t(1 To Return_Ele)
For i = 1 To Return_Ele
S(i) = Se * Sqr(InverseA(i, i))
PathAnalysis_Part3 = PathAnalysis_Part3 + "S(" + CStr(i) + ")=" + CStr(S(i)) + Chr(13) + Chr(10)
t(i) = b1(i, 1) / S(i)
'PathAnalysis_Part3 = PathAnalysis_Part3 + "t(" + CStr(i) + ")=" + CStr(t(i)) + Chr(13) + Chr(10)
Next i
Dim Return_T001 As Double, Return_T005 As Double
Dim obj5 As New T001
Dim DF_r As Long
DF_r = Num - Return_Ele - 1
'Dim s() As Double, t() As Double
Return_T001 = obj5.t(0.005, DF_r) '16-9-7 日改
Return_T005 = obj5.t(0.025, DF_r)
Dim Return_TReport As String

Dim T1() As Double
n = 0
For i = LBound(t) To UBound(t)
n = n + 1
ReDim Preserve T1(n)
T1(i) = Abs(t(i))
PathAnalysis_Part3 = PathAnalysis_Part3 + "t(" + CStr(i) + ")=" + CStr(t(i)) + Chr(13) + Chr(10)
'Debug.Print "t(" + CStr(i) + ")=" + CStr(t(i))
Next i
n = 0
'************************************************
Dim Remove() As Double '  为剔除 Upi 值
Dim NumRemove As Long
n = 0
PathAnalysis_Part3 = PathAnalysis_Part3 + "以下 T 为绝对值" + Chr(13) + Chr(10)
For i = 1 To Return_Ele
   If (T1(i) > Return_T001) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "显著的回归关系" + "T" + CStr(i) + "="
+ CStr(T1(i)) + ">" + "T0.01=" + CStr(Return_T001) + Chr(13) + Chr(10)
   ElseIf (Return_T001 > T1(i) And T1(i) > Return_T005) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "当 AERF=0.05 有显著的回归关系，
AERF=0.01 无显著回归关系" + "T" + CStr(i) + "=" + CStr(T1(i)) + ">" + "T0.05=" +
CStr(Return_T005) + Chr(13) + Chr(10)
   ElseIf (Return_T005 > T1(i)) Then
   Return_TReport = Return_TReport + "T" + CStr(i) + ":" + "无显著的回归关系" + "T0.05=" +
CStr(Return_T005) + ">" + "T" + CStr(i) + "=" + CStr(T1(i)) + Chr(13) + Chr(10)

   '****************************************
```

```
   Rem 16-9-22 剔除检验不合格的
   NumRemove = NumRemove + 1
   ReDim Preserve Remove(NumRemove)
   Remove(NumRemove) = i
'****************************************
   End If
Next i
PathAnalysis_Part3 = PathAnalysis_Part3 + "T 检验结果" + Chr(13) + Chr(10)
PathAnalysis_Part3 = PathAnalysis_Part3 + Return_TReport + Chr(13) + Chr(10)
PathAnalysis_Part3 = PathAnalysis_Part3 +
"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" + Chr(13) + Chr(10)
'*******************************************************************************
Dim UpArray() As Double
Dim rlu() As Double, rul() As Double
Dim Which() As Long

n = 0

If NumRemove >= 1 Then
    For i = LBound(Remove) To UBound(Remove)
    n = n + 1
    ReDim Preserve UpArray(n)
    UpArray(n) = T1(Remove(i))
    Next i

    n = 0

    For i = LBound(UpArray) To UBound(UpArray)
    PathAnalysis_Part3 = PathAnalysis_Part3 + "UpArray(" + CStr(i) + ")=" + CStr(UpArray(i)) +
Chr(13) + Chr(10)
    Next i

    Dim MixNum As Double
    MixNum = obj2.Array1D_Mix(UpArray) ' 如果有两个数据一样，会给出一个 16-8-30
    For i = LBound(T1) To UBound(T1)
       If MixNum = T1(i) Then ' 如果有两个数据一样，会给出前面的 16-8-30
       KickPy = i
       PathAnalysis_Part3 = PathAnalysis_Part3 + "需要剔除的组数是" + CStr(KickPy) + Chr(13) +
Chr(10)
       End If
    Next i
      PathAnalysis_Part3 = PathAnalysis_Part3 + "PathAnalysis_Part3 结束" + Chr(13) + Chr(10)
Else
KickPy = 0 '16-9-23
```

```
PathAnalysis_Part3 = PathAnalysis_Part3 + "PathAnalysis_Part3 结束" + Chr(13) + Chr(10)
End If

End Function
```

'Times 资料次数的计算

Rem 资料次数的计算

Rem 2019-4-30

```
Public Function Times(ByRef Data() As Double, ByVal ClassNumber As Long, ByRef
ClassLowerLimit() As Double, ByRef ClassUpperLimit() As Double, ByRef Return_Time() As Long)
Dim m As Long, i As Long
ReDim Return_Time(1 To ClassNumber + 1)
For i = LBound(Data()) To UBound(Data())
    For m = 1 To ClassNumber + 1
        If (Data(i) >= ClassLowerLimit(m) And Data(i) < ClassUpperLimit(m)) Then
        Return_Time(m) = Return_Time(m) + 1
        m = ClassNumber + 1 ' 找到后跳出循环，增加速度
        End If
        If m = ClassNumber + 1 Then
            If (Data(i) = ClassUpperLimit(m)) Then '最大数 = 上限的情况
            Return_Time(m) = Return_Time(m) + 1
            m = ClassNumber + 1 ' 找到后跳出循环，增加速度
            End If
        End If
        'Debug.Print "Data(" & i & ")= " & Data(i) & "ClassLowerLimit(" & m & ")=" &
ClassLowerLimit(m) & "ClassUpperLimit(" & m & ")=" & ClassUpperLimit(m)
        If (Data(i) = ClassLowerLimit(m) And Data(i) = ClassUpperLimit(m)) Then
        Return_Time(m) = Return_Time(m) + 1
        m = ClassNumber + 1 ' 找到后跳出循环，增加速度
        End If
    Next m
Next i

Dim TimeNumber    As Long
For i = 1 To UBound(Return_Time)
TimeNumber = TimeNumber + Return_Time(i)
Next i
If TimeNumber = UBound(Data) Then

Else
MsgBox "出错，次数之和不等于总数。次数=" + CStr(TimeNumber) + "总数=" +
CStr(UBound(Data))
```

```
End If
End Function
```

'NormalDistributionPosibilityRectangle 正态分布概率新矩形法

Rem 正态分布概率新矩形法
Rem μ是位置参数
Rem σ是函数胖瘦参数
Rem PrecisionPieces 分块数
Rem μ1,μ2 积分下限，积分上限
Rem 函数会出现范围值越小、分割块数越多越准确。所以还是需要正态分布表加以校对

```
Public Function NormalDistributionPosibilityRectangle(ByVal μ As Double, _
ByVal σ As Double, ByVal PrecisionPieces As Double, ByVal μ1 As Double, _
ByVal μ2 As Double) As Double
Dim X As Double, i As Long, Distance As Double
ReDim ReturnRecordForShape(1 To PrecisionPieces)
Dim NDP As Double, μD As Double
Dim Part1 As Double
μD = μ2 - μ1
Distance = μD / PrecisionPieces
If μ2 <= μ Then '积分上限小于等于正太函数对称中点的情况
Rectangle μ1, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityRectangle
End If
If μ1 >= μ Then '积分下限大于等于正太函数对称中点的情况
Rectangle1 μ2, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityRectangle
End If
If μ1 < μ And μ < μ2 Then '正太函数对称中点在积分上下限之间的情况
μD = μ - μ1
Distance = μD / PrecisionPieces
Rectangle μ1, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityRectangle
Part1 = NormalDistributionPosibilityRectangle
NormalDistributionPosibilityRectangle = 0
μD = μ2 - μ
Distance = μD / PrecisionPieces
Rectangle1 μ2, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityRectangle
NormalDistributionPosibilityRectangle = NormalDistributionPosibilityRectangle + Part1
End If
DoEvents
End Function
```

Rem NormalDistributionPosibilityRectangle 子程序
Rem 正向使用矩形法

```
Private Sub Rectangle(μ1, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityRectangle)
Dim X As Double, i As Long, Part1 As Double
```

```
For i = 0 To PrecisionPieces - 1
X = μ1 + Distance * i
Part1 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X - μ) ^ 2) / (2 * σ ^ 2))
NormalDistributionPosibilityRectangle = NormalDistributionPosibilityRectangle + Part1 * Distance
Next i
End Sub
```

Rem NormalDistributionPosibilityRectangle 子程序

Rem 逆向使用矩形法

```
Private Sub Rectangle1(μ2, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityRectangle)
Dim X As Double, i As Long, Part1 As Double
For i = 0 To PrecisionPieces - 1
X = μ2 - Distance * i
Part1 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X - μ) ^ 2) / (2 * σ ^ 2))
NormalDistributionPosibilityRectangle = NormalDistributionPosibilityRectangle + Part1 * Distance
Next i
End Sub
```

'ArithmeticAverage 算数平均数

Rem 这个更能够代表平均值

Rem 如畜禽、水产养殖的增长率，抗体的滴度，药物的效价，畜禽疾病的潜伏期等

Rem 用几何平均数比用算术平均数更能代表其平均水平。

```
Public Function ArithmeticAverage(ByRef Data() As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
For i = LBound(Data()) To UBound(Data())
Sum1 = Sum1 + Log(Data(i))
n = n + 1
Next i
ArithmeticAverage = Exp(Sum1 / n)
End Function
```

'PopulationVariance 总体方差

Rem 总体方差**σ^2**

```
Public Function PopulationVariance(ByRef Data() As Double, ByRef Return_SumOfSquares As
Double, ByRef Return_PopulationStandardDeviation As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
Dim Average As Double, SumOfSquares As Double
For i = LBound(Data()) To UBound(Data())
Sum1 = Sum1 + Data(i)
n = n + 1
Next i
```

```
Average = Sum1 / n
For i = LBound(Data()) To UBound(Data())
Return_SumOfSquares = Return_SumOfSquares + (Data(i) - Average) ^ 2
Next i
PopulationVariance = Return_SumOfSquares / n
Return_PopulationStandardDeviation = Sqr(PopulationVariance)
End Function
```

'SampleVariance 样本方差

Rem 样本方差 S^2

```
Public Function SampleVariance(ByRef Data() As Double, ByRef Return_SumOfSquares As Double,
ByRef Return_SampleStandardDeviation As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
Dim Average As Double, SumOfSquares As Double
For i = LBound(Data()) To UBound(Data())
Sum1 = Sum1 + Data(i)
n = n + 1
Next i
Average = Sum1 / n
For i = LBound(Data()) To UBound(Data())
Return_SumOfSquares = Return_SumOfSquares + (Data(i) - Average) ^ 2
Next i
SampleVariance = Return_SumOfSquares / (n - 1)
Return_SampleStandardDeviation = Sqr(SampleVariance)
End Function
```

34. MouseWheel【模块】

详细说明：

函数或方法名称	作用
WndProc	程序滚轮前后滚动
ForwardX	导致放大
ForwardY	导致缩小

代码：

Rem 鼠标中间滚轮操作

```
Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" (ByVal lpPrevWndFunc As Long, ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
Public Const GWL_WNDPROC = (-4)
Public Const WM_MOUSEWHEEL = &H20A
Public PrevWndProc As Long
Public Wheel As Double, StayWheel As Boolean, Forward As Boolean, n1 As Long
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem 用于储存获取鼠标位置的值，和滚轮没有关系
Private Type POINTAPI
        X As Long
        Y As Long
End Type
Private Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
Public PA As POINTAPI
Public PA1 As POINTAPI
```

'Windows 程序滚轮前后滚动

```
Public Function WndProc(ByVal hwnd As Long, ByVal uMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long                    '写自己处理鼠标滚动的事件，这里让 Form 上下滚动
Dim t(0 To 1) As Integer
If uMsg = WM_MOUSEWHEEL Then
        If wParam < 0 Then              'backward
```

```
                    'If ((Wheel + 1) * ScalingPlateau - MaxX1 < 0) Then
                    Wheel = Wheel - 1
                    StayWheel = True
                    Forward = False
                    'Else
                      'ScalingPlateau = ScalingPlateau * 10
                      'Wheel = Wheel / 10
                    'End If
          Else      'forforward
                    'If Wheel < 1000 Then

                    If MaxX1 >= MaxY1 Then
                    Call ForwardX
                    Else
                    Call ForwardY
                    End If



          End If
   Else
WndProc = CallWindowProc(PrevWndProc, hwnd, uMsg, wParam, lParam) '让 Windows 处理其他事件
   End If

End Function
```

Rem WndProc 子程序

```
Sub ForwardX()
  If ((Wheel + 1) * ScalingPlateau - MaxX1 < 0) Then '防止放大超过极限，导致缩小
                    Wheel = Wheel + 1
                    StayWheel = True
                    Forward = True
                      '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                       Rem  用于原位置鼠标未移动，滚轮滚动原地缩放
                       n1 = n1 + 1
                       If n1 = 1 Then
                       GetCursorPos PA '  容易出错点，分辨率改变或者其他
                       Else
                       GetCursorPos PA1
                       n1 = 0
                       End If
                       '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                    Else
                     ScalingPlateau = ScalingPlateau / 10
                     'Wheel = Wheel * ((1 - ScalingPlateau / 50)) * 10
```

```
                    Wheel = Wheel * 10
                    'Wheel = Wheel
                End If
                'End If
End Sub
```

Rem WndProc 子程序

```
Sub ForwardY()
  If ((Wheel + 1) * ScalingPlateau - MaxY1 < 0) Then '防止放大超过极限，导致缩小
                Wheel = Wheel + 1
                StayWheel = True
                Forward = True
                  '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                    Rem 用于原位置鼠标未移动，滚轮滚动原地缩放
                    n1 = n1 + 1
                    If n1 = 1 Then
                    GetCursorPos PA ' 容易出错点，分辨率改变或者其他
                    Else
                    GetCursorPos PA1
                    n1 = 0
                    End If
                    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                Else
                  ScalingPlateau = ScalingPlateau / 10
                  'Wheel = Wheel * ((1 - ScalingPlateau / 50)) * 10
                  Wheel = Wheel * 10
                  'Wheel = Wheel
                End If
                'End If
End Sub
```

35. Angle(DX7 飞船)

详细说明：

函数或方法名称	作用
GetAngel	获取角度
GetAngelVBFrom	获取角度从 VB 窗口
AngelChangePic	随角度改变图片

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*********************************************************************************
'名称 = Angle.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 12 月 31 日
'简介 = 角度关联图片类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=None
'操作历史：
'*********************************************************************************
```

'GetAngel 获取角度

```
Rem http://www.360doc.com/content/15/0906/06/9261962_497159278.shtml
Rem 求知 881 2015-09-06 / 2020-4-20
Public Function GetAngel(ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long)
As Double
Const PI    As Double = 3.1415926
Dim Judge    As Boolean
    'VB 中默认 boolean 值为 False，这里如果是默认（即 judge=False），结果永远等于 0
Judge = True
Dim Ang As Long
Select Case Judge
        Case X1 < X2 And Y1 < Y2
            Ang = Atn((Y2 - Y1) / (X2 - X1)) * 180 / PI
        Case X1 < X2 And Y1 = Y2
            Ang = 0
        Case X1 < X2 And Y2 > Y1 'cuo wo
```

```
            Ang = Atn((Y2 - Y1) / (X2 - X1)) * 180 / PI + 360
        Case X1 = X2 And Y1 < Y2
            Ang = 90
        Case X1 = X2 And Y1 = Y2
        MsgBox "两个点重复，请从新输入！"
        Case X1 = X2 And Y1 > Y2
            Ang = 270
        Case X1 > X2 And Y1 < Y2
          Ang = Atn((Y2 - Y1) / (X2 - X1)) * 180 / PI + 180
        Case X1 > X2 And Y1 = Y2
            Ang = 180
        Case X1 > X2 And Y1 > Y2
        Ang = Atn((Y2 - Y1) / (X2 - X1)) * 180 / PI + 180
    End Select
GetAngel = Ang
End Function
```

'GetAngelVBFrom 获取角度从 VB 窗口

Rem 这个坐标的角度是按照 VB 窗口数值确定的:X 坐标轴正常，Y 坐标轴翻转
Rem X1,Y1 起始位置
Rem X2,Y2 结束位置
Rem GetAngelVBFrom=返回角度值,这个值和我们链接 X1,X2,Y1,Y2 后的直线所成的角度是一致的

```
Public Function GetAngelVBFrom(ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Double
Const PI As Double = 3.1415926
Dim Judge  As Boolean  'VB 中默认 boolean 值为 False，这里如果是默认（即 judge=False），结果永远等于 0
Judge = True
Dim Ang As Double
Select Case Judge
        Case X1 < X2 And Y1 < Y2 '第 4 象限
            Ang = 360 - Atn((Y2 - Y1) / (X2 - X1)) * 180 / PI
        Case X1 < X2 And Y1 = Y2
            Ang = 0
        Case X1 < X2 And Y1 > Y2 '第 1 象限
            Ang = -Atn((Y2 - Y1) / (X2 - X1)) * 180 / PI
        Case X1 = X2 And Y1 < Y2
            Ang = 270
        Case X1 = X2 And Y1 = Y2
            'MsgBox "两个点重复，请从新输入！"
        Case X1 = X2 And Y1 > Y2
```

```
            Ang = 90
        Case X1 > X2 And Y1 < Y2 '第 3 象限
            Ang = -(Atn((Y2 - Y1) / (X2 - X1)) * 180 / PI - 180)
        Case X1 > X2 And Y1 = Y2
            Ang = 180
        Case X1 > X2 And Y1 > Y2 '第 2 象限
            Ang = 180 - Atn((Y2 - Y1) / (X2 - X1)) * 180 / PI
    End Select
GetAngelVBFrom = Ang
End Function
```

'AngelChangePic 随角度改变图片

```
Public Function AngelChangePic(Which As Long, Angle() As Double, CharPicPath() As String,
StopSingle As Boolean) As Double
Dim Count As Long
For i = 1 To 72
    Count = Count + 1
    If Angle(Which) >= (i * 5 - 2.5) And Angle(Which) <= (i * 5 + 2.5) Then
        CharPicPath(Which) = App.Path + "\Pic\Ship\Test\" + CStr(i) + ".jpg"
    End If
Next i
End Function
```

36. CharactorMove(DX7 飞船)

详细说明：

函数或方法名称	作用
CharactorStop	角色区域判定移动停止
CharactorAccelerate	角色匀速加速直到最大速度
CharactorDecelerateOld_Unuse	角色减速程序（未使用）
CharactorDecelerate	角色减速，按照距离递减
CharactorMoveAngleAndStopPlace	角色的角度和角色停止区域
ResetCharactorChangeInfor	重置角色改变信息
ResetCharactorChangeInforMouseDown	增加修改目的地角色移动初始化，作用防止点击后停止不动

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
```

```
'********************************************************************************
'名称 = CharactorMove.Cls
'作者 = 张宏
'最后修改时间 = 2021 年 12 月 31 日
'简介 = 角色移动类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=None
'操作历史：
'********************************************************************************
```

```
Dim ObjContanct As New Contanct001
Dim ObjMoveDirection As New MoveDirection
Dim ObjAngel As New Angle
```

'CharactorStop 角色区域判定移动停止

```
Rem CharactorStop 角色区域判定移动停止
Rem Which=角色
Rem CharactorRECT=角色图片显示部分矩形
Rem StopRECT=停止区域矩形
Rem CharX()=角色位置 X
Rem CharY()=角色位置 Y
Rem CharPicWidth()=角色总图片宽
Rem CharPicHeight()=角色总图片高
Rem CharTotalPartX()=角色图片横向分块数
Rem CharTotalPartY()=角色图片纵向分块数
Rem ReturnSpeedX()=返回横向移动速度
Rem ReturnSpeedY()=返回纵向移动速度
Rem CharactorStop=True 停止信号
```

```
Public Function CharactorStop(Which As Long, CharactorRECT As RECT, StopRECT As RECT, _
CharFullDistance() As Double, CharMovedDistanceBeforDecelerate() As Double,
CharDecelerateDistance() As Double, CharX() As Single, _
CharY() As Single, CharPicWidth() As Long, CharPicHeight() As Long, CharTotalPartX() As Long,
CharTotalPartY() As Long, _
ReturnSpeedX() As Double, ReturnSpeedY() As Double) As Boolean
If CharactorStop = False Then
CharactorRECT.Left = CharX(Which)
CharactorRECT.Top = CharY(Which)
CharactorRECT.Right = CharX(Which) + CharPicWidth(Which) / CharTotalPartX(Which)
```

```
CharactorRECT.Bottom = CharY(Which) + CharPicHeight(Which) / CharTotalPartY(Which)
If CharFullDistance(Which) > 100 Then '由于减速距离和实际减速距离有偏差，所以要加上调整系数 80，30
    If CharFullDistance(Which) <= CharMovedDistanceBeforDecelerate(Which) + CharDecelerateDistance(Which) + 80 Then
    CharactorStop = True
    End If
    Else
    If CharFullDistance(Which) <= CharMovedDistanceBeforDecelerate(Which) + CharDecelerateDistance(Which) + 30 Then
    CharactorStop = True
    End If
End If
End If
End Function
```

'CharactorAccelerate 角色匀速加速直到最大速度

Rem 角色匀速加速直到最大速度
Rem Which=角色
Rem CharAngle()=角色运动方向
Rem CharFullSpeed()=角色最大速度
Rem CharSpeedX()=角色水平速度
Rem CharSpeedY()=角色纵向速度
Rem CharAccelerate()=加速度大小

```
Public Function CharactorAccelerate(Which As Long, CharAngle() As Double, CharFullSpeed() As Double, _
CharSpeedX() As Double, CharSpeedY() As Double, CharSpeedNow() As Double, CharAccelerate() As Double, CharDecelerate() As Double, StopSingle As Boolean, _
CharFullDistance() As Double, CharMovedDistanceBeforDecelerate() As Double, CharDecelerateDistance() As Double, CharBeforMoveX() As Single, _
CharBeforMoveY() As Single, CharX() As Single, CharY() As Single, SpeedMax As Double) As Boolean
Static Time As Long
Dim T1 As Double

If StopSingle = False Then
    If CharSpeedX(Which) = 0 And CharSpeedY(Which) = 0 Then
    Time = 0 '角色停止时停止时间
    End If
    Time = Time + 1
    T1 = Time * (CLng(Frm_Main.TimerMove.Interval) / 1000) '加速和匀速运动的时间长度
    CharSpeedNow(Which) = CharSpeedNow(Which) + CharAccelerate(Which) * T1 '逐渐增加
```

```
速度
    If CharSpeedNow(Which) >= CharFullSpeed(Which) Then '当速度达到最大值得时候
    CharSpeedNow(Which) = CharFullSpeed(Which) '速度保持最大速度
    End If
    CharMovedDistanceBeforDecelerate(Which) = Sqr((CharBeforMoveX(Which) - CharX(Which))
^ 2 + (CharBeforMoveY(Which) - CharY(Which)) ^ 2)
    If CharSpeedNow(Which) < CharFullSpeed(Which) Then
    CharDecelerateDistance(Which) = CharSpeedNow(Which) ^ 2 / (2 * CharDecelerate(Which))
'未达到最大速度的减速距离
    SpeedMax = CharSpeedNow(Which)
    Else '达到最大速度时需要的减速距离
    CharDecelerateDistance(Which) = CharFullSpeed(Which) ^ 2 / (2 * CharDecelerate(Which))
    SpeedMax = CharFullSpeed(Which)
    End If
    ObjMoveDirection.MoveDirection CharSpeedNow(Which), CharAngle(Which),
CharSpeedX(Which), CharSpeedY(Which)
End If
End Function
```

'CharactorDecelerateOld_Unuse 角色减速程序（未使用）

Rem 角色减速程序

Rem 原来的是按照时间导致速度递减

```
Public Sub CharactorDecelerateOld_Unuse(Which As Long, CharAngle() As Double,
CharSpeedNow() As Double, _
CharSpeedX() As Double, CharSpeedY() As Double, CharDecelerate() As Double, StopSingle As
Boolean)
Static Time As Long
Dim T2 As Double
If StopSingle = True Then
    If CharSpeedX(Which) = 0 And CharSpeedY(Which) = 0 Then
    Time = 0 '角色停止时停止时间
    End If
    If CharSpeedX(Which) <> 0 And CharSpeedY(Which) <> 0 Then
    Time = Time + 1
    T2 = Time * (CLng(Frm_Main.TimerMove.Interval) / 1000) '减速运动时间长度
    CharSpeedNow(Which) = CharSpeedNow(Which) - CharDecelerate(Which) * T2 '逐渐减少速
度
    If CharSpeedNow(Which) < 0 Then
    CharSpeedNow(Which) = 0
    End If
    ObjMoveDirection.MoveDirection CharSpeedNow(Which), CharAngle(Which),
CharSpeedX(Which), CharSpeedY(Which)
```

```
    End If
End If
End Sub
```

'CharactorDecelerate 角色减速，按照距离递减

Rem 角色减速，按照距离递减

```
Public Sub CharactorDecelerate(Which As Long, CharAngle() As Double, CharSpeedNow() As
Double, _
CharSpeedX() As Double, CharSpeedY() As Double, CharDecelerate() As Double, StopSingle As
Boolean, _
CharFullDistance() As Double, CharBeforMoveX() As Single, CharBeforMoveY() As Single, _
CharX() As Single, CharY() As Single, CharStopMinSpeed() As Double, SpeedMax As Double)
Static Time As Long
Static SpeedC As Double '变化的速度
Dim RemainDistance As Double '到达终点前的距离
If StopSingle = False Then
SpeedC = SpeedMax
End If
If StopSingle = True Then
    If CharSpeedX(Which) = 0 And CharSpeedY(Which) = 0 Then
    Time = 0 '角色停止时停止时间
    End If
    If CharSpeedX(Which) <> 0 And CharSpeedY(Which) <> 0 Then
    Time = Time + 1
    RemainDistance = CharFullDistance(Which) - Sqr((CharBeforMoveX(Which) - CharX(Which))
^ 2 + (CharBeforMoveY(Which) - CharY(Which)) ^ 2)
            If RemainDistance > 0 Then
            SpeedC = SpeedC - SpeedC / RemainDistance '逐渐减少速度
            CharSpeedNow(Which) = SpeedC
            'Debug.Print CharSpeedNow(Which)
                If SpeedC <= CharStopMinSpeed(Which) Then
                SpeedC = 0
                CharSpeedNow(Which) = 0
                End If
            Else
            CharSpeedNow(Which) = 0
            End If
    ObjMoveDirection.MoveDirection CharSpeedNow(Which), CharAngle(Which),
CharSpeedX(Which), CharSpeedY(Which)
    End If
End If
End Sub
```

'CharactorMoveAngleAndStopPlace 角色的角度和角色停止区域

Rem 角色的角度和角色停止区域
Rem CharChoice()=角色选择
Rem CharX()=角色开始 **X** 位置
Rem CharY()=角色开始 **Y** 位置
Rem StopRECT=角色停止区域
Rem X，Y=鼠标点击位置，既停止位置

```
Public Sub CharactorMoveAngleAndStopPlace(CharAngle() As Double, CharChoice() As Long,
CharX() As Single, CharY() As Single, _
StopRECT As RECT, X As Single, Y As Single, StopSingle As Boolean, CharFullDistance() As Double)
    Dim i As Long
    For i = LBound(CharChoice) To UBound(CharChoice)
        If CharChoice(i) <> 0 Then
        CharAngle(CharChoice(i)) = ObjAngel.GetAngelVBFrom(CharX(CharChoice(i)),
CharY(CharChoice(i)), X, Y)
        CharFullDistance(i) = Sqr((CharX(i) - X) ^ 2 + (CharY(i) - Y) ^ 2)
        End If
    Next i
    Debug.Print CharFullDistance(1)
        StopRECT.Left = X
        StopRECT.Top = Y
        StopRECT.Right = X + 48
        StopRECT.Bottom = Y + 64
        StopSingle = False
    End Sub
```

'ResetCharactorChangeInfor 重置角色改变信息

Rem 重置角色改变信息

```
Public Sub ResetCharactorChangeInfor(Which, CharFullDistance() As Double,
CharMovedDistanceBeforDecelerate() As Double, _
CharDecelerateDistance() As Double, CharBeforMoveX() As Single, CharBeforMoveY() As Single, _
CharSpeedNow() As Double, CharX() As Single, CharY() As Single)
If CharSpeedNow(Which) = 0 Then
CharFullDistance(Which) = 0
CharMovedDistanceBeforDecelerate(Which) = 0
CharDecelerateDistance(Which) = 0
CharBeforMoveX(Which) = CharX(Which)
CharBeforMoveY(Which) = CharY(Which)
Frm_Main.TimerMove.Enabled = False
Debug.Print "ResetCharactorChangeInfor"
```

```
End If
End Sub
```

'ResetCharactorChangeInforMouseDown 增加修改目的地角色移动初始化，作用防止点击后停止不动

Rem 增加修改目的地角色移动初始化，作用防止点击后停止不动

```
Public Sub ResetCharactorChangeInforMouseDown(Which, CharFullDistance() As Double,
CharMovedDistanceBeforDecelerate() As Double, _
CharDecelerateDistance() As Double, CharBeforMoveX() As Single, CharBeforMoveY() As Single, _
CharSpeedNow() As Double, CharX() As Single, CharY() As Single)
CharFullDistance(Which) = 0
CharMovedDistanceBeforDecelerate(Which) = 0
CharDecelerateDistance(Which) = 0
CharBeforMoveX(Which) = CharX(Which)
CharBeforMoveY(Which) = CharY(Which)
Frm_Main.TimerMove.Enabled = False
'Debug.Print "ResetCharactorChangeInforMouseDown"
End Sub
```

37. Contanct(DX7 飞船)

详细说明：

函数或方法名称	作用
Contanct_SpotAndRectangle	点和矩形是否重叠
Contanct_RectangleAndRectangle	矩形和矩形是否重叠
Contanct_RectangleAndRectangleRECT	矩形和矩形是否重叠
Contanct_SpotAndCircle	点和圆是否重叠
Contanct_CircleAndCircle	圆和圆是否重叠

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
```

'*

'名称 = Contanct.Cls

'作者 = 张宏

'最后修改时间 = 2020 年 4 月 23 日

'简介 = 物体和物体是否相互重叠
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=None
'操作历史：
'**********

'Contanct_SpotAndRectangle 点和矩形是否重叠

Rem 点和矩形是否重叠
Rem SpotX,SpotY 点的位置
Rem RectangleX,Y2 矩形左上点的位置
Rem RectangleWidth,Height2 矩形的宽和高

```
Public Function Contanct_SpotAndRectangle(ByVal SpotX As Double, ByVal SpotY As Double, _
ByVal RectangleX As Double, ByVal RectangleY As Double, ByVal RectangleWidth As Double, ByVal
RectangleHeight As Double) As Boolean
If RectangleX <= SpotX And SpotX <= RectangleX + RectangleWidth And RectangleY <= SpotY And
SpotY <= RectangleY + RectangleHeight Then
Contanct_SpotAndRectangle = True
Else
Contanct_SpotAndRectangle = False
End If
End Function
```

'Contanct_RectangleAndRectangle 矩形和矩形是否重叠

Rem 矩形和矩形是否重叠
Rem Rectangle1X,Rectangle1Y 矩形 **1** 左上角点
Rem Rectangle1Width,Rectangle1Height 矩形 **1** 宽度高度
Rem Rectangle2X,Rectangle2Y 矩形 **2** 左上角点
Rem Rectangle2Width,Rectangle2Height 矩形 **2** 宽度高度

```
Public Function Contanct_RectangleAndRectangle(ByVal Rectangle1X As Double, ByVal
Rectangle1Y As Double, _
ByVal Rectangle1Width As Double, ByVal Rectangle1Height As Double, ByVal Rectangle2X As
Double, ByVal Rectangle2Y As Double, _
ByVal Rectangle2Width As Double, ByVal Rectangle2Height As Double) As Boolean
Contanct_RectangleAndRectangle = False
If (Rectangle2X <= Rectangle1X + Rectangle1Width And Rectangle1X + Rectangle1Width <=
Rectangle2X + Rectangle2Width) Or (Rectangle2X <= Rectangle1X And Rectangle1X <=
Rectangle2X + Rectangle2Width) Then
If (Rectangle2Y <= Rectangle1Y + Rectangle1Height And Rectangle1Y + Rectangle1Height <=
Rectangle2Y + Rectangle2Height) Or (Rectangle2Y <= Rectangle1Y And Rectangle1Y <=
Rectangle2Y + Rectangle2Height) Then
```

```
Contanct_RectangleAndRectangle = True
End If
End If
End Function
```

'Contanct_RectangleAndRectangleRECT 矩形和矩形是否重叠

Rem 矩形和矩形是否重叠
Rem Rectangle1X,Rectangle1Y 矩形 **1** 左上角点
Rem Rectangle1Width,Rectangle1Height 矩形 **1** 宽度高度
Rem Rectangle2X,Rectangle2Y 矩形 **2** 左上角点
Rem Rectangle2Width,Rectangle2Height 矩形 **2** 宽度高度

```
Public Function Contanct_RectangleAndRectangleRECT(ByRef Rectangle1RECT As RECT, ByRef
Rectangle2RECT As RECT) As Boolean
Contanct_RectangleAndRectangleRECT = False
If (Rectangle2RECT.Left <= Rectangle1RECT.Right And Rectangle1RECT.Right <=
Rectangle2RECT.Right) Or (Rectangle2RECT.Left <= Rectangle1RECT.Left And Rectangle1RECT.Left
<= Rectangle2RECT.Right) Then
If (Rectangle2RECT.Top <= Rectangle1RECT.Bottom And Rectangle1RECT.Bottom <=
Rectangle2RECT.Bottom) Or (Rectangle2RECT.Top <= Rectangle1RECT.Top And Rectangle1RECT.Top
<= Rectangle2RECT.Bottom) Then
Contanct_RectangleAndRectangleRECT = True
End If
End If
End Function
```

'Contanct_SpotAndCircle 点和圆是否重叠

Rem 点和圆是否重叠
Rem SpotX,SpotY 点的位置
Rem CentrePointX,CentrePointY 圆的圆心
Rem CircleRadius 圆的半径

```
Public Function Contanct_SpotAndCircle(ByVal SpotX As Double, ByVal SpotY As Double, _
ByVal CentrePointX As Double, ByVal CentrePointY As Double, ByVal CircleRadius As Double) As
Boolean
If CentrePointX - CircleRadius <= SpotX And SpotX <= CentrePointX + CircleRadius And
CentrePointY - CircleRadius <= SpotY And SpotY <= CentrePointY + CircleRadius Then
Contanct_SpotAndCircle = True
Else
Contanct_SpotAndCircle = False
End If
End Function
```

'Contanct_CircleAndCircle 圆和圆是否重叠

Rem 圆和圆是否重叠
Rem CentrePoint1X,CentrePoint1Y 一号圆的圆心位置
Rem Radius1 一号圆半径
Rem CentrePoint2X,CentrePoint2Y 二号圆的圆心位置
Rem Radius2 二号圆半径

```
Public Function Contanct_CircleAndCircle(ByVal CentrePoint1X As Double, ByVal CentrePoint1Y
As Double, ByVal Radius1 As Double, _
ByVal CentrePoint2x As Double, ByVal CentrePoint2Y As Double, ByVal Radius2 As Double) As
Boolean
Contanct_CircleAndCircle = False
Dim CharFullDistance As Double
CharFullDistance = Sqr((CentrePoint2x - CentrePoint1X) ^ 2 + (CentrePoint2Y - CentrePoint1Y) ^ 2)
If Radius1 + Radius2 >= CharFullDistance Then
'If (CentrePoint2x - Radius2 <= CentrePoint1X + Radius1 And CentrePoint1X + Radius1 <=
CentrePoint2x + Radius2) Or (CentrePoint2x - Radius2 <= CentrePoint1X - Radius1 And
CentrePoint1X - Radius1 <= CentrePoint2x + Radius2) Then
'If (CentrePoint2Y - Radius2 <= CentrePoint1Y + Radius1 And CentrePoint1Y + Radius1 <=
CentrePoint2Y + Radius2) Or (CentrePoint2Y - Radius2 <= CentrePoint1Y - Radius1 And
CentrePoint1Y - Radius1 <= CentrePoint2Y + Radius2) Then
Contanct_CircleAndCircle = True
'End If
End If
End Function
```

38. DX7ZH(DX7 飞船)

详细说明：

函数或方法名称	作用
ExModeActive	回复可能丢失的画面 SpriteSurfaceRestorePic 子程序
UserInitializeDX7	初始化 DX7
InitSurfaceLake	初始化背景面初始化 UserInitializeDX7 子程序
UserDrawLakeSurface	创建角色的背景图面
InitSurfaceSprite	初始化角色面，然后创建角色面 SpriteSurfaceRestorePic 子程序
RepaintEntireBackground	把 Lake 背景画在缓存中 InitSurfaceSprite 子程序
UserDrawSpriteSurface	绘制角色的主要页面
SpriteSurfaceDrawBackBuffer	绘制角色的主要页面 UserDrawSpriteSurface 子程序
UserDrawPrimerySurface	把缓冲面画到主页面
UserQuitDX7	卸载 DX7
SpriteSurfaceSetTransparentColor	设置角色透明色 UserDrawSpriteSurface 子程序
SpriteSurfaceCreateSprite	创建一个角色 UserDrawSpriteSurface 子程序
SpriteSurfaceMove	角色移动 UserDrawSpriteSurface 子程序
SpriteSurfaceShowWhichPart	显示角色图片的哪一部分 UserDrawSpriteSurface 子程序
UserDrawSpriteSurfaceFast	绘制角色的主要页面

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'********************************************************************************
'名称 = DX7ZH.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 4 月 20 日
'简介 =调用 DX7ZH 的时候申明放在窗体最上边
'声明 = 程序中的英文只做代号，不是标准翻译
' 使用方法： 1.Dim DX7ZH as new object 放在窗体声明部分
'           2.UserInitializeDX7 首先在 FORM_LOAD 中使用
'           3.UserDrawSpriteSurface 放在 TIMER1 下，可以放多个用在画多个图像
'           4.UserDrawPrimerySurface 放在 TIMER1 所有 UserDrawSpriteSurface 的后面
'           5.UserDrawLakeSurface 放在 Timer2 下，可以和 Timer1 一起启用不更换背景
```

不用

' 6.UserQuitDX7 放在 Form_Unload 用于卸载 DX7

'

Rem 低速优化：位移小于 1 像素的时候不重新绘。2018-12-1 提出未完成

```
Dim DirectX As DirectX7
Dim DirectDraw As DirectDraw7
Dim BackBuffer As DirectDrawSurface7 '缓冲面
Dim LakeSurface As DirectDrawSurface7 '背景面
Dim SpriteSurface As DirectDrawSurface7 '角色面
Dim PrimSurface As DirectDrawSurface7 '显示面
'Dim ChangeLakeSurface As DirectDrawSurface7

Dim Clipper As DirectDrawClipper '裁剪器
'Dim Direct3D As Direct3D7
'Dim Device As Direct3DDevice7
Dim ddsdLake As DDSURFACEDESC2
Dim ddsdSprite As DDSURFACEDESC2
'Dim ddsdChangeLake As DDSURFACEDESC2

Dim lastX() As Long '角色图片上回最终位置，用在清除原图片再画程序
Dim lastY() As Long '角色图片上回最终位置，用在清除原图片再画程序
Dim rLake As RECT '角色背景显示矩形
Dim rSprite As RECT '角色显示矩形
Dim rBackBuffer As RECT '缓存显示矩形
Dim rTarget As RECT
Dim Fps As Single '画面显示帧数
Dim bInit As Boolean 'Blt 只翻动背景 遗留函数
Dim running As Boolean '非外接只内部循环显示图片时候 遗留函数
```

Rem SpriteSurfaceRestorePic 子程序

Rem 回复可能丢失的画面

```
Function ExModeActive() As Boolean
    Dim TestCoopRes As Long
    TestCoopRes = DirectDraw.TestCooperativeLevel
    If (TestCoopRes = DD_OK) Then
        ExModeActive = True
    Else
        ExModeActive = False
    End If
End Function
```

'UserInitializeDX7 初始化 DX7

Rem 初始化只需要一次。不放在 TIMER 下。

Rem BackPictureObject　要显示的目的控件

Rem FileNameBack　　初始的背景图案位置

```
Sub UserInitializeDX7(BackPictureObject As Object, FileNameBack As String)
    Set DirectX = New DirectX7
    Dim DDsd1 As DDSURFACEDESC2
    Set DirectDraw = DirectX.DirectDrawCreate("")
    DirectDraw.SetCooperativeLevel BackPictureObject.hWnd, DDSCL_NORMAL

    '创建主界面==============================================
    With DDsd1 '
        .lFlags = DDSD_CAPS '
        .ddsCaps.lCaps = DDSCAPS_PRIMARYSURFACE
        '.lBackBufferCount = 1'全屏
    End With '

    Set PrimSurface = DirectDraw.CreateSurface(DDsd1) '
    '=====================================================
    '创建缓存界面============================================
    With DDsd1 '
        .lFlags = DDSD_CAPS Or DDSD_WIDTH Or DDSD_HEIGHT '
        .ddsCaps.lCaps = DDSCAPS_OFFSCREENPLAIN Or DDSCAPS_SYSTEMMEMORY '
        .lWidth = BackPictureObject.ScaleWidth      '缓存画面大小为整个背景图片
        .lHeight = BackPictureObject.ScaleHeight '
    End With '
    Set BackBuffer = DirectDraw.CreateSurface(DDsd1) '
    '=====================================================
    '创建裁剪器==============================================
    Set Clipper = DirectDraw.CreateClipper(0)
    Clipper.SetHWnd BackPictureObject.hWnd
    PrimSurface.SetClipper Clipper
    '=====================================================
    rBackBuffer.Bottom = DDsd1.lHeight
    rBackBuffer.Right = DDsd1.lWidth
    '创建可移动图片==========================================
    InitSurfaceLake BackPictureObject, FileNameBack, rLake

    ' =====================================================
    'RunDX = True '这个可以用在外部窗体调用，显示 DX7 已经可用
    '类模块内运行============================================
    'running = True
    'Do While running
        'UserDrawSpriteSurface BackPictureObject, FileNameBack, FileNameSprite
        'DoEvents
    'Loop
```

```
    ' ====================================================


End Sub
```

Rem InitSurfaceLake 初始化背景面初始化 UserInitializeDX7 子程序
Rem 背景面初始化

```
Sub InitSurfaceLake(BackPictureObject As Object, FileNameBack As String, rLake As RECT)
    '创建角色的背景图面=======================================
    ddsdLake.lFlags = DDSD_CAPS Or DDSD_WIDTH Or DDSD_HEIGHT
    ddsdLake.ddsCaps.lCaps = DDSCAPS_OFFSCREENPLAIN
    ddsdLake.lWidth = BackPictureObject.ScaleWidth
    ddsdLake.lHeight = BackPictureObject.ScaleHeight
    Rem 实验：改为可视的大小,图像拉伸变形。
    'ddsdLake.lWidth = Frm_Main.Pic_Contain.ScaleWidth
    'ddsdLake.lHeight = Frm_Main.Pic_Contain.ScaleHeight
    Set LakeSurface = DirectDraw.CreateSurfaceFromFile(FileNameBack, ddsdLake)
    '====================================================
    '把背景复制到合成图面=====================================
     'copy the background to the compositing surface
    RepaintEntireBackground '把 Lake 背景画在缓存中
    '====================================================
    rLake.Bottom = ddsdLake.lHeight
    rLake.Right = ddsdLake.lWidth
End Sub
```

'UserDrawLakeSurface 创建角色的背景图面

Rem 可以在 UserInitializeDX7 初始化后直接在 TIMER 中调用，用在一段时间更改背景图案
Rem BackPictureObject 把图片显示在这个控件上，这里主要起到限制大小
Rem FileNameBack 显示的背景图片位置，可以每次不同做出动画效果

```
Sub UserDrawLakeSurface(BackPictureObject As Object, FileNameBack As String)
    '创建角色的背景图面=======================================
    ddsdLake.lFlags = DDSD_CAPS Or DDSD_WIDTH Or DDSD_HEIGHT
    ddsdLake.ddsCaps.lCaps = DDSCAPS_OFFSCREENPLAIN
    ddsdLake.lWidth = BackPictureObject.ScaleWidth
    ddsdLake.lHeight = BackPictureObject.ScaleHeight
    Set LakeSurface = DirectDraw.CreateSurfaceFromFile(FileNameBack, ddsdLake)
    '====================================================
    RepaintEntireBackground '把 Lake 背景画在缓存中
End Sub
```

Rem InitSurfaceSprite 初始化角色面，然后创建角色面 SpriteSurfaceRestorePic 子程序
Rem 初始化角色面，然后创建角色面
Rem FileNameSprite 角色图片位置
Rem SpriteShowSizeX,SpriteShowSizeY 角色图片的显示大小

```
Sub InitSurfaceSprite(FileNameSprite() As String, SpriteShowSizeX() As Long, SpriteShowSizeY() As Long)
    Dim i As Long
    '创建角色画面==========================================
    ddsdSprite.lFlags = DDSD_CAPS Or DDSD_WIDTH Or DDSD_HEIGHT
    For i = LBound(FileNameSprite) To UBound(FileNameSprite)
    Rem  这里是透明图片显示大小设置
    ddsdSprite.lWidth = SpriteShowSizeX(i) '宽 48
    ddsdSprite.lHeight = SpriteShowSizeY(i) '长 64 像素
    ddsdSprite.ddsCaps.lCaps = DDSCAPS_OFFSCREENPLAIN
    Set SpriteSurface = DirectDraw.CreateSurfaceFromFile(FileNameSprite(i), ddsdSprite) '创建一个角色面从文件地址
    Next i
    '=====================================================
    '设置透明色区间，0 是黑色=================================
    '----- setting the transparent color of the CharPicPath
    Dim Key As DDCOLORKEY
    Key.Low = 0
    Key.high = 0
    SpriteSurface.SetColorKey DDCKEY_SRCBLT, Key
    '=====================================================
    rSprite.Bottom = ddsdSprite.lHeight
    rSprite.Right = ddsdSprite.lWidth
    RepaintEntireBackground '把 Lake 背景画在缓存中
End Sub
```

Rem RepaintEntireBackground 把 Lake 背景画在缓存中 InitSurfaceSprite 子程序

Rem 把 Lake 背景画在缓存中

```
Sub RepaintEntireBackground()
    Dim n As Long
    n = BackBuffer.BltFast(0, 0, LakeSurface, rLake, DDBLTFAST_WAIT)
End Sub

Sub SpriteAngle()
Rem  角色显示角度，这个没加入
    Static a As Single
    Static T1 As Single
    Static T2 As Single
    T2 = Timer
    If T1 <> 0 Then
    a = a + (T2 - T1) * 100
    If a > 360 Then a = a - 360
    End If
    T1 = T2
End Sub
```

Rem 找回可能丢失的图片

```
Sub SpriteSurfaceRestorePic(BackPictureObject As Object, FileNameBack As String,
FileNameSprite() As String, SpriteShowSizeX() As Long, SpriteShowSizeY() As Long)
    Dim i As Long
    Dim bRestore As Boolean
    ' this will keep us from trying to blt in case we lose the surfaces (another fullscreen app takes
over)
    bRestore = False
    Do Until ExModeActive
        DoEvents
         bRestore = True
    Loop
    'if we lost and got back the surfaces, then restore them
    DoEvents
    If bRestore Then
        bRestore = False
        DirectDraw.RestoreAllSurfaces
    For i = LBound(FileNameSprite) To UBound(FileNameSprite)
    'InitSurfaces BackPictureObject, FileNameBack, FileNameSprite ' must UserInitializeDX7 the
surfaces again if they we're lost
    InitSurfaceLake BackPictureObject, FileNameBack, rLake
    InitSurfaceSprite FileNameSprite, SpriteShowSizeX, SpriteShowSizeY
    Next i
    End If
End Sub
```

Rem 图片显示的帧数

```
Sub PicFPS(ShowAt As Object)
  Static i As Long
      Static tLast As Single
    Static tNow As Single
    'calculate FPS
    i = i + 1
    If i = 30 Then
        tNow = Timer
        If tNow <> tLast Then
            Fps = 30 / (Timer - tLast)
            tLast = Timer
            i = 0
           ShowAt.Caption = "DD Transparency       Frames per Second =" + Format$(Fps, "#.0")
        End If
    End If
End Sub
```

Rem 角色图片等分后显示其中的一部分

Rem rSprite 要显示的矩形框

Rem CharShowPartX，PartY 要显示的横纵位置

Rem CharTotalPartX，TotalPartY 要等分成多少份

```
Private Sub SpriteShowWhichEqualPart(rSprite As RECT, CharShowPartX As Long, CharShowPartY As Long, CharTotalPartX As Long, CharTotalPartY As Long)
    rSprite.Left = ddsdSprite.lWidth / CharTotalPartX * (CharShowPartX - 1)
    rSprite.Right = ddsdSprite.lWidth / CharTotalPartX * CharShowPartX
    rSprite.Top = ddsdSprite.lHeight / CharTotalPartY * (CharShowPartY - 1)
    rSprite.Bottom = ddsdSprite.lHeight / CharTotalPartY * CharShowPartY
End Sub
```

Rem 清除上次绘制的图片，准备绘制新的图片

```
Private Sub SpriteSurfaceCleanLastTimePicture(Count As Long, StartX() As Single, StartY() As Single, CharTotalPartX() As Long, CharTotalPartY() As Long)
    'clean up background from last frame
    'by only reparing the background where it needs to
    'be you wont need to reblit the whole thing
    Dim rClean As RECT
        rClean.Left = StartX(Count)
        rClean.Top = StartY(Count)
        rClean.Right = StartX(Count) + ddsdSprite.lWidth / CharTotalPartX(Count)
        rClean.Bottom = StartY(Count) + ddsdSprite.lHeight / CharTotalPartY(Count)
       Call BackBuffer.BltFast(StartX(Count), StartY(Count), LakeSurface, rClean, DDBLTFAST_WAIT)

End Sub
```

Rem 清除上次绘制的图片，准备绘制新的图片

Rem 用在多个可重叠的有透明部分的图片

```
Public Sub SpriteSurfaceCleanLastTimePicturePublic(Low As Long, Up As Long, StartX() As Single, StartY() As Single, CharTotalPartX() As Long, CharTotalPartY() As Long)
    'clean up background from last frame
    'by only reparing the background where it needs to
    'be you wont need to reblit the whole thing
Dim i As Long
    For i = Low To Up
    Dim rClean As RECT
        rClean.Left = StartX(i)
        rClean.Top = StartY(i)
        rClean.Right = StartX(i) + ddsdSprite.lWidth / CharTotalPartX(i)
        rClean.Bottom = StartY(i) + ddsdSprite.lHeight / CharTotalPartY(i)
       Call BackBuffer.BltFast(StartX(i), StartY(i), LakeSurface, rClean, DDBLTFAST_WAIT)
   Next i
End Sub
```

'UserDrawSpriteSurface 绘制角色的主要页面

Rem 绘制角色的主要页面
Rem BackPictureObject 要绘制图片到的控件
Rem FileNameBack 要显示的背景图片位置
Rem FileNameSprite 要显示的角色图片位置
Rem 角色初始位置 StartX，StartY
Rem Mode 角色移动方式
Rem Speed 角色移动速度
Rem CharShowPartX CharShowPartY 显示的角色图片上的部分
Rem CharTotalPartX CharTotalPartY 角色图片分成多少份 TotalPartX 横向份数
Rem SpriteShowSizeX 角色图片显示大小

```
Sub UserDrawSpriteSurface(BackPictureObject As Object, FileNameBack As String,
FileNameSprite() As String, StartX() As Single, StartY() As Single, _
CharSpeedX() As Double, CharSpeedY() As Double, CharShowPartX() As Long, CharShowPartY() As
Long, CharTotalPartX() As Long, CharTotalPartY() As Long, SpriteShowSizeX() As Long,
SpriteShowSizeY() As Long)
Dim X As Single, Y As Single, i As Long
    'InitSurfaceSprite FileNameSprite, SpriteShowSizeX, SpriteShowSizeY
    '创建角色画面==========================================
    Rem 这里是透明图片显示大小设置
    For i = LBound(FileNameSprite) To UBound(FileNameSprite)
    SpriteSurfaceCreateSprite i, FileNameSprite, SpriteShowSizeX, SpriteShowSizeY '创建一个角
色
    SpriteSurfaceSetTransparentColor '设置透明色区间，0 是黑色
    'RepaintEntireBackground '把 Lake 背景画在缓存中这个没有启用,启用后只有一个角色显
示减少一半的刷新速度
    SpriteSurfaceRestorePic BackPictureObject, FileNameBack, FileNameSprite, SpriteShowSizeX,
SpriteShowSizeY '找回丢失图片
    SpriteSurfaceCleanLastTimePicture i, StartX, StartY, CharTotalPartX, CharTotalPartY '清除上
次绘制的图片，准备绘制新的图片
    SpriteSurfaceMove i, StartX, StartY, CharSpeedX, CharSpeedY, X, Y '角色移动
    SpriteSurfaceShowWhichPart i, CharTotalPartX, CharTotalPartY, CharShowPartX,
CharShowPartY '显示角色图片哪一部分
    SpriteSurfaceDrawBackBuffer rSprite, X, Y '把角色画到缓存中
    StartX(i) = X
    StartY(i) = Y
    Next i
End Sub
```

Rem SpriteSurfaceDrawBackBuffer 绘制角色的主要页面 UserDrawSpriteSurface 子程序
Rem 把角色画到缓存中

```
Private Sub SpriteSurfaceDrawBackBuffer(rSprite As RECT, X As Single, Y As Single)
    Call BackBuffer.BltFast(X, Y, SpriteSurface, rSprite, DDBLTFAST_SRCCOLORKEY) 'bltfast 不能够
```

```
用在伸缩镜像，和附加裁剪器
End Sub
```

'UserDrawPrimerySurface 把缓冲面画到主页面

Rem 把缓冲面画到主页面

```
Sub UserDrawPrimerySurface()
Dim rPrim As RECT
Dim n As Long
    'Get the position of our picture box in screen coordinates
    DirectX.GetWindowRect Frm_Main.PictureBack.hWnd, rPrim
    'blt our back buffer to the screen
    n = PrimSurface.Blt(rPrim, BackBuffer, rBackBuffer, DDBLT_WAIT)
End Sub
```

'UserQuitDX7 卸载 DX7

Rem 卸载加载的内容

```
Sub UserQuitDX7()
On Error Resume Next
    'Set Device = Nothing
    'Set Direct3D = Nothing
    Set Clipper = Nothing
    Set BackBuffer = Nothing
    Set PrimSurface = Nothing
    Set DirectDraw = Nothing
    Set DirectX = Nothing
    'RunDX = False
    running = False
End Sub
```

Rem UserDrawSpriteSurface 子程序

Rem 设置角色透明色

```
Private Sub SpriteSurfaceSetTransparentColor()
    Dim Key As DDCOLORKEY
    Key.Low = 0
    Key.high = 0
    SpriteSurface.SetColorKey DDCKEY_SRCBLT, Key
End Sub
```

Rem UserDrawSpriteSurface 子程序

Rem 创建一个角色

```
Private Sub SpriteSurfaceCreateSprite(Count As Long, FileNameSprite() As String,
SpriteShowSizeX() As Long, SpriteShowSizeY() As Long)
```

```
    ddsdSprite.lFlags = DDSD_CAPS Or DDSD_WIDTH Or DDSD_HEIGHT
    ddsdSprite.lWidth = SpriteShowSizeX(Count) '例子总宽 192 像素
    ddsdSprite.lHeight = SpriteShowSizeY(Count) '例子总长 256 像素
    ddsdSprite.ddsCaps.lCaps = DDSCAPS_OFFSCREENPLAIN
    Set SpriteSurface = DirectDraw.CreateSurfaceFromFile(FileNameSprite(Count), ddsdSprite) '
创建一个角色面从文件地址
End Sub
```

Rem UserDrawSpriteSurface 子程序

Rem 角色移动

```
Private Sub SpriteSurfaceMove(Count As Long, StartX() As Single, StartY() As Single, CharSpeedX()
As Double, CharSpeedY() As Double, X As Single, Y As Single)
    X = StartX(Count)
    Y = StartY(Count)
    X = X + CharSpeedX(Count)
    Y = Y + CharSpeedY(Count)
End Sub
```

Rem 显示角色图片的哪一部分 UserDrawSpriteSurface 子程序

Rem 显示角色图片的哪一部分

```
Private Sub SpriteSurfaceShowWhichPart(Count As Long, CharTotalPartX() As Long,
CharTotalPartY() As Long, CharShowPartX() As Long, CharShowPartY() As Long)
    rSprite.Left = ddsdSprite.lWidth / CharTotalPartX(Count) * (CharShowPartX(Count) - 1)
    rSprite.Right = ddsdSprite.lWidth / CharTotalPartX(Count) * CharShowPartX(Count)
    rSprite.Top = ddsdSprite.lHeight / CharTotalPartY(Count) * (CharShowPartY(Count) - 1)
    rSprite.Bottom = ddsdSprite.lHeight / CharTotalPartY(Count) * CharShowPartY(Count)
End Sub
```

'UserDrawSpriteSurfaceFast 绘制角色的主要页面

Rem 绘制角色的主要页面

Rem BackPictureObject 要绘制图片到的控件

Rem FileNameBack 要显示的背景图片位置

Rem FileNameSprite 要显示的角色图片位置

Rem 角色初始位置 StartX，StartY

Rem Mode 角色移动方式

Rem Speed 角色移动速度

Rem CharShowPartX CharShowPartY 显示的角色图片上的部分

Rem CharTotalPartX CharTotalPartY 角色图片分成多少份 TotalPartX 横向份数

Rem SpriteShowSizeX 角色图片显示大小

```
Sub UserDrawSpriteSurfaceFast(BackPictureObject As Object, FileNameBack As String,
WhichSprite As Long, FileNameSprite() As String, StartX() As Single, StartY() As Single, _
CharSpeedX() As Double, CharSpeedY() As Double, CharShowPartX() As Long, CharShowPartY() As
Long, CharTotalPartX() As Long, CharTotalPartY() As Long, SpriteShowSizeX() As Long,
SpriteShowSizeY() As Long)
```

```
Dim X As Single, Y As Single, i As Long
    'InitSurfaceSprite FileNameSprite, SpriteShowSizeX, SpriteShowSizeY
    '创建角色画面==========================================
    Rem  这里是透明图片显示大小设置
    i = WhichSprite
    SpriteSurfaceCreateSprite i, FileNameSprite, SpriteShowSizeX, SpriteShowSizeY '创建一个角色面
    SpriteSurfaceSetTransparentColor '设置透明色区间，0 是黑色
    'RepaintEntireBackground '把 Lake 背景画在缓存中这个没有启用,启用后只有一个角色显示减少一半的刷新速度
    SpriteSurfaceRestorePic BackPictureObject, FileNameBack, FileNameSprite, SpriteShowSizeX, SpriteShowSizeY '找回丢失图片
    'SpriteSurfaceCleanLastTimePicture i, StartX, StartY, CharTotalPartX, CharTotalPartY '清除上次绘制的图片，准备绘制新的图片,但会导致重叠图片非透明部分显示不正常
    SpriteSurfaceMove i, StartX, StartY, CharSpeedX, CharSpeedY, X, Y '角色移动
    SpriteSurfaceShowWhichPart i, CharTotalPartX, CharTotalPartY, CharShowPartX, CharShowPartY '显示角色图片哪一部分
    Call BackBuffer.BltFast(X, Y, SpriteSurface, rSprite, DDBLTFAST_SRCCOLORKEY) '把角色画到缓存中
    'Call BackBuffer.Blt(rTarget, SpriteSurface, rSprite, DDBLT_ROTATIONANGLE)’位块转移操作的旋转角度
    Rem  这时吧 BackBuffer 改成背景图
    StartX(i) = X
    StartY(i) = Y
End Sub
```

39.Lines(DX7 飞船)

详细说明：

函数或方法名称	作用
Lines_IsoscelesTriangle_A	返回等腰三角形的绘制“△”
Lines_IsoscelesTriangle_B	返回等腰三角形的绘制“△”
Lines_IsoscelesTriangle_AngleOrthocenter_A	返回垂心
Lines_IsoscelesTriangle_Incircle_A	返回等腰三角形内切圆圆心“△”
Lines_IsoscelesTriangle_Incircle_Vertex	从内切圆返回等腰三角形顶点位置
AddItem Combo1	增加内容

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'**************************************************************************
'名称 = Lines.Cls
'作者 = 张宏
'最后修改时间 = 2017 年 6 月 20 日
'简介 =几何操作
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=None
'操作历史：
'**************************************************************************
```

'Lines_IsoscelesTriangle_A 返回等腰三角形的绘制“△”

```
Rem 返回等腰三角形的绘制“△”
Rem VertexX , VertexY 顶点坐标
Rem WaistLength 腰长，BaseLength 底长
Public Function Lines_IsoscelesTriangle_A(ByVal WaistLength As Double, ByVal BaseLength As
Double, _
ByVal VertexX As Double, _
ByVal VertexY As Double, _
ByRef Return_Line1X1 As Double, ByRef Return_Line1Y1 As Double, _
ByRef Return_Line1X2 As Double, ByRef Return_Line1Y2 As Double, _
ByRef Return_Line2X1 As Double, ByRef Return_Line2Y1 As Double, _
ByRef Return_Line2X2 As Double, ByRef Return_Line2Y2 As Double, _
ByRef Return_Line3X1 As Double, ByRef Return_Line3Y1 As Double, _
ByRef Return_Line3X2 As Double, ByRef Return_Line3Y2 As Double) As Boolean
```

```
Dim i As Long, l As Long, m As Long, nn As Long, n As Long
Dim ii As Long, ll As Long
Return_Line1X1 = VertexX
Return_Line1Y1 = VertexY
Return_Line1X2 = VertexX - BaseLength / 2
Return_Line1Y2 = VertexY + Sqr(WaistLength ^ 2 - (BaseLength / 2) ^ 2)

Return_Line2X1 = VertexX - BaseLength / 2
Return_Line2Y1 = VertexY + Sqr(WaistLength ^ 2 - (BaseLength / 2) ^ 2)
Return_Line2X2 = VertexX + BaseLength / 2
Return_Line2Y2 = VertexY + Sqr(WaistLength ^ 2 - (BaseLength / 2) ^ 2)

Return_Line3X1 = VertexX + BaseLength / 2
Return_Line3Y1 = VertexY + Sqr(WaistLength ^ 2 - (BaseLength / 2) ^ 2)
Return_Line3X2 = VertexX
Return_Line3Y2 = VertexY
End Function
```

'Lines_IsoscelesTriangle_B 返回等腰三角形的绘制“△”

Rem 返回等腰三角形的绘制“△”
Rem TOP , LEFT 座标
Rem WaistLength 腰长，BaseLength 底长

```
Public Function Lines_IsoscelesTriangle_B(ByVal WaistLength As Double, ByVal BaseLength As
Double, _
ByVal Left As Double, _
ByVal Top As Double, _
ByRef Return_Line1X1 As Double, ByRef Return_Line1Y1 As Double, _
ByRef Return_Line1X2 As Double, ByRef Return_Line1Y2 As Double, _
ByRef Return_Line2X1 As Double, ByRef Return_Line2Y1 As Double, _
ByRef Return_Line2X2 As Double, ByRef Return_Line2Y2 As Double, _
ByRef Return_Line3X1 As Double, ByRef Return_Line3Y1 As Double, _
ByRef Return_Line3X2 As Double, ByRef Return_Line3Y2 As Double) As Boolean
Dim i As Long, l As Long, m As Long, nn As Long, n As Long
Dim ii As Long, ll As Long
Rem 以定点为坐标
Return_Line1X1 = Left + BaseLength / 2
Return_Line1Y1 = Top
Return_Line1X2 = Left
Return_Line1Y2 = Top + Sqr(WaistLength ^ 2 - (BaseLength / 2) ^ 2)

Return_Line2X1 = Left
Return_Line2Y1 = Top + Sqr(WaistLength ^ 2 - (BaseLength / 2) ^ 2)
```

```
Return_Line2X2 = Left + BaseLength
Return_Line2Y2 = Top + Sqr(WaistLength ^ 2 - (BaseLength / 2) ^ 2)

Return_Line3X1 = Left + BaseLength
Return_Line3Y1 = Top + Sqr(WaistLength ^ 2 - (BaseLength / 2) ^ 2)
Return_Line3X2 = Left + BaseLength / 2
Return_Line3Y2 = Top
End Function
```

'Lines_IsoscelesTriangle_AngleOrthocenter_A 返回垂心

Rem 返回垂心
Rem (6 次 2 元方程未解出)发现 2 元方程就可以
Rem 等腰角形的绘制 “ △ ”
Rem VertexX , VertexY 顶点坐标
Rem WaistLength 腰长，BaseLength 底长
Rem Angle 角度 Return_AngleOrthocenter

```
Public Function Lines_IsoscelesTriangle_AngleOrthocenter_A(ByVal Angle As Double, ByVal
WaistLength As Double, ByVal BaseLength As Double, _
ByVal VertexX As Double, _
ByVal VertexY As Double, _
ByRef Return_AngleOrthocenterX As Double, _
ByRef Return_AngleOrthocenterY As Double) As Boolean
Dim i As Long, l As Long, m As Long, nn As Long, n As Long
Dim ii As Long, ll As Long
Dim BL As Double, WL As Double, H1 As Double
BL = BL
WL = BL
H1 = Sqr((BL ^ 2 + 4 * WL ^ 4 - 4 * WL ^ 2 * BL ^ 2) / (4 * WL ^ 2 - BL ^ 2))
If Sqr(2) / 2 * BL = WL Then
Return_AngleOrthocenterX = VertexX
Return_AngleOrthocenterY = VertexY
End If
If Sqr(2) / 2 * BL > WL Then
Return_AngleOrthocenterX = VertexX
Return_AngleOrthocenterY = VertexY - H1
MsgBox "垂心在外面"
End If
If Sqr(2) / 2 * BL < WL Then
Return_AngleOrthocenterX = VertexX
Return_AngleOrthocenterY = VertexY + H1
End If
```

```
End Function
```

'Lines_IsoscelesTriangle_Incircle_A 返回等腰三角形内切圆圆心“△”

Rem 返回等腰三角形内切圆圆心“△”
Rem VertexX , VertexY 顶点坐标
Rem WaistLength 腰长，**BaseLength** 底长
Rem Return_IncircleX，Return_IncircleY 内切圆圆心坐标
Rem Return_Radius 内切圆半径
Rem (ByVal Angle As Double)

```
Public Function Lines_IsoscelesTriangle_Incircle_A(ByVal WaistLength As Double, ByVal
BaseLength As Double, _
ByVal VertexX As Double, _
ByVal VertexY As Double, _
ByRef Return_Radius As Double, _
ByRef Return_IncircleX As Double, _
ByRef Return_IncircleY As Double) As Boolean
Dim i As Long, l As Long, m As Long, nn As Long, n As Long
Dim ii As Long, ll As Long
Dim BL As Double, WL As Double, N1 As Double
BL = BaseLength
WL = WaistLength
N1 = 2 * WL * Sqr(WL ^ 2 - BL ^ 2 / 4) / (BL + 2 * WL)
Return_IncircleX = VertexX
Return_IncircleY = VertexY + N1
Return_Radius = Sqr(WL ^ 2 - BL ^ 2 / 4) - N1
End Function
```

'Lines_IsoscelesTriangle_Incircle_Vertex 从内切圆返回等腰三角形顶点位置

Rem 从内切圆返回等腰三角形顶点位置
Rem VertexX , VertexY 顶点坐标
Rem WaistLength 腰长，**BaseLength** 底长
Rem Return_IncircleX，Return_IncircleY 内切圆圆心坐标
Rem Return_Radius 内切圆半径

```
Public Function Lines_IsoscelesTriangle_Incircle_Vertex(ByVal WaistLength As Double, ByVal
BaseLength As Double, _
ByVal IncircleX As Double, _
```

```
ByVal IncircleY As Double, _
ByRef Return_VertexX As Double, _
ByRef Return_VertexY As Double) As Boolean
Dim i As Long, l As Long, m As Long, nn As Long, n As Long
Dim ii As Long, ll As Long
Dim BL As Double, WL As Double, N1 As Double
BL = BaseLength
WL = WaistLength
N1 = 2 * WL * Sqr(WL ^ 2 - BL ^ 2 / 4) / (BL + 2 * WL)
Return_VertexX = IncircleX
Return_VertexY = IncircleY - N1
End Function
```

'AddItem Combo1 增加内容

'定义的时候加上 Optional 关键字，就可以了
'要注意的是，可选参数后面如果还有其他的参数，则必须都是可选参数。
'另外，定义可选参数，需要定义默认值。如果调用时，没有指定这个可选参数的值，则使用默认值
'IsMissing 对简单数据类型（例如 Integer 或 Double）不起作用，因为与 Variants 不同，它们没有“丢失”标志位的前提。正由于此，对于可选参数类型，可以指定缺省值。如果调用过程时，参数被忽略，则该参数将具有该缺省值，如下列示例中所示：

```
Public Sub AddItem(ByVal Item As String, Optional ByVal Index As Integer = 0)
Combo1.AddItem Item, Index
End Sub
```

40.MoveDirection(DX7 飞船)

详细说明：

函数或方法名称	作用
MoveDirection	角色移动分解到水平和垂直上的值

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'**************************************************************************
'名称 = MoveDirection.Cls
'作者 = 张宏
'最后修改时间 = 2022 年 1 月 2 日
'简介 = 移动方向
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=None
'操作历史：
'**************************************************************************
```

'MoveDirection 角色移动分解到水平和垂直上的值

```
Rem 角色移动分解到水平和垂直上的值
Rem CharFullSpeed=移动基础速度
Rem CharAngle=移动角度既是方向
Rem ReturnSpeedX=返回水平移动速度（把斜线运动分解为 2 个方向）
Rem ReturnSpeedY=返回垂直移动速度（把斜线运动分解为 2 个方向）
Public Function MoveDirection(ByVal CharFullSpeed As Double, ByVal CharAngle As Double, _
ByRef ReturnSpeedX As Double, ByRef ReturnSpeedY As Double)
Const PI As Double = 3.1415926
If CharAngle > 0 And CharAngle < 90 Then
ReturnSpeedX = Cos(CharAngle * PI / 180) * CharFullSpeed
ReturnSpeedY = -Sin(CharAngle * PI / 180) * CharFullSpeed
End If
If CharAngle > 90 And CharAngle < 180 Then
ReturnSpeedX = -Sin((CharAngle - 90) * PI / 180) * CharFullSpeed
ReturnSpeedY = -Cos((CharAngle - 90) * PI / 180) * CharFullSpeed
End If
If CharAngle > 180 And CharAngle < 270 Then
ReturnSpeedX = -Cos((CharAngle - 180) * PI / 180) * CharFullSpeed
```

```
ReturnSpeedY = Sin((CharAngle - 180) * PI / 180) * CharFullSpeed
End If
If CharAngle > 270 And CharAngle < 360 Then
ReturnSpeedX = Sin((CharAngle - 270) * PI / 180) * CharFullSpeed
ReturnSpeedY = Cos((CharAngle - 270) * PI / 180) * CharFullSpeed
End If
If CharAngle = 0 Or CharAngle = 360 Then
ReturnSpeedX = CharFullSpeed
ReturnSpeedY = 0
End If
If CharAngle = 90 Then
ReturnSpeedX = 0
ReturnSpeedY = -1 * CharFullSpeed
End If
If CharAngle = 180 Then
ReturnSpeedX = -1 * CharFullSpeed
ReturnSpeedY = 0
End If
If CharAngle = 270 Then
ReturnSpeedX = 0
ReturnSpeedY = CharFullSpeed
End If
End Function
```

41. StatisticsCorrelationAnalysis(EVE 脚本)

详细说明：

函数或方法名称	作用
AverageValue	平均值
TotalValue	总值
R	相关系数
ρ	斯皮尔曼相关系数
Array1D_Sort_Which	ρ子程序
FunctionError	ρ子程序判断数据是否相同
FunctionErrorτ_a	判断数据是否有相同元素
τ_a	计算
Array1D_DifferentNum	多程序子程序
τ_b	计算（有错）
τ_c	计算（有错）
TestOfSignificance_r	回归系数显著性
TestOfSignificance_ρ	回归系数显著性

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*********************************************************************************
'名称 = StatisticsCorrelationAnalysis.Cls
'作者 = 张宏
'最后修改时间 = 2018 年 12 月 17 日
'简介 = 数学统计模块
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'说明：肯德尔等级有错
'*********************************************************************************
'需要 Statistics_T002 类模块
Dim obj As New Statistics_T002
```

'AverageValue 平均值

```
Public Function AverageValue(Data() As Double) As Double
AverageValue = TotalValue(Data()) / (UBound(Data) - LBound(Data) + 1)
End Function
```

'TotalValue 总值

```
Public Function TotalValue(Data() As Double) As Double
Dim Num As Variant
For Each Num In Data
TotalValue = Num + TotalValue
Next Num
End Function
```

```
Public Function SquareTotalValue(Data() As Double) As Double
SquareTotalValue = TotalValue ^ 2
End Function
```

'R 相关系数

'(1)、两个变量之间是线性关系，都是连续数据。
'(2)、两个变量的总体是正态分布，或接近正态的单峰分布。
'(3)、两个变量的观测值是成对的，每对观测值之间相互独立。
Rem 简单相关分析，初步测试通过
Rem Pearson 相关性分析系数 r

```
Public Function R(DataX() As Double, DataY() As Double) As Double
Dim SP As Double, SSx As Double, SSy As Double
Dim i As Long, l As Long
Dim AverageX As Double, AverageY    As Double
Dim MakeError As Boolean
On Error GoTo a1
'================================================
MakeError = FunctionError(DataX, DataY)
If MakeError = True Then GoTo a2
'================================================
Rem 平均值
AverageX = AverageValue(DataX)
AverageY = AverageValue(DataY)
'================================================
For i = LBound(DataX) To UBound(DataX)
SP = SP + (DataX(i) - AverageX) * (DataY(i) - AverageY)
SSx = SSx + (DataX(i) - AverageX) ^ 2
SSy = SSy + (DataY(i) - AverageY) ^ 2
Next i
'================================================
If SSx <> 0 And SSy <> 0 Then
R = SP / Sqr(SSx * SSy)
```

```
Else
MsgBox "Statistics_CorrelationAnalysis001 r 出错，除数为 0"
MsgBox "改用 Spearman 斯皮尔曼相关性系数"
R = ρ(DataX, DataY)
End If
Exit Function
'=================================================
Rem 错误处理
a1:
MsgBox "Statistics_CorrelationAnalysis001 r 出错，其他错误请检查数据 返回值=0"
a2:
R = 0
End Function
```

'ρ 斯皮尔曼相关系数

'Spearman 斯皮尔曼相关性系数没有那些数据条件要求，适用的范围就广多了。
'在我们生物实验数据分析中，尤其是在分析多组学交叉的数据中说明不同组学数据之间的相关性时，使用的频率很高。
'斯皮尔曼等级相关系数对数据条件的要求没有皮尔逊相关系数严格，'
'只要两个变量的观测值是成对的等级评定资料，或者是由连续变量观测资料转化得到的等级资料，'
'不论两个变量的总体分布形态、样本容量的大小如何，都可以用斯皮尔曼等级相关系数来进行研究。

```
Public Function ρ(DataX() As Double, DataY() As Double) As Double
Dim a As Double, b As Double, c As Double
Dim i As Long, l As Long
Dim Number As Long
Dim R1() As Double, R2() As Long, R3() As Double, R4() As Long, R5() As Long
Dim di As Double
'~~~~~~~~~~~~~~~~
Dim MakeError As Boolean
On Error GoTo a1
MakeError = FunctionError(DataX, DataY)
If MakeError = True Then GoTo a2
Rem 获得数据从大到小排序
Array1D_Sort_Which DataX, R1, R2, R3, R4
Array1D_Sort_Which DataY, R1, R2, R3, R5
'=============================================
Rem 获得 2 组数据从大到小排序后的秩次差的平方求和
For i = LBound(R4) To UBound(R4)
di = di + (R4(i) - R5(i)) ^ 2
Next i
```

```
'=============================================
Rem 获得数据总数
Number = (UBound(DataX) - LBound(DataX) + 1)
ρ = 1 - 6 * di / (Number * (Number ^ 2 - 1))
Exit Function
'=============================================
Rem 错误处理
a1:
MsgBox "Statistics_CorrelationAnalysis001 ρ出错，其他错误请检查数据 返回值=0"
a2:
ρ = 0
End Function
```

Rem ρ子程序

Rem Spearman 斯皮尔曼相关性系数需要用到排序

```
Public Function Array1D_Sort_Which(ByRef Data() As Double, ByRef Return_SortLtoU() As Double,
ByRef Return_SortLtoU_Which() As Long, ByRef Return_SortUtoL() As Double, ByRef
Return_SortUtoL_Which() As Long) As Boolean
Dim i As Long, l As Long, t As Double
Dim OData() As Double
OData = Data
For l = LBound(Data) To UBound(Data)
For i = LBound(Data) To (UBound(Data) - l) '冒泡排序这里不能用 i=L!!!只能是(UBound(Data) -
l)
If i = UBound(Data) Then
i = LBound(Data)
l = l + 1
If l = UBound(Data) Then
Exit For
End If
End If
If Data(i) > Data(i + 1) Then
t = Data(i)
Data(i) = Data(i + 1)
Data(i + 1) = t
End If
'Debug.Print "Data(" & i & ")= " & Data(i) & "    *****      " & l
Next
Next
ReDim Return_SortLtoU(LBound(Data) To UBound(Data))
ReDim Return_SortUtoL(LBound(Data) To UBound(Data))
For i = LBound(Data) To UBound(Data)
'Debug.Print "Data(" & i & ")= " & Data(i) & "    PaiXu up"
Return_SortLtoU(i) = Data(i)
Next
```

```
ReDim Return_SortLtoU_Which(LBound(Data) To UBound(Data))

For i = LBound(OData) To UBound(OData)
For l = LBound(OData) To UBound(OData)
If OData(i) = Return_SortLtoU(l) Then
Return_SortLtoU_Which(i) = l
'Debug.Print "Return_SortLtoU_Which(" & i & ")= " & l
End If
Next l
Next i

l = LBound(Data) - 1
For i = UBound(Data) To LBound(Data) Step -1
'Debug.Print "Data(" & i & ")= " & Data(i) & "    PaiXu down"
l = l + 1
Return_SortUtoL(l) = Data(i)
Next

ReDim Return_SortUtoL_Which(LBound(Data) To UBound(Data))
For i = LBound(OData) To UBound(OData)
For l = LBound(OData) To UBound(OData)
If OData(i) = Return_SortUtoL(l) Then
Return_SortUtoL_Which(i) = l
'Debug.Print "Return_SortUtoL_Which(" & i & ")= " & l
End If
Next l
Next i

End Function
```

Rem ρ子程序 FunctionError 判断数据是否相同

Rem 判断数据是否相同

```
Private Function FunctionError(DataX() As Double, DataY() As Double) As Boolean
FunctionError = False
If (UBound(DataX) - LBound(DataX) + 1) <> (UBound(DataY) - LBound(DataY) + 1) Then
MsgBox "Statistics_CorrelationAnalysis001 出错，2 组数据行数不同。退出。值返回为 0"
FunctionError = True
End If
End Function
```

'FunctionErrorτ_a 判断数据是否有相同元素

Rem 判断数据是否有相同元素

```
Private Function FunctionErrorτ_a(DataX() As Double, DataY() As Double) As Boolean
```

```
Dim Num As Integer, R1() As Double, R2() As Long, R3 As Long
FunctionErrorτ_a = False
Num = Array1D_DifferentNum(DataX, R1(), R2(), R3)
If Num <> (UBound(DataX) - LBound(DataX) + 1) Then
FunctionErrorτ_a = True
MsgBox "Statistics_CorrelationAnalysis001 τ_a 出错，DataX 单组数据中存在相同元素。退出。值返回为 0"
Exit Function
End If
Num = Array1D_DifferentNum(DataY, R1(), R2(), R3)
If Num <> (UBound(DataY) - LBound(DataY) + 1) Then
FunctionErrorτ_a = True
MsgBox "Statistics_CorrelationAnalysis001 τ_a 出错，DataY 单组数据中存在相同元素。退出。值返回为 0"
Exit Function
End If
End Function
```

```
Private Function FunctionErrorτ_c(DataX() As Double, DataY() As Double) As Boolean
Rem 判断数据是否有相同元素
End Function
```

Rem 2018-12-15 不正确的公式 C,D 值不是相等与否
'https://blog.csdn.net/zhaozhn5/article/details/78392220
'Kendall Rank（肯德尔等级）相关系数
'一个肯德尔检验是一个无参数假设检验，它使用计算而得的相关系数去检验两个随机变量的统计依赖性。
'注意：无参数!!
'注意：τ_a 这一公式仅适用于集合 X 与 Y 中均不存在相同元素的情况（集合中各个元素唯一!）。
'肯德尔检验和斯皮尔曼等级相关系数对数据条件的要求没有皮尔逊相关系数严格，'
'只要两个变量的观测值是成对的等级评定资料，或者是由连续变量观测资料转化得到的等级资料，'
'不论两个变量的总体分布形态、样本容量的大小如何，都可以用肯德尔和斯皮尔曼等级相关系数来进行研究。
'~~~~~~
Rem 假设两个随机变量分别为 X、Y（也可以看做两个集合），它们的元素个数均为 N，'
'两个随即变量取的第 i（1<=i<=N)个值分别用 Xi、Yi 表示?
'X 与 Y 中的对应元素组成一个元素对集合 XY，其包含的元素为(Xi, Yi)（1<=i<=N）。
'当集合 XY 中任意两个元素(Xi, Yi)与(Xj, Yj)的排行相同时'
'（也就是说当出现情况 1 或 2 时；情况 1：Xi>Xj 且 Yi>Yj，情况 2：Xi<Xj 且 Yi<Yj），
'这两个元素就被认为是一致的。当出现情况 3 或 4 时（情况 3：Xi>Xj 且 Yi<Yj，情况 4：Xi<Xj 且 Yi>Yj）
'，这两个元素被认为是不一致的。当出现情况 5 或 6 时（情况 5：Xi=Xj，情况 6：Yi=Yj），
'这两个元素既不是一致的也不是不一致的。

'~~~~~~

'τ_a 计算

```
Public Function τ_a(DataX() As Double, DataY() As Double) As Double
Dim c As Long, D As Long
Dim MakeError As Boolean
Dim Number As Long
Dim i As Long
On Error GoTo a1
MakeError = FunctionError(DataX, DataY)
If MakeError = True Then GoTo a2
'MakeError = FunctionErrorτ_a(DataX, DataY)
If MakeError = True Then GoTo a2

Rem Matlab 实现程序
'for i = 1 : N - 1
'       for j = i + 1 : N
'             if abs(sum(XY(: , i) ~= XY(: , j))) == 2
'                   switch abs(sum(XY(: , i) > XY(: , j)))
 '                        Case 0
'                              C = C + 1;
 '                        Case 1
 '                             D = D + 1;
 '                        Case 2
'                              C = C + 1;
'                 End
'           End
'      End
'End

For i = LBound(DataX) To UBound(DataX)
If DataX(i) = DataY(i) Then
c = c + 1 '一致性的元素对数
Else
D = D + 1 '不一致性的元素对数
End If
Next i
Rem  获得数据总数
Number = (UBound(DataX) - LBound(DataX) + 1)
Rem  公式：
τ_a = (c - D) / (0.5 * Number * (Number - 1))
```

```
Exit Function
'=============================================
Rem 错误处理
a1:
MsgBox "Statistics_CorrelationAnalysis001 τ_a 出错，其他错误请检查数据 返回值=0"
a2:
τ_a = 0
End Function
```

Rem Array1D_DifferentNum 多程序子程序
Rem 得到数据中无重的数据
Rem 例如：110 和 110 是重复数据，120 和 130 不是重复数据，111 和 112 不是重复数据
Rem Data()原始数据
Rem Return_Different()数据中没有重复的数据
Rem Return_SameGroupNum()数据中相同数的个数 1，1，1，2，3，3 返回 3，1，2
Rem Array1D_DifferentNum 返回无重复元素的数据个数

```
Public Function Array1D_DifferentNum(ByRef Data() As Double, ByRef Return_Different() As
Double, _
ByRef Return_SameGroupNum() As Long, ByRef Return_HaveSameNumbersGroupNum As Long)
As Integer
    Dim m As Long, n As Long
    Dim Same As Boolean
    Dim SameGroupNum    As Long
    Dim Count() As Boolean
    Dim l As Long, i As Long
    Dim CountN() As Long
    Same = False

For l = LBound(Data()) To UBound(Data())
For i = l To UBound(Data())
    Rem 自动跳转
    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    If i = UBound(Data()) Then
        If Same = True Then
        Same = False
        End If
       i = l + 1
       l = l + 1
       If l = UBound(Data()) Then
       GoTo a2
       End If
  End If
    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


  If i + 1 <= UBound(Data()) Then
```

```
   ReDim Preserve Count(LBound(Data()) To UBound(Data()))
   'Debug.Print "data(" & L; ")= " & Data(L) & "      data(" & i + 1; ")=    " & Data(i + 1) & "
Count(" & L & ")= " & Count(L)
   ReDim Preserve CountN(LBound(Data()) To UBound(Data()))
   If Data(l) = Data(i + 1) And Count(l) = False Then
   Same = True
   Count(i + 1) = True
    '****************************************** 统计相同数的个数
    CountN(l) = CountN(l) + 1
    '******************************************
  End If
  End If
Next i
Next l
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem   显示不同的数
a2:
For i = LBound(Data()) To UBound(Data())
If Count(i) = False Then
Array1D_DifferentNum = Array1D_DifferentNum + 1
ReDim Preserve Return_Different(Array1D_DifferentNum)
Return_Different(Array1D_DifferentNum) = Data(i)
End If
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem    统计相同数的个数
Dim O As Long
ReDim Return_SameGroupNum(1 To Array1D_DifferentNum)
For i = LBound(Data()) To UBound(Data())
If Count(i) = False Then
O = O + 1
Return_SameGroupNum(O) = CountN(i) + 1
End If
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Rem    统计含有相同数的集合的个数
n = 0
For i = LBound(Return_SameGroupNum) To UBound(Return_SameGroupNum)
If Return_SameGroupNum(i) > 1 Then
n = n + 1
End If
Next i
Return_HaveSameNumbersGroupNum = n
```

```
End Function
```

'τ_b 计算

Rem 2018-12-15 不正确的公式 C,D 值不是相等与否
'Kendall Rank（肯德尔等级）相关系数
'注意：这一公式适用于集合 X 或 Y 中存在相同元素的情况
'（当然，如果 X 或 Y 中均不存在相同的元素时，公式二便等同于公式一）。
'肯德尔检验和斯皮尔曼等级相关系数对数据条件的要求没有皮尔逊相关系数严格，'
'只要两个变量的观测值是成对的等级评定资料，或者是由连续变量观测资料转化得到的等级资料，'
'不论两个变量的总体分布形态、样本容量的大小如何，都可以用肯德尔和斯皮尔曼等级相关系数来进行研究。

```
Public Function τ_b(DataX() As Double, DataY() As Double) As Double
Dim c As Long, D As Long
Dim MakeError As Boolean
Dim Number As Long
Dim i As Long
Dim Num As Integer, R1() As Double, R2() As Long, R3 As Long
Dim n1 As Double, n2 As Double, N3 As Double
On Error GoTo a1
MakeError = FunctionError(DataX, DataY)
If MakeError = True Then GoTo a2

For i = LBound(DataX) To UBound(DataX)
If DataX(i) = DataY(i) Then
c = c + 1 '一致性的元素对数
Else
D = D + 1 '不一致性的元素对数
End If
Next i
Rem 获得数据总数
Number = (UBound(DataX) - LBound(DataX) + 1)
Rem
Num = Array1D_DifferentNum(DataX, R1(), R2(), R3)
For i = 1 To R3
n1 = n1 + 0.5 * R2(i) * (R2(i) - 1)
Next i
Rem
Num = Array1D_DifferentNum(DataY, R1(), R2(), R3)
For i = 1 To R3
n2 = n2 + 0.5 * R2(i) * (R2(i) - 1)
Next i
```

```
Rem
N3 = 0.5 * Number * (Number - 1)
Rem
τ_b = (c - D) / Sqr((N3 - n1) * (N3 - n2))
Exit Function
'=============================================
Rem 错误处理
a1:
MsgBox "Statistics_CorrelationAnalysis001 τ_b 出错，其他错误请检查数据 返回值=0"
a2:
τ_b = 0
End Function
```

'τ_c 计算

Rem 2018-12-15 不正确的公式 C,D 值不是相等与否
Rem 这一公式中没有再考虑集合 X、或 Y 中存在相同元素给最后的统计值带来的影响。
'公式三的这一计算形式仅适用于用表格表示的随机变量 X、Y 之间相关系数的计算
'公式三则只是用来计算由长方形表格表示的二维变量的 Kendall 相关系数。
'M 表示长方形表格中行数与列数中较小的一个
'肯德尔检验和斯皮尔曼等级相关系数对数据条件的要求没有皮尔逊相关系数严格，'
'只要两个变量的观测值是成对的等级评定资料，或者是由连续变量观测资料转化得到的等级资料，'
'不论两个变量的总体分布形态、样本容量的大小如何，都可以用肯德尔和斯皮尔曼等级相关系数来进行研究。

```
Public Function τ_c(DataX() As Double, DataY() As Double) As Double
Dim c As Long, D As Long
Dim MakeError As Boolean
Dim Number As Long
Dim i As Long
Dim m As Double
On Error GoTo a1
MakeError = FunctionError(DataX, DataY)
If MakeError = True Then GoTo a2
MakeError = FunctionErrorτ_c(DataX, DataY)
If MakeError = True Then GoTo a2

For i = LBound(DataX) To UBound(DataX)
If DataX(i) = DataY(i) Then
c = c + 1 '一致性的元素对数
Else
D = D + 1 '不一致性的元素对数
End If
```

```
Next i
Rem 获得数据总数
Number = (UBound(DataX) - LBound(DataX) + 1)
Rem M 表示长方形表格中行数与列数中较小的一个

'M =
Rem 公式：
τ_c = (c - D) / (0.5 * Number ^ 2 * ((m - 1) / m))
Exit Function
'================================================
Rem 错误处理
a1:
MsgBox "Statistics_CorrelationAnalysis001 τ_c 出错，其他错误请检查数据 返回值=0"
a2:
τ_c = 0
End Function
```

'TestOfSignificance_r 回归系数显著性

Rem 需要 Statistics_T002 类模块
Rem 皮尔逊相关分析显著性检验
Rem Data1(),Data2()原始数据
Rem r 相关系数
Rem 返回 Return_df 自由度
Rem 返回 Return_T t 值用在显著性检验

```
Public Function TestOfSignificance_r(ByRef Data1() As Double, _
ByRef Data2() As Double, ByVal R As Double, ByRef Return_df As Long, _
ByRef Return_T As Double) As String

Dim n As Long

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim i As Long
For i = LBound(Data1) To UBound(Data1)
n = n + 1
Next i
Return_T = R / Sqr((1 - R ^ 2) / (n - 2))
Return_df = n - 2
TestOfSignificance_r = TestOfSignificance_r + "r=" + CStr(Round(R, 4)) + vbCrLf
TestOfSignificance_r = TestOfSignificance_r + "n=" + CStr(n) + vbCrLf
TestOfSignificance_r = TestOfSignificance_r + "df=" + CStr(Return_df) + vbCrLf
TestOfSignificance_r = TestOfSignificance_r + "t=" + CStr(Round(Return_T, 4)) + vbCrLf
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Dim p1 As Double, p2 As Double, p3 As Double
p1 = obj.t(0.005, Return_df)
    'If Return_T >= p1 And Return_T <= p2 Then
    'TestOfSignificance_r = TestOfSignificance_r + "显著的相关系数" + "Aerf0.001=" +
CStr(Round(p2, 4)) + ">" + CStr(Round(Return_T, 4)) + ">" + "Aerf0.05=" + CStr(Round(p1, 4)) +
vbCrLf
    'End If
    'If Return_T >= p2 Then
    'TestOfSignificance_r = TestOfSignificance_r + "极其显著的相关系数" +
CStr(Round(Return_T, 4)) + ">" + "Aerf0.001=" + CStr(Round(p2, 4)) + vbCrLf
    'End If
    If Return_T < p1 Then
    TestOfSignificance_r = TestOfSignificance_r + "不显著的相关系数" + "Aerf0.01=" +
CStr(Round(p1, 4)) + ">" + CStr(Round(Return_T, 4)) + vbCrLf
    End If
    If Return_T >= p1 Then
    TestOfSignificance_r = TestOfSignificance_r + "显著的相关系数" + CStr(Round(Return_T, 4))
+ ">" + "Aerf0.01=" + CStr(Round(p1, 4)) + vbCrLf
    End If
DoEvents
End Function
```

'TestOfSignificance_ρ回归系数显著性

Rem 需要 Statistics_T002 类模块
Rem Spearman correlation coefficient(斯皮尔曼相关性系数)显著性检验
Rem Data1(),Data2()原始数据
Rem r 相关系数
Rem 返回 Return_df 自由度
Rem 返回 Return_T t 值用在显著性检验

```
Public Function TestOfSignificance_ρ(ByVal SimpleNumber As Long, _
ByVal R As Double, ByRef Return_df As Long, _
ByRef Return_T As Double) As String
Dim i As Long, n  As Long
For i = 1 To SimpleNumber
n = n + 1
Next i
If R <> 1 Then
Return_T = R * Sqr((n - 2) / (1 - R ^ 2))
Else
Return_T = 1000
End If
Return_df = n - 2
```

```
TestOfSignificance_ρ = TestOfSignificance_ρ + "r=" + CStr(Round(R, 4)) + vbCrLf
TestOfSignificance_ρ = TestOfSignificance_ρ + "n=" + CStr(n) + vbCrLf
TestOfSignificance_ρ = TestOfSignificance_ρ + "df=" + CStr(Return_df) + vbCrLf
TestOfSignificance_ρ = TestOfSignificance_ρ + "t=" + CStr(Round(Return_T, 4)) + vbCrLf
Dim p1 As Double, p2 As Double, p3 As Double
p1 = obj.t(0.005, Return_df)

    If Return_T < p1 Then
    TestOfSignificance_ρ = TestOfSignificance_ρ + "不显著的相关系数" + "Aerf0.01=" +
CStr(Round(p1, 4)) + ">" + CStr(Round(Return_T, 4)) + vbCrLf
    End If
    If Return_T >= p1 Then
    TestOfSignificance_ρ = TestOfSignificance_ρ + "显著的相关系数" + CStr(Round(Return_T, 4))
+ ">" + "Aerf0.01=" + CStr(Round(p1, 4)) + vbCrLf
    End If
DoEvents
End Function
```

42. FileChange

详细说明：

函数或方法名称	作用
FolderChange	替换文件

代码：

```
Option Explicit
```

Rem 替换文件

Rem FolderNameAndPath=文件路径名称

```
Public Function FolderChange(ChoiceFile, FolderNameAndPath) As Variant
If Dir(FolderNameAndPath, vbDirectory) = "" Then
    MsgBox "没文件出错"
Else
    Dim SourceFile, DestinationFile
    SourceFile = ChoiceFile     ' 指定源文件路径和名。
    DestinationFile = FolderNameAndPath    ' 指定目标路径和文件名。
    If DestinationFile <> SourceFile Then
    FileCopy SourceFile, DestinationFile       ' 将源文件的内容复制到目的文件中
    End If
End If
End Function
```

43.FileSize

详细说明：

函数或方法名称	作用
FolderSize	文件夹大小

代码：

Rem 文件夹大小

Rem FolderNameAndPath=文件路径名称

```
Public Function FolderSize(FolderNameAndPath) As Variant
Dim FSO As Object
If Dir(FolderNameAndPath, vbDirectory) = "" Then
```

```
    MsgBox "没文件出错"
Else
    Set FSO = CreateObject("Scripting.FileSystemObject")
    Set F = FSO.GetFile(FolderNameAndPath)
    FolderSize = F.Size
    Debug.Print FolderNameAndPath
End If
End Function
```

44.FindNum

详细说明：

函数或方法名称	作用
FindNum_Lng	在字符串中找到数字

代码：

```
Option Explicit
Option Base 1
'*****************************************************************************'
Class:      FindNum
' Filename: FindNum.cls
' Author:    zhannghong2002
' Date:      2018 10 30
'*****************************************************************************
Rem 在字符串中找到数字
Sub FindNum_Lng(ByVal Data As Variant, ByRef R_Data() As Long)
Dim A() As Variant
Dim f As Boolean
Dim i As Long, j As Long
Dim n As Long
For i = 1 To Len(Data)
        f = False
        If IsNumeric(Mid(Data, i, 1)) Then
                If i > 1 Then
                    If IsNumeric(Mid(Data, i - 1, 1)) Then
                    f = True
                    A(n) = A(n) & Mid(Data, i, 1)
                    'Debug.Print A(n)
                    End If
                End If
```

```
                    If f = False Then
                        n = n + 1
                        ReDim Preserve A(n)
                        ReDim Preserve R_Data(n)
                        A(n) = Mid(Data, i, 1)
                    End If
        End If
 Next i
If UBound(A) > 0 Then
            For i = 1 To UBound(A)
               R_Data(i) = CLng(A(i))
               'Debug.Print "R_Data(" & i; ")=" & R_Data(i)
            Next i
End If
End Sub
```

45.BinomialDistribution(数学统计)

详细说明：

函数或方法名称	作用
BinomialDistribution	二项式计算
μ	二项分布的随机变量之平均数μ
σ	二项分布标准差σ
Factorial	阶乘

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'**************************************************************************
'名称 = BinomialDistribution.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 2 日
'简介 = 二项分布类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'**************************************************************************
```

'BinomialDistribution 二项式计算

Rem 2 项式计算

Rem 参数 n,p,x

```
Public Function BinomialDistribution(ByVal n As Long, ByVal P As Double, _
ByVal X As Long) As Double
BinomialDistribution = Factorial(n) / _
(Factorial(X) * Factorial(n - X)) * (P ^ X) * ((1 - P) ^ (n - X))
End Function
```

'μ二项分布的随机变量之平均数μ

Rem 二项分布的随机变量之平均数μ

```
Public Function μ(ByVal n As Long, ByVal P As Double) As Double
μ = n * P
End Function
```

'σ二项分布标准差σ

Rem 二项分布标准差σ

```
Public Function σ(ByVal n As Long, ByVal P As Double) As Double
  σ = Sqr(n * P * (1 - P))
End Function
```

'Factorial 阶乘

Rem 阶乘

```
Public Function Factorial(ByVal n As Long) As Double
Dim i As Long
Factorial = 1
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Part1 As Double, Part2 As Double
Part1 = 1
Part2 = 1
If (n > 171) Then
'MsgBox "无法计算 171 以上数字阶乘"
    If (n Mod 2 = 1) Then
        For i = 1 To (n / 2 + 0.5)
        Part1 = Part1 * i
        Next i
```

```
            For i = (n / 2 + 0.5 + 1) To n
            Part2 = Part2 * i
            Next i
            Factorial = Part1 * Part2
        End If
        If (n Mod 2 = 0) Then
            For i = 1 To (n / 2)
            Part1 = Part1 * i
            Next i
            For i = (n / 2 + 1) To n
            Part2 = Part2 * i
            Next i
            Factorial = Part1 * Part2
        End If
Else
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~`
For i = 1 To n
Factorial = Factorial * i
Next i
End If
End Function
```

46.Mean(田间统计)

详细说明：

函数或方法名称	作用
Data	一个计数资料数据
ArithmeticMean	算术平均数
ArithmeticMeanMethodOfWeighting	算术平均数（arithmetic mean）加权法
Median	中位数
MedianMethodOfTable	已分组资料中位数的计算方法
Mode	众数
ModeMethodOfTable	次数分布表的众数
GeometricMean	几何平均数
HarmonicMean	调和平均数
MaxL	整型最大值 ModeMethodOfTable 子程序
Max	Mode 子程序

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*****************************************************************************
'名称 = Mean.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 10 日
'简介 = 资料的集中性描述平均数类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'修改注释                                                    2020-3-10
'*****************************************************************************
```

'Data 一个计数资料数据

Rem 一个计数资料数据

```
Public Sub Data(ByRef D() As Double)
ReDim D(1 To 140) As Double
D(1) = 177: D(2) = 215: D(3) = 197: D(4) = 97: D(5) = 123
D(6) = 159: D(7) = 245: D(8) = 119: D(9) = 119: D(10) = 131
D(11) = 149: D(12) = 152: D(13) = 167: D(14) = 104
D(15) = 161: D(16) = 214: D(17) = 125: D(18) = 175: D(19) = 219
D(20) = 118: D(21) = 192: D(22) = 176: D(23) = 175: D(24) = 95
D(25) = 136: D(26) = 199: D(27) = 116: D(28) = 165
```

```
D(29) = 214: D(30) = 95: D(31) = 158: D(32) = 83: D(33) = 137
D(34) = 80: D(35) = 138: D(36) = 151: D(37) = 187: D(38) = 126
D(39) = 196: D(40) = 134: D(41) = 206: D(42) = 137:
D(43) = 98: D(44) = 97: D(45) = 129: D(46) = 143: D(47) = 179
D(48) = 174: D(49) = 159: D(50) = 165: D(51) = 136: D(52) = 108
D(53) = 101: D(54) = 141: D(55) = 148: D(56) = 168
D(57) = 163: D(58) = 176: D(59) = 102: D(60) = 194: D(61) = 145
D(62) = 173: D(63) = 75: D(64) = 130: D(65) = 149: D(66) = 150
D(67) = 161: D(68) = 155: D(69) = 111: D(70) = 158
D(71) = 131: D(72) = 189: D(73) = 91: D(74) = 142: D(75) = 140
D(76) = 154: D(77) = 152: D(78) = 163: D(79) = 123: D(80) = 205
D(81) = 149: D(82) = 155: D(83) = 131: D(84) = 209
D(85) = 183: D(86) = 97: D(87) = 119: D(88) = 181: D(89) = 149
D(90) = 187: D(91) = 131: D(92) = 215: D(93) = 111: D(94) = 186
D(95) = 118: D(96) = 150: D(97) = 155: D(98) = 197
D(99) = 116: D(100) = 254: D(101) = 239: D(102) = 160: D(103) = 172
D(104) = 179: D(105) = 151: D(106) = 198: D(107) = 124: D(108) = 179
D(109) = 135: D(110) = 184: D(111) = 168: D(112) = 169
D(113) = 173: D(114) = 181: D(115) = 188: D(116) = 211: D(117) = 197
D(118) = 175: D(119) = 122: D(120) = 151: D(121) = 171: D(122) = 166
D(123) = 175: D(124) = 143: D(125) = 190: D(126) = 213
D(127) = 192: D(128) = 231: D(129) = 163: D(130) = 159: D(131) = 158
D(132) = 159: D(133) = 177: D(134) = 147: D(135) = 194: D(136) = 227
D(137) = 141: D(138) = 169: D(139) = 124: D(140) = 159
End Sub
```

'ArithmeticMean 算术平均数

Rem 算术平均数（arithmetic mean）
Rem Data()=输入数据
Rem ReturnSum=返回总值

```
Public Function ArithmeticMean(ByRef Data() As Double, _
ByRef ReturnSum As Double) As Double
Dim Sum As Double, l As Long, i As Long
For i = LBound(Data) To UBound(Data)
Sum = Sum + Data(i)
l = l + 1
Next i
ArithmeticMean = Sum / l
ReturnSum = Sum
End Function
```

'ArithmeticMeanMethodOfWeighting 算术平均数（arithmetic mean）加权法

Rem 算术平均数（arithmetic mean）加权法
Rem ClassMidValue()=组中值
Rem NumberOfTimes()=次数
Rem Clusters=组数

```
Public Function ArithmeticMeanMethodOfWeighting(ByRef ClassMidValue() As Double, _
ByRef NumberOfTimes() As Long, ByVal Clusters As Long) As Double
Dim i As Long
Dim Sum1 As Double, Sum2 As Double
Dim Sum3 As Long
If ((UBound(ClassMidValue) - LBound(ClassMidValue)) = (UBound(NumberOfTimes) -
LBound(NumberOfTimes))) Then
For i = LBound(ClassMidValue) To UBound(ClassMidValue)
Sum1 = Sum1 + ClassMidValue(i) * NumberOfTimes(i)
Sum2 = Sum2 + NumberOfTimes(i)
Sum3 = Sum3 + 1
Next i
ArithmeticMeanMethodOfWeighting = Sum1 / Sum2

Rem 自检程序
If (Clusters <> 0) Then
If (Sum3 <> Clusters) Then
MsgBox "数据有问题", vbCritical
End If
End If
Else
MsgBox "数据有问题", vbCritical
End If
End Function
```

'Median 中位数

Rem 中位数（median）
Rem Data()=输入数据
Rem ReturnTotalNumbersOdd=返回奇偶性

```
Public Function Median(ByRef Data() As Double, _
ByVal ReturnTotalNumbersOdd As Boolean) As Double 'Odd 奇数
Dim i As Long, n As Long, m As Long
n = 0
```

```
For i = LBound(Data) To UBound(Data)
'Debug.Print "data(" & i & ")=" & Data(i)
n = n + 1
Next
m = 1 - LBound(Data)
If (n Mod 2 = 1) Then
Median = Data(((n + 1) / 2) - m)
ReturnTotalNumbersOdd = True
End If
If (n Mod 2 = 0) Then
Median = (Data((n / 2) - m) + Data((n / 2 + 1) - m)) / 2
ReturnTotalNumbersOdd = False
End If
End Function
```

'MedianMethodOfTable 已分组资料中位数的计算方法

Rem 已分组资料中位数的计算方法；
Rem 编制成次数分布表，则可利用次数分布表来计算中位数；
Rem LBoundOfTheGroupWhichMedianIn=中位数所在组的下限；
Rem ClassInterval=组距；
Rem TimesOfTheGroupWhichMedianIn =中位数所在组的次数；
Rem TotalTimes=总次数；
Rem TimesOfTheGroupsWhichSmallerThanMedianIn=小于中数所在组的累加次数

```
Public Function MedianMethodOfTable(ByRef LBoundOfTheGroupWhichMedianIn As Double, _
ByVal ClassInterval As Double, ByVal TimesOfTheGroupWhichMedianIn As Double, _
ByVal TotalTimes As Double, ByVal TimesOfTheGroupsWhichSmallerThanMedianIn) As Double
MedianMethodOfTable = LBoundOfTheGroupWhichMedianIn + ClassInterval / _
TimesOfTheGroupWhichMedianIn * (TotalTimes / 2 -
TimesOfTheGroupsWhichSmallerThanMedianIn)
End Function
```

'Mode 众数

Rem 众数（**mode**）
Rem Data()=输入数据
Rem ReturnMode()=返回不唯一的众数。

```
Public Function Mode(ByRef Data() As Double, ByRef ReturnMode() As Double) As Double
Dim i As Long, l As Long, n As Long, m As Long
Dim CompareData() As Double
For i = LBound(Data) To UBound(Data)
m = m + 1
```

```
ReDim Preserve CompareData(m)
n = 0
For l = (i + 1) To UBound(Data)
'Debug.Print "data(" & i & ")=" & Data(i) & "    data(" & L & ")=" & Data(L)
If (Data(i) = Data(l)) Then
n = n + 1
End If
Next l
CompareData(m) = n
'Debug.Print "CompareData(" + CStr(m) + ")=" + CStr(CompareData(m))
Next i
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim O As Long, P As Long
For i = 1 To UBound(CompareData)
If (CompareData(i) = Max(CompareData())) Then
O = O + 1
'~~~~~~~~~~~~~~~~~~~~~~~~
P = i - (1 - LBound(Data()))                    '可能有问题
'~~~~~~~~~~~~~~~~~~~~~~~~
Mode = Data(P)
ReDim Preserve ReturnMode(O)
ReturnMode(O) = Data(P)
End If
Next i
If (O > 1) Then
MsgBox "有 2 个或以上值，数据在 ReturnMode()", vbCritical
End If
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'MsgBox "非组中值众数"
End Function
```

'ModeMethodOfTable 次数分布表的众数

Rem 次数分布表的众数（mode）
Rem Times()=次数
Rem ClassMidValue() =组中值

```
Public Function ModeMethodOfTable(ByRef Times() As Long, _
ByRef ClassMidValue() As Double) As Double
Dim i As Long, l As Long, n As Long, m As Long
For i = LBound(Times) To UBound(Times)
    If Times(i) = MaxL(Times) Then
    ModeMethodOfTable = ClassMidValue(i)
    n = n + 1
```

```
    End If
Next i
If n > 1 Then
    MsgBox "存在" & n & "个众数"
End If
End Function
```

'GeometricMean 几何平均数

Rem 几何平均数 geometric mean
Rem Data()=输入数据

```
Public Function GeometricMean(ByRef Data() As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
n = 0
For i = LBound(Data()) To UBound(Data())
If Data(i) = 0 Then
MsgBox "GeometricMean 数据存在 0,不能计算几何平均数,以 0 位结果", vbCritical
MsgBox "用 1E-150 继续运算"
Data(i) = 1E-150
End If
Sum1 = Sum1 + Log(Data(i))
n = n + 1
Next i
GeometricMean = Exp(Sum1 / n)
End Function
```

'HarmonicMean 调和平均数

Rem 调和平均数（harmonic mean）
Rem Data()=输入数据

```
Public Function HarmonicMean(ByRef Data() As Double) As Double
Dim Sum As Double, n As Long, i As Long
Dim Part1 As Double
For i = LBound(Data) To UBound(Data)
If Data(i) = 0 Then
MsgBox "HarmonicMean 数据存在 0,不能计算调和平均数,以 0 位结果", vbCritical
MsgBox "用 1E-150 继续运算"
Data(i) = 1E-150
End If
Sum = Sum + 1 / Data(i)
n = n + 1
Next
```

```
Part1 = Sum / n
HarmonicMean = 1 / Part1
End Function
```

Rem ModeMethodOfTable 子程序

Rem 整型最大值

```
Public Function MaxL(ByRef Data() As Long) As Double
Dim i As Long
Dim cData() As Long
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next i
For i = LBound(cData()) To UBound(cData())
    If i < UBound(cData()) Then
        If cData(i) > cData(i + 1) Then
            MaxL = cData(i)
            cData(i + 1) = cData(i)
        Else
            MaxL = cData(i + 1)
        End If
    End If
Next i
End Function
```

Rem Mode 子程序

Rem 双精度型最大值

```
Public Function Max(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next i

For i = LBound(cData()) To UBound(cData())
    If i < UBound(cData()) Then
        If cData(i) > cData(i + 1) Then
            Max = cData(i)
            cData(i + 1) = cData(i)
        Else
            Max = cData(i + 1)
        End If
    End If
Next i
End Function
```

47.NormalDistributionAccuracy(田间统计)

详细说明：

函数或方法名称	作用
NormalDistributionPosibility	正态分布概率
NormalDistributionPosibilityAccuracy	正态分布概率准确值的确定
NormalDistributionPosibilityOutRectangle	正态分布概率外矩形法 NormalDistributionPosibilityAccuracy 子程序
NormalDistributionPosibilityInRectangle	正态分布概率内矩形法 NormalDistributionPosibilityAccuracy 子程序
Rectangle	多程序子程序
Rectangle1	多程序子程序

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'**************************************************************************
'名称 = NormalDistributionAccuracy.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 18 日
'简介 = 标准正太分布表类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'修改 NormalDistributionPosibility 中循环 i 的上限为 9        2020-1-14
'修改注解                                                    2020-3-14
'修改 NormalDistributionPosibility Goto 换成 Exit            2020-3-18
'**************************************************************************
Const conPi = 3.14159265358979
Const conE = 2.71828182845905
```

'NormalDistributionPosibility 正态分布概率

```
Rem 正态分布概率
Rem μ=位置参数
Rem σ=函数胖瘦参数
Rem μ1,μ2=积分下限，积分上限
Rem ChoiceDecimals= 规定小数范围
Public Function NormalDistributionPosibility(ByVal μ As Double, _
ByVal σ As Double, ByVal μ1 As Double, ByVal μ2 As Double, ByVal ChoiceDecimals As Long) As
```

```
Double
Dim i As Long, ReturnDecimals As Long
For i = 3 To 9 '分块数 9 位以后等待时间长，暂时不支持。
NormalDistributionPosibility = NormalDistributionPosibilityAccuracy(μ, σ, 10 ^ i, μ1, μ2,
ReturnDecimals)
    If ReturnDecimals >= ChoiceDecimals Then
    Exit Function '2020-3-18 修改
    End If
    If i = 9 Then
    MsgBox "超出最高精确度范围，现有最佳精确度位数数值为" & ReturnDecimals
    Exit Function '2020-3-18 修改
    End If
Next i
End Function
```

'NormalDistributionPosibilityAccuracy 正态分布概率准确值的确定

Rem 正态分布概率准确值的确定
Rem μ=位置参数
Rem σ=函数胖瘦参数
Rem PrecisionPieces=分块数值不取 0
Rem μ1,μ2=积分下限，积分上限
Rem 函数会出现范围值越小、分割块数越多越准确。所以还是需要正态分布表加以校对
Rem ReturnDecimals =返回精确位数

```
Public Function NormalDistributionPosibilityAccuracy(ByVal μ As Double, _
ByVal σ As Double, ByVal PrecisionPieces As Double, ByVal μ1 As Double, _
ByVal μ2 As Double, ByRef ReturnDecimals As Long) As Double
Dim S1 As Double, S2 As Double, S3 As Double, S4 As Double
Dim i As Long, n As Long
S1 = NormalDistributionPosibilityOutRectangle(μ, σ, PrecisionPieces, μ1, μ2)
S2 = NormalDistributionPosibilityInRectangle(μ, σ, PrecisionPieces, μ1, μ2)
If S2 <> 0 Then
        Rem 获得小数位数
        If S2 < 1 Then
        Do
        n = n + 1
        S3 = S2 * (10 ^ n)
        Loop While S3 < 1
        S4 = S1 * 10 ^ n
        End If
        Rem 获得内外矩形法四舍六入五逢双相等时候的小数位数
```

```
        For i = 22 To 0 Step -1 '这里最大值 22，大于该值会显示参数无法调用
                If Round(S3, i) = Round(S4, i) Then
                NormalDistributionPosibilityAccuracy = Round(S3, i) * (10 ^ (-n))
                ReturnDecimals = n + i
                Exit For
                End If
        Next i
Else
NormalDistributionPosibilityAccuracy = 0
ReturnDecimals = 0
End If
If Round(S1, 1) <> Round(S2, 1) Then
'MsgBox "需要增加 PrecisionPieces 值"
End If
If S1 = S2 And S1 = 0 Then '都等于 0 的特殊情况
NormalDistributionPosibilityAccuracy = 0
ReturnDecimals = 999
End If
End Function
```

Rem NormalDistributionPosibilityOutRectangle 正态分布概率外矩形法
NormalDistributionPosibilityAccuracy 子程序
Rem 正态分布概率外矩形法
Rem μ=位置参数
Rem σ=函数胖瘦参数
Rem PrecisionPieces=分块数值不取 0
Rem μ1,μ2=积分下限，积分上限
Rem 函数会出现范围值越小、分割块数越多越准确。所以还是需要正态分布表加以校对

```
Public Function NormalDistributionPosibilityOutRectangle(ByVal μ As Double, _
ByVal σ As Double, ByVal PrecisionPieces As Double, ByVal μ1 As Double, _
ByVal μ2 As Double) As Double
Dim X As Double, i As Long, Distance As Double
Dim NDP As Double, μD As Double
Dim Part1 As Double
μD = μ2 - μ1
Distance = μD / PrecisionPieces
If μ2 <= μ Then '积分上限小于等于正太函数对称中点的情况
Rectangle1 μ2, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityOutRectangle
End If
If μ1 >= μ Then '积分下限大于等于正太函数对称中点的情况
Rectangle μ1, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityOutRectangle
End If
If μ1 < μ And μ < μ2 Then '正太函数对称中点在积分上下限之间的情况
μD = μ - μ1
Distance = μD / PrecisionPieces
```

```
Rectangle μ, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityOutRectangle
Part1 = NormalDistributionPosibilityOutRectangle
NormalDistributionPosibilityOutRectangle = 0
μD = μ2 - μ
Distance = μD / PrecisionPieces
Rectangle1 μ, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityOutRectangle
NormalDistributionPosibilityOutRectangle = NormalDistributionPosibilityOutRectangle + Part1
End If
DoEvents
End Function
```

Rem NormalDistributionPosibilityInRectangle 正态分布概率内矩形法
NormalDistributionPosibilityAccuracy 子程序
Rem 正态分布概率内矩形法
Rem μ=位置参数
Rem σ=函数胖瘦参数
Rem PrecisionPieces=分块数值不取 0
Rem μ1,μ2=积分下限，积分上限
Rem 函数会出现范围值越小、分割块数越多越准确。所以还是需要正态分布表加以校对

```
Public Function NormalDistributionPosibilityInRectangle(ByVal μ As Double, _
ByVal σ As Double, ByVal PrecisionPieces As Double, ByVal μ1 As Double, _
ByVal μ2 As Double) As Double
Dim X As Double, i As Long, Distance As Double
Dim NDP As Double, μD As Double
Dim Part1 As Double
μD = μ2 - μ1
Distance = μD / PrecisionPieces
If μ2 <= μ Then '积分上限小于等于正太函数对称中点的情况
Rectangle μ1, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityInRectangle
End If
If μ1 >= μ Then '积分下限大于等于正太函数对称中点的情况
Rectangle1 μ2, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityInRectangle
End If
If μ1 < μ And μ < μ2 Then '正太函数对称中点在积分上下限之间的情况
μD = μ - μ1
Distance = μD / PrecisionPieces
Rectangle μ1, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityInRectangle
Part1 = NormalDistributionPosibilityInRectangle
NormalDistributionPosibilityInRectangle = 0
μD = μ2 - μ
Distance = μD / PrecisionPieces
Rectangle1 μ2, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityInRectangle
NormalDistributionPosibilityInRectangle = NormalDistributionPosibilityInRectangle + Part1
End If
DoEvents
```

```
End Function
```

Rem 多程序子程序

Rem 正向使用矩形法

```
Sub Rectangle(μ1, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityInRectangle)
Dim X As Double, i As Long, Part1 As Double
For i = 0 To PrecisionPieces - 1
X = μ1 + Distance * i
Part1 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X - μ) ^ 2) / (2 * σ ^ 2))
NormalDistributionPosibilityInRectangle = NormalDistributionPosibilityInRectangle _
+ Part1 * Distance
Next i
End Sub
```

Rem 多程序子程序

Rem 逆向使用矩形法

```
Sub Rectangle1(μ2, σ, μ, PrecisionPieces, Distance, NormalDistributionPosibilityInRectangle)
Dim X As Double, i As Long, Part1 As Double
For i = 0 To PrecisionPieces - 1
X = μ2 - Distance * i
Part1 = (1 / (σ * Sqr(2 * conPi))) * conE ^ -(((X - μ) ^ 2) / (2 * σ ^ 2))
NormalDistributionPosibilityInRectangle = NormalDistributionPosibilityInRectangle + Part1 *
Distance
Next i
End Sub
```

48.PoissonDistribution(田间统计)

详细说明：

函数或方法名称	作用
PoissonDistribution	泊松分布计算
Factorial	阶乘

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*****************************************************************************
'名称 = PoissonDistribution.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 2 日
'简介 = 泊松分布类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'*****************************************************************************
Const conE = 2.71828182845905
```

'PoissonDistribution 泊松分布计算

Rem 泊松分布计算

```
Public Function PoissonDistribution(ByVal λ As Double, ByVal k As Double) As Double
PoissonDistribution = (λ ^ k / Factorial(k)) * (conE ^ (-λ))
End Function
```

'Factorial 阶乘

Rem 阶乘

```
Public Function Factorial(ByVal n As Long) As Double
Dim i As Long
Factorial = 1
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim Part1 As Double, Part2 As Double
Part1 = 1
Part2 = 1
If (n > 171) Then
```

```
'MsgBox "无法计算 171 以上数字阶乘"
    If (n Mod 2 = 1) Then
        For i = 1 To (n / 2 + 0.5)
        Part1 = Part1 * i
        Next i
        For i = (n / 2 + 0.5 + 1) To n
        Part2 = Part2 * i
        Next i
        Factorial = Part1 * Part2
    End If
    If (n Mod 2 = 0) Then
        For i = 1 To (n / 2)
        Part1 = Part1 * i
        Next i
        For i = (n / 2 + 1) To n
        Part2 = Part2 * i
        Next i
        Factorial = Part1 * Part2
    End If
Else
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~`
For i = 1 To n
Factorial = Factorial * i
Next i
End If
End Function
```

49.FDistribution(田间统计)

详细说明：

函数或方法名称	作用
F	返回 F 分布值表法
SubFLinearInterPolation	F 分析值线性插值法 F 分布的子程序
SubDf2Have	Df2 存在表值 FLinearInterpolation 的子程序
SubDf1Have	Df1 存在表值 FLinearInterpolation 的子程序
SubNotHave	Df1 和 Df2 都不存在表值 FLinearInterpolation 的子程序
SubFHead	多程序子程序
SubHead	多程序子程序
SubFHeadFindDF	F 子程序
SubInF	多程序子程序

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*****************************************************************************
'名称 = FDistribution.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 10 日
'简介= F 分布查表法
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=1.工程需要添加 FileRead190505 类模块
'操作历史：
'修改程序结构                                                    2020-2-26
'修改注解                                                        2020-3-10
'*****************************************************************************
Dim Obj As New S4_FileRead190505 '会使用到文件读取类模块
```

'F 返回 F 分布值表法

```
Rem 返回 F 分布表法
Rem Df1=传入自由度 1
Rem Df2=传入自由度 2
Rem Aerf=传入阿尔法值
Public Function F(ByRef Df1 As Long, ByRef Df2 As Long, ByVal Aerf As Double) As Double
Dim Name As String, R2D() As Variant, R2DT() As Variant
```

```
Dim RDf1 As Long, RDf2 As Long
Rem  读取 txt 文件中的数据
If Aerf = 0.05 Then
Name = App.Path + "/程序用表/F005.txt"
Obj.FileReadByLineTexT2D Name, R2D, R2DT
End If
If Aerf = 0.01 Then
Name = App.Path + "/程序用表/F001.txt"
Obj.FileReadByLineTexT2D Name, R2D, R2DT
End If
SubFHeadFindDF Df1, Df2, RDf1, RDf2
If RDf1 = 0 Or RDf2 = 0 Then
F = SubFLinearInterPolation(Df1, Df2, Aerf) '启用线性插值法
Else
F = R2DT(RDf1, RDf2) '直接提取 txt 中的数据
End If
End Function
```

Rem F 的子程序

Rem F 分析值线性插值法

Rem Df1=传入自由度 1

Rem Df2=传入自由度 2

Rem Aerf=传入阿尔法值

```
Private Function SubFLinearInterPolation(ByVal Df1 As Long, ByVal Df2 As Long, _
ByVal Aerf As Double) As Double
Dim Df1L As Long, Df1U As Long, Df2L As Long, Df2U As Long
Dim LNum As Double, UNum As Double
Dim HaveDf1 As Boolean, HaveDf2 As Boolean
Df1L = Df1: Df1U = Df1: Df2L = Df2: Df2U = Df2
Call SubFHead(Df1, Df2, HaveDf1, HaveDf2)
SubDf2Have HaveDf1, HaveDf2, Df1L, Df1U, Df1, Df2, Aerf, LNum, UNum,
SubFLinearInterPolation
SubDf1Have HaveDf1, HaveDf2, Df2L, Df2U, Df1, Df2, Aerf, LNum, UNum,
SubFLinearInterPolation
SubNotHave HaveDf1, HaveDf2, Df1L, Df1U, Df2L, Df2U, Df1, Df2, Aerf, LNum, UNum, _
SubFLinearInterPolation
End Function
```

Rem FLinearInterpolation 的子程序

Rem Df2 存在表值

```
Private Function SubDf2Have(ByVal HaveDf1 As Boolean, ByVal HaveDf2 As Boolean, _
ByVal Df1L As Long, ByVal Df1U As Long, ByVal Df1 As Long, ByVal Df2 As Long, _
ByVal Aerf As Double, ByRef LNum As Double, ByRef UNum As Double, _
ByRef SubFLinearInterPolation As Double) As Double
If HaveDf1 = False And HaveDf2 = True Then
    Do
```

```
    Df1L = Df1L - 1
    LNum = SubInF(Df1L, Df2, Aerf)
    Loop While LNum = 0 And Df1L > -1
    Do
    Df1U = Df1U + 1
    UNum = SubInF(Df1U, Df2, Aerf)
    Loop While UNum = 0 And Df1U < 500
    SubFLinearInterPolation = LNum - (Df1 - Df1L) * (LNum - UNum) / (Df1U - Df1L)
End If
End Function
```

Rem FLinearInterpolation 的子程序

Rem Df1 存在表值

```
Private Function SubDf1Have(ByVal HaveDf1 As Boolean, ByVal HaveDf2 As Boolean, _
ByVal Df2L As Long, ByVal Df2U As Long, ByVal Df1 As Long, ByVal Df2 As Long, _
ByVal Aerf As Double, ByRef LNum As Double, ByRef UNum As Double, _
ByRef SubFLinearInterPolation As Double) As Double
If HaveDf1 = True And HaveDf2 = False Then
    Do
    Df2L = Df2L - 1
    LNum = SubInF(Df1, Df2L, Aerf)
    Loop While LNum = 0 And Df2L > -1
    Do
    Df2U = Df2U + 1
    UNum = SubInF(Df1, Df2U, Aerf)
    Loop While UNum = 0 And Df2U < 1000
    SubFLinearInterPolation = (LNum + UNum) / 2
    SubFLinearInterPolation = LNum - (Df2 - Df2L) * (LNum - UNum) / (Df2U - Df2L)
End If
End Function
```

Rem FLinearInterpolation 的子程序

Rem Df1 和 Df2 都不存在表值

```
Private Function SubNotHave(ByVal HaveDf1 As Boolean, ByVal HaveDf2 As Boolean, _
ByVal Df1L As Long, ByVal Df1U As Long, ByVal Df2L As Long, ByVal Df2U As Long, _
ByVal Df1 As Long, ByVal Df2 As Long, ByVal Aerf As Double, ByRef LNum As Double, _
ByRef UNum As Double, ByRef SubFLinearInterPolation As Double) As Double
Dim FLinearInterpolationL As Double, FLinearInterpolationU As Double
If HaveDf1 = False And HaveDf2 = False Then
    Do
    Df1L = Df1L - 1
    Call SubFHead(Df1L, Df2, HaveDf1, HaveDf2)
    Loop While HaveDf1 = False And Df1L > -1
    HaveDf1 = False
    Do
    Df1U = Df1U + 1
```

```
    Call SubFHead(Df1U, Df2, HaveDf1, HaveDf2)
    Loop While HaveDf1 = False And Df1U <= 500
    Do
    Df2L = Df2L - 1
    Call SubFHead(Df1L, Df2L, HaveDf1, HaveDf2)
    Loop While HaveDf2 = False And Df2L > -1
    HaveDf2 = False
    Do
    Df2U = Df2U + 1
    Call SubFHead(Df1U, Df2U, HaveDf1, HaveDf2)
    Loop While HaveDf2 = False And Df2U <= 1000
    LNum = SubInF(Df1L, Df2L, Aerf)
    UNum = SubInF(Df1U, Df2L, Aerf)
    FLinearInterpolationL = LNum - (Df1 - Df1L) * (LNum - UNum) / (Df1U - Df1L)
    LNum = SubInF(Df1L, Df2U, Aerf)
    UNum = SubInF(Df1U, Df2U, Aerf)
    FLinearInterpolationU = LNum - (Df1 - Df1L) * (LNum - UNum) / (Df1U - Df1L)
    SubFLinearInterPolation = FLinearInterpolationL - (Df2 - Df2L) * _
    (FLinearInterpolationL - FLinearInterpolationU) / (Df2U - Df2L)
End If
End Function
```

Rem FLinearInterpolation 的子程序;SubNotHave 的子程序

Rem 自由度对应表头

Rem Df1=传入自由度 1

Rem Df2=传入自由度 2

Rem HaveDf1=返回自由度 1 对应的表头是否存在

Rem HaveDf2=返回自由度 2 对应的表头是否存在

```
Private Sub SubFHead(ByVal Df1 As Long, ByVal Df2 As Long, ByRef HaveDf1 As Boolean, _
ByRef HaveDf2 As Boolean)
Dim i As Long, l As Long
Dim DataHeadDf1() As Long: Dim DataHeadDf2() As Long: Dim DataHeadK() As Long
SubHead DataHeadDf1, DataHeadDf2
Rem 寻找对应表头=========================================
For i = 1 To 23
If DataHeadDf1(i) = Df1 Or Df1 > 500 Then
HaveDf1 = True
Exit For
End If
Next i
For i = 1 To 36
If DataHeadDf2(i) = Df2 Or Df2 > 1000 Then
HaveDf2 = True
Exit For
End If
```

```
Next i
End Sub
```

Rem SubFHead 的子程序；SubFHeadFindDF 的子程序

Rem F 分布表表头

```
Private Sub SubHead(ByRef DataHeadDf1() As Long, ByRef DataHeadDf2() As Long)
ReDim DataHeadDf1(1 To 23): ReDim DataHeadDf2(1 To 36)
DataHeadDf1(1) = 1: DataHeadDf1(2) = 2: DataHeadDf1(3) = 3: DataHeadDf1(4) = 4
DataHeadDf1(5) = 5: DataHeadDf1(6) = 6: DataHeadDf1(7) = 7: DataHeadDf1(8) = 8
DataHeadDf1(9) = 9: DataHeadDf1(10) = 10: DataHeadDf1(11) = 11: DataHeadDf1(12) = 12
DataHeadDf1(13) = 14: DataHeadDf1(14) = 16: DataHeadDf1(15) = 20: DataHeadDf1(16) = 24
DataHeadDf1(17) = 30: DataHeadDf1(18) = 40: DataHeadDf1(19) = 50: DataHeadDf1(20) = 75
DataHeadDf1(21) = 100: DataHeadDf1(22) = 200: DataHeadDf1(23) = 500
DataHeadDf2(1) = 1: DataHeadDf2(2) = 2: DataHeadDf2(3) = 3: DataHeadDf2(4) = 4
DataHeadDf2(5) = 5: DataHeadDf2(6) = 6: DataHeadDf2(7) = 7: DataHeadDf2(8) = 8
DataHeadDf2(9) = 9: DataHeadDf2(10) = 10: DataHeadDf2(11) = 11: DataHeadDf2(12) = 12
DataHeadDf2(13) = 13: DataHeadDf2(14) = 14: DataHeadDf2(15) = 15: DataHeadDf2(16) = 16
DataHeadDf2(17) = 17: DataHeadDf2(18) = 18: DataHeadDf2(19) = 19: DataHeadDf2(20) = 20
DataHeadDf2(21) = 22: DataHeadDf2(22) = 24: DataHeadDf2(23) = 26: DataHeadDf2(24) = 28
DataHeadDf2(25) = 30: DataHeadDf2(26) = 36: DataHeadDf2(27) = 42: DataHeadDf2(28) = 50
DataHeadDf2(29) = 60: DataHeadDf2(30) = 70: DataHeadDf2(31) = 80: DataHeadDf2(32) = 100
DataHeadDf2(33) = 150: DataHeadDf2(34) = 200: DataHeadDf2(35) = 400: _
DataHeadDf2(36) = 1000
End Sub
```

Rem F 子程序

Rem 通过自由度找到表头

Rem 输入表头 Num1，Num2

Rem 返回表头对应 df 数值 ReturnDf1,ReturnDf2

```
Private Sub SubFHeadFindDF(ByVal Num1 As Long, ByVal Num2 As Long, _
ByRef ReturnDf1 As Long, ByRef ReturnDf2 As Long)
Dim i As Long
Dim DataHeadDf1() As Long: Dim DataHeadDf2() As Long: Dim DataHeadK() As Long
Rem 按照参考书表给定的表头
SubHead DataHeadDf1, DataHeadDf2
Rem 找表头程序==========================================
ReturnDf1 = 0: ReturnDf2 = 0
For i = 1 To UBound(DataHeadDf1)
If Num1 = DataHeadDf1(i) Then
ReturnDf1 = i
Exit For
End If
Next i
For i = 1 To UBound(DataHeadDf2)
If Num2 = DataHeadDf2(i) Then
ReturnDf2 = i
```

```
Exit For
End If
Next i
Rem  特殊情况==========================================
If Num1 > 500 Then
ReturnDf1 = 23 '表头号
End If
If Num2 > 1000 Then
ReturnDf2 = 36 '表头号
End If
End Sub
```

Rem SubDf2Have 的子程序;SubDf1Have 的子程序;SubNotHave 的子程序

Rem SubInF=返回 F 分布表值

Rem Df1=传入自由度 1

Rem Df2=传入自由度 2

Rem Aerf=传入阿尔法值

```
Private Function SubInF(ByRef Df1 As Long, ByRef Df2 As Long, ByVal Aerf As Double) As Double
Dim Name As String, R2D() As Variant, R2DT() As Variant
Dim RDf1 As Long, RDf2 As Long
If Aerf = 0.05 Then
Name = App.Path + "/程序用表/F005.txt"
Obj.FileReadByLineTexT2D Name, R2D, R2DT
End If
If Aerf = 0.01 Then
Name = App.Path + "/程序用表/F001.txt"
Obj.FileReadByLineTexT2D Name, R2D, R2DT
End If
SubFHeadFindDF Df1, Df2, RDf1, RDf2
If RDf1 = 0 Or RDf2 = 0 Then
SubInF = 0
Else
SubInF = R2DT(RDf1, RDf2)
End If
End Function
```

50.ChiDistribution(田间统计)

详细说明：

函数或方法名称	作用
Chi	返回 Chi 值表法
SubTLinearInterPolation	T 的子程序
SubTHead	多程序子程序
SubTHeadFindDF	多程序子程序
SubInT	SubDfHave 的子程序

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'**************************************************************************
'名称 = S4_ChiDistribution.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 7 月 29 日
'简介= Chi 查表法
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=1.工程需要添加 FileRead190505 类模块
'操作历史：
'修改程序结构                                                    2020-2-26
'修改注解                                                        2020-3-10
'**************************************************************************
Dim Obj As New S4_FileRead190505 '会使用到文件读取类模块
```

'Chi 返回 Chi 值表法

Rem 返回 Chi 值表法
Rem Df=传入自由度
Rem Aerf=传入阿尔法值

```
Public Function Chi(ByVal Aerf As Double, ByRef Df As Long) As Double
Dim Name As String, R1D() As Variant
Dim RDf As Long
Rem 读取 txt 文件中的数据
If Aerf = 0.05 Then
Name = App.Path + "/程序用表/Chi005.txt"
Obj.FileReadByLineTexT Name, R1D
End If
```

```
If Aerf = 0.01 Then
Name = App.Path + "/程序用表/Chi001.txt"
Obj.FileReadByLineTexT Name, R1D
End If
SubTHeadFindDF Df, RDf
If RDf = 0 Then
Chi = SubTLinearInterPolation(Df, Aerf) '启用线性插值法
Else
Chi = R1D(RDf) '直接提取 txt 中的数据
End If
End Function
```

Rem T 的子程序

Rem T 分析值线性插值法

Rem Df=传入自由度

Rem Aerf=传入阿尔法值

```
Private Function SubTLinearInterPolation(ByVal Df As Long, ByVal Aerf As Double) As Double
Dim DfL As Long, DfU As Long
Dim LNum As Double, UNum As Double
DfL = Df: DfU = Df
    Do
    DfL = DfL - 1
    LNum = SubInT(DfL, Aerf)
    Loop While LNum = 0 And DfL > -1
    Do
    DfU = DfU + 1
    UNum = SubInT(DfU, Aerf)
    Loop While UNum = 0 And DfU < 100
    SubTLinearInterPolation = LNum - (Df - DfL) * (LNum - UNum) / (DfU - DfL)
End Function
```

Rem TLinearInterpolation 的子程序;SubNotHave 的子程序

Rem 自由度对应表头

Rem Df=传入自由度

Rem HaveDf=返回自由度对应的表头是否存在

```
Private Sub SubTHead(ByVal Df As Long, ByRef HaveDf As Boolean)
Dim i As Long, l As Long
Dim DataHeadDf() As Long: Dim DataHeadK() As Long
SubHead DataHeadDf
Rem 寻找对应表头=========================================
For i = 1 To 37
If DataHeadDf(i) = Df Or Df > 100 Then
HaveDf = True
Exit For
End If
Next i
```

```
End Sub
```

Rem SubTHead 的子程序；SubTHeadFindDF 的子程序

Rem F 分布表表头

```
Private Sub SubHead(ByRef DataHeadDf() As Long)
ReDim DataHeadDf(1 To 37)
DataHeadDf(1) = 1: DataHeadDf(2) = 2: DataHeadDf(3) = 3: DataHeadDf(4) = 4
DataHeadDf(5) = 5: DataHeadDf(6) = 6: DataHeadDf(7) = 7: DataHeadDf(8) = 8
DataHeadDf(9) = 9: DataHeadDf(10) = 10: DataHeadDf(11) = 11: DataHeadDf(12) = 12
DataHeadDf(13) = 13: DataHeadDf(14) = 14: DataHeadDf(15) = 15: DataHeadDf(16) = 16
DataHeadDf(17) = 17: DataHeadDf(18) = 18: DataHeadDf(19) = 19: DataHeadDf(20) = 20
DataHeadDf(21) = 21: DataHeadDf(22) = 22: DataHeadDf(23) = 23: DataHeadDf(24) = 24
DataHeadDf(25) = 25: DataHeadDf(26) = 26: DataHeadDf(27) = 27: DataHeadDf(28) = 28
DataHeadDf(29) = 29: DataHeadDf(30) = 30: DataHeadDf(31) = 40: DataHeadDf(32) = 50
DataHeadDf(33) = 60: DataHeadDf(34) = 70: DataHeadDf(35) = 80: DataHeadDf(36) = 90
DataHeadDf(37) = 100
End Sub
```

Rem 通过自由度找到表头

Rem 输入表头 Num1

Rem 返回表头对应 df 数值 ReturnDf

```
Private Sub SubTHeadFindDF(ByVal Num1 As Long, ByRef ReturnDf As Long)
Dim i As Long
Dim DataHeadDf() As Long: Dim DataHeadK() As Long
Rem 按照参考书表给定的表头
SubHead DataHeadDf
Rem 找表头程序==========================================
ReturnDf = 0
For i = 1 To UBound(DataHeadDf)
If Num1 = DataHeadDf(i) Then
ReturnDf = i
Exit For
End If
Next i
Rem 特殊情况==========================================
If Num1 > 100 Then
ReturnDf = 37 '表头号
MsgBox Num1 & "超过 100,最大自由度 100，以 100 继续计算", vbCritical
End If
End Sub
```

Rem SubDfHave 的子程序

Rem SubInT=返回 F 分布表值

Rem Df=传入自由度

Rem Aerf=传入阿尔法值

```
Private Function SubInT(ByRef Df As Long, ByVal Aerf As Double) As Double
Dim Name As String, R1D() As Variant
```

```
Dim RDf As Long
If Aerf = 0.05 Then
Name = App.Path + "/程序用表/Chi005.txt"
Obj.FileReadByLineTexT Name, R1D
End If
If Aerf = 0.01 Then
Name = App.Path + "/程序用表/Chi001.txt"
Obj.FileReadByLineTexT Name, R1D
End If
SubTHeadFindDF Df, RDf
If RDf = 0 Then
SubInT = 0
Else
SubInT = R1D(RDf)
End If
End Function
```

51. TDistribution(田间统计)

详细说明：

函数或方法名称	作用
t	t 分布表法
SubTLinearInterPolation	t 分布子程序
SubTHead	多程序子程序
SubHead	多程序子程序
SubTHeadFindDF	多程序子程序

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*****************************************************************************
'名称 = TDistribution.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 3 月 10 日
'简介= T 分布查表法
'声明 = 程序中的英文只做代号，不是标准翻译
'使用说明=1.工程需要添加 FileRead190505 类模块
'操作历史：
'修改程序结构                                                    2020-2-26
'修改注解                                                        2020-3-10
'*****************************************************************************
Dim Obj As New S4_FileRead190505 '会使用到文件读取类模块
```

't 分布表法

```
Rem 返回 T 分布表法
Rem Df=传入自由度
Rem Aerf=传入阿尔法值
Public Function t(ByVal Aerf As Double, ByRef Df As Long) As Double
Dim Name As String, R1D() As Variant
Dim RDf As Long
Rem 读取 txt 文件中的数据
If Aerf = 0.05 Then
Name = App.Path + "/程序用表/T005.txt"
Obj.FileReadByLineTexT Name, R1D
End If
```

```
If Aerf = 0.01 Then
Name = App.Path + "/程序用表/T001.txt"
Obj.FileReadByLineTexT Name, R1D
End If
SubTHeadFindDF Df, RDf
If RDf = 0 Then
t = SubTLinearInterPolation(Df, Aerf) '启用线性插值法
Else
t = R1D(RDf) '直接提取 txt 中的数据
End If
End Function
```

Rem T 的子程序

Rem T 分析值线性插值法

Rem Df=传入自由度

Rem Aerf=传入阿尔法值

```
Private Function SubTLinearInterPolation(ByVal Df As Long, ByVal Aerf As Double) As Double
Dim DfL As Long, DfU As Long
Dim LNum As Double, UNum As Double
DfL = Df: DfU = Df
    Do
    DfL = DfL - 1
    LNum = SubInT(DfL, Aerf)
    Loop While LNum = 0 And DfL > -1
    Do
    DfU = DfU + 1
    UNum = SubInT(DfU, Aerf)
    Loop While UNum = 0 And DfU < 120
    SubTLinearInterPolation = LNum - (Df - DfL) * (LNum - UNum) / (DfU - DfL)
End Function
```

Rem TLinearInterpolation 的子程序;SubNotHave 的子程序

Rem 自由度对应表头

Rem Df=传入自由度

Rem HaveDf=返回自由度对应的表头是否存在

```
Private Sub SubTHead(ByVal Df As Long, ByRef HaveDf As Boolean)
Dim i As Long, l As Long
Dim DataHeadDf() As Long: Dim DataHeadK() As Long
SubHead DataHeadDf
Rem 寻找对应表头=========================================
For i = 1 To 41
If DataHeadDf(i) = Df Or Df > 120 Then
HaveDf = True
Exit For
End If
Next i
```

```
End Sub
```

Rem SubTHead 的子程序；SubTHeadFindDF 的子程序

Rem F 分布表表头

```
Private Sub SubHead(ByRef DataHeadDf() As Long)
ReDim DataHeadDf(1 To 42)
DataHeadDf(1) = 1: DataHeadDf(2) = 2: DataHeadDf(3) = 3: DataHeadDf(4) = 4
DataHeadDf(5) = 5: DataHeadDf(6) = 6: DataHeadDf(7) = 7: DataHeadDf(8) = 8
DataHeadDf(9) = 9: DataHeadDf(10) = 10: DataHeadDf(11) = 11: DataHeadDf(12) = 12
DataHeadDf(13) = 13: DataHeadDf(14) = 14: DataHeadDf(15) = 15: DataHeadDf(16) = 16
DataHeadDf(17) = 17: DataHeadDf(18) = 18: DataHeadDf(19) = 19: DataHeadDf(20) = 20
DataHeadDf(21) = 21: DataHeadDf(22) = 22: DataHeadDf(23) = 23: DataHeadDf(24) = 24
DataHeadDf(25) = 25: DataHeadDf(26) = 26: DataHeadDf(27) = 27: DataHeadDf(28) = 28
DataHeadDf(29) = 29: DataHeadDf(30) = 30: DataHeadDf(31) = 35: DataHeadDf(32) = 40
DataHeadDf(33) = 45: DataHeadDf(34) = 50: DataHeadDf(35) = 55: DataHeadDf(36) = 60
DataHeadDf(37) = 70: DataHeadDf(38) = 80: DataHeadDf(39) = 90: DataHeadDf(40) = 100
DataHeadDf(41) = 120: DataHeadDf(42) = 121
End Sub
```

Rem 多程序子程序

Rem 通过自由度找到表头

Rem 输入表头 Num1

Rem 返回表头对应 df 数值 ReturnDf

```
Private Sub SubTHeadFindDF(ByVal Num1 As Long, ByRef ReturnDf As Long)
Dim i As Long
Dim DataHeadDf() As Long: Dim DataHeadK() As Long
Rem 按照参考书表给定的表头
SubHead DataHeadDf
Rem 找表头程序==========================================
ReturnDf = 0
For i = 1 To UBound(DataHeadDf)
If Num1 = DataHeadDf(i) Then
ReturnDf = i
Exit For
End If
Next i
Rem 特殊情况==========================================
If Num1 > 120 Then
ReturnDf = 42
End If
End Sub
```

Rem SubDfHave 的子程序

Rem SubInT=返回 F 分布表值

Rem Df=传入自由度

Rem Aerf=传入阿尔法值

```
Private Function SubInT(ByRef Df As Long, ByVal Aerf As Double) As Double
```

```
Dim Name As String, R1D() As Variant
Dim RDf As Long
If Aerf = 0.05 Then
Name = App.Path + "/程序用表/T005.txt"
Obj.FileReadByLineTexT Name, R1D
End If
If Aerf = 0.01 Then
Name = App.Path + "/程序用表/T001.txt"
Obj.FileReadByLineTexT Name, R1D
End If
SubTHeadFindDF Df, RDf
If RDf = 0 Then
SubInT = 0
Else
SubInT = R1D(RDf)
End If
End Function
```

52. Variance(田间统计)

详细说明：

函数或方法名称	作用
Data	一个计数资料数据
Range	全距
Max	最大值 Range 子程序
Mix	最小值 Range 子程序
MeanSquare	样本方差
StandardDeviationMethodOfWeighting	样本标准差加权法
CoefficientOfVariation	变异系数
Squareσ	总体方差

代码：

```
Option Explicit '强制显式声明模块中的所有变量,即变量只有声明后才能使用。
Option Base 1 '指定声明数组时下标下界省略时的默认值，这里为 1
'*****************************************************************************
'名称 = Variance.Cls
'作者 = 张宏
'最后修改时间 = 2020 年 6 月 21 日
'简介 = 资料的离散性描述变异数类模块
'声明 = 程序中的英文只做代号，不是标准翻译
'操作历史：
'修改注释
2020-3-10
'修改 MeanSquare 去掉 Exit Function 直接 End 2020-6-21
'修改 StandardDeviationMethodOfWeighting 去掉 Exit Function 直接 End 2020-6-21
'修改 CoefficientOfVariation End 为 End IF 2020-6-21
'*****************************************************************************
```

'Data 一个计数资料数据

```
Rem 一个计数资料数据
Public Sub Data(ByRef D() As Double)
ReDim D(1 To 140) As Double
D(1) = 177: D(2) = 215: D(3) = 197: D(4) = 97: D(5) = 123
D(6) = 159: D(7) = 245: D(8) = 119: D(9) = 119: D(10) = 131
D(11) = 149: D(12) = 152: D(13) = 167: D(14) = 104
D(15) = 161: D(16) = 214: D(17) = 125: D(18) = 175: D(19) = 219
```

```
D(20) = 118: D(21) = 192: D(22) = 176: D(23) = 175: D(24) = 95
D(25) = 136: D(26) = 199: D(27) = 116: D(28) = 165
D(29) = 214: D(30) = 95: D(31) = 158: D(32) = 83: D(33) = 137
D(34) = 80: D(35) = 138: D(36) = 151: D(37) = 187: D(38) = 126
D(39) = 196: D(40) = 134: D(41) = 206: D(42) = 137:
D(43) = 98: D(44) = 97: D(45) = 129: D(46) = 143: D(47) = 179
D(48) = 174: D(49) = 159: D(50) = 165: D(51) = 136: D(52) = 108
D(53) = 101: D(54) = 141: D(55) = 148: D(56) = 168
D(57) = 163: D(58) = 176: D(59) = 102: D(60) = 194: D(61) = 145
D(62) = 173: D(63) = 75: D(64) = 130: D(65) = 149: D(66) = 150
D(67) = 161: D(68) = 155: D(69) = 111: D(70) = 158
D(71) = 131: D(72) = 189: D(73) = 91: D(74) = 142: D(75) = 140
D(76) = 154: D(77) = 152: D(78) = 163: D(79) = 123: D(80) = 205
D(81) = 149: D(82) = 155: D(83) = 131: D(84) = 209
D(85) = 183: D(86) = 97: D(87) = 119: D(88) = 181: D(89) = 149
D(90) = 187: D(91) = 131: D(92) = 215: D(93) = 111: D(94) = 186
D(95) = 118: D(96) = 150: D(97) = 155: D(98) = 197
D(99) = 116: D(100) = 254: D(101) = 239: D(102) = 160: D(103) = 172
D(104) = 179: D(105) = 151: D(106) = 198: D(107) = 124: D(108) = 179
D(109) = 135: D(110) = 184: D(111) = 168: D(112) = 169
D(113) = 173: D(114) = 181: D(115) = 188: D(116) = 211: D(117) = 197
D(118) = 175: D(119) = 122: D(120) = 151: D(121) = 171: D(122) = 166
D(123) = 175: D(124) = 143: D(125) = 190: D(126) = 213
D(127) = 192: D(128) = 231: D(129) = 163: D(130) = 159: D(131) = 158
D(132) = 159: D(133) = 177: D(134) = 147: D(135) = 194: D(136) = 227
D(137) = 141: D(138) = 169: D(139) = 124: D(140) = 159
End Sub
```

Rem Max 最大值 Range 子程序

Rem 最大值

```
Public Function Max(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(cData()) To UBound(cData())
cData(i) = Data(i)
Next
For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) > cData(i + 1) Then
Max = cData(i)
cData(i + 1) = cData(i)
Else
Max = cData(i + 1)
End If
```

```
End If
Next i
End Function
```

Rem Mix 最小值 Range 子程序

Rem 最小值

```
Public Function Mix(ByRef Data() As Double) As Double
Dim i As Long
Dim cData() As Double
ReDim cData(LBound(Data()) To UBound(Data()))
For i = LBound(Data()) To UBound(Data())
cData(i) = Data(i)
Next
For i = LBound(cData()) To UBound(cData())
If i < UBound(cData()) Then
If cData(i) < cData(i + 1) Then
Mix = cData(i)
cData(i + 1) = cData(i)
Else
Mix = cData(i + 1)
End If
End If
Next i
End Function
```

'Range 全距

Rem 全距

```
Public Function Range(ByRef Data() As Double) As Double
Range = Max(Data) - Mix(Data)
End Function
```

'MeanSquare 样本方差

Rem 样本方差 S^2

Rem Data() =输入数据

Rem ReturnSumOfSquares=返回平方和

Rem ReturnSampleStandardDeviation=返回样本标准差

```
Public Function MeanSquare(ByRef Data() As Double, ByRef ReturnSumOfSquares As Double, _
ByRef ReturnSampleStandardDeviation As Double) As Double
Dim i As Long, n As Long, Sum1 As Double
Dim Average As Double, SumOfSquares As Double
For i = LBound(Data()) To UBound(Data())
```

```
Sum1 = Sum1 + Data(i)
n = n + 1
Next i
Average = Sum1 / n
For i = LBound(Data()) To UBound(Data())
ReturnSumOfSquares = ReturnSumOfSquares + (Data(i) - Average) ^ 2
Next i
If n = 1 Then '防止分母为 0
MsgBox "MeanSquare 出错，不支 1 个样本的情况，退出程序", vbCritical
End
End If
MeanSquare = ReturnSumOfSquares / (n - 1)
ReturnSampleStandardDeviation = Sqr(MeanSquare)
End Function
```

'StandardDeviationMethodOfWeighting 样本标准差加权法

Rem 样本标准差加权法
Rem NumberOfTimes()=分组的次数
Rem ClassMidValue() =分组的组中值
Rem SampleNumber=样本数

```
Public Function StandardDeviationMethodOfWeighting(ByRef NumberOfTimes() As Long, _
ByRef ClassMidValue() As Double, ByVal SampleNumber As Long) As Double
Dim i As Long, l As Long, Part1 As Double, Part2 As Double, n As Long
Dim Part3 As Double
n = SampleNumber
For i = LBound(NumberOfTimes) To UBound(NumberOfTimes)
Part1 = Part1 + NumberOfTimes(i) * ClassMidValue(i) ^ 2
Part2 = Part2 + NumberOfTimes(i) * ClassMidValue(i)
Next i
Part3 = Part1 - Part2 ^ 2 / n
If n = 1 Then
MsgBox "StandardDeviationMethodOfWeighting 出错，不支 1 个样本的情况，退出程序",
vbCritical
End
End If
StandardDeviationMethodOfWeighting = Sqr(Part3 / (n - 1))
Rem 自检程序
Part1 = UBound(NumberOfTimes) - LBound(NumberOfTimes)
Part2 = UBound(ClassMidValue) - LBound(ClassMidValue)
If Part1 <> Part2 Then
MsgBox "次数个数不等于组中值个数", vbCritical
End If
```

```
End Function
```

'CoefficientOfVariation 变异系数

Rem 变异系数
Rem SampleStandardDeviation=样本标准差
Rem AverageNumber=样本平均数
Rem DecimalDigits=小数位数

```
Public Function CoefficientOfVariation(ByVal SampleStandardDeviation As Double, _
ByVal AverageNumber As Double, ByVal DecimalDigits As Long) As String
If AverageNumber = 0 Then
MsgBox "CoefficientOfVariation 出错，不支样本平均数为 0，退出程序", vbCritical
MsgBox "AverageNumber=1E-150 继续运算"
AverageNumber = 1E-150
End If
CoefficientOfVariation = SampleStandardDeviation / AverageNumber
If DecimalDigits = 0 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0%")
ElseIf DecimalDigits = 1 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0.0%")
ElseIf DecimalDigits = 2 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0.00%")
ElseIf DecimalDigits = 3 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0.000%")
ElseIf DecimalDigits = 4 Then
CoefficientOfVariation = Format(CoefficientOfVariation, "0.0000%")
Else
CoefficientOfVariation = SampleStandardDeviation / AverageNumber
End If
End Function
```

'Squareσ总体方差

Rem 总体方差
Rem Data()输入数据

```
Public Function Squareσ(ByRef Data() As Double) As Double
Dim i As Long, l As Long
Dim Sum As Double, Total As Long, AverageNumber As Double, Part1 As Double
For i = LBound(Data) To UBound(Data)
Sum = Sum + Data(i)
Next i
Total = UBound(Data) - LBound(Data) + 1
```

```
AverageNumber = Sum / Total
For i = LBound(Data) To UBound(Data)
Part1 = Part1 + (Data(i) - AverageNumber) ^ 2
Next i
Squareσ = Part1 / Total
End Function
```

53. ChangeColor(图片像素搜素)

详细说明：

函数或方法名称	作用
ChangeColorW	变色为白色
ChangeColorB	变色为黑色

代码：

```
Option Explicit
Option Base 1
'*********************************************************************************
' Class:          MSZ_ChangeColor
' Filename:    MSZ_ChangeColor.cls
' Author:      zhannghong2002
' Date:          2018 11 24
'*********************************************************************************|
Private Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long)
As Long
```

'ChangeColorW 变色为白色

Rem 变色为白色

```
Sub ChangeColorW()
Dim i As Long, l As Long
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
ReDim Pic_Pixel(0 To FrmMain.Pic_Get.Width - 1, 0 To FrmMain.Pic_Get.Height - 1)
If FrmMain.Pic_Reslt.Width <= FrmMain.Pic_Get.Width And FrmMain.Pic_Reslt.Height <=
FrmMain.Pic_Get.Height Then
For i = 0 To FrmMain.Pic_Reslt.Width - 1
      For l = 0 To FrmMain.Pic_Reslt.Height - 1
                  Pic_Pixel(i, l) = GetPixel(FrmMain.Pic_Get.hdc, i, l)
                  Red = (Pic_Pixel(i, l) And &HFF)
                  Green = (Pic_Pixel(i, l) And &HFF00&) / 256
                  Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
                  If FrmMain.Pic_Choice.BackColor = RGB(Red, Green, Blue) Then
                  FrmMain.Pic_Reslt.PSet (i, l), RGB(255, 255, 255)
                  Else
                  FrmMain.Pic_Reslt.PSet (i, l), RGB(Red, Green, Blue)
```

```
                End If
    Next l
Next i
Else
ReDim Pic_Pixel(0 To FrmMain.Pic_Reslt.Width - 1, 0 To FrmMain.Pic_Reslt.Height - 1)
            For i = 0 To FrmMain.Pic_Reslt.Width - 1
                    For l = 0 To FrmMain.Pic_Reslt.Height - 1
                                    Pic_Pixel(i, l) = GetPixel(FrmMain.Pic_Reslt.hdc, i, l)
                                    Red = (Pic_Pixel(i, l) And &HFF)
                                    Green = (Pic_Pixel(i, l) And &HFF00&) / 256
                                    Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL
数据转化成 RGB
                                If RGB(0, 0, 0) = RGB(Red, Green, Blue) Then
                                FrmMain.Pic_Reslt.PSet (i, l), RGB(255, 255, 255)
                                Else
                                FrmMain.Pic_Reslt.PSet (i, l), RGB(Red, Green, Blue)
                                End If
                    Next l
          Next i
End If
End Sub
```

'ChangeColorB 变色为黑色

Rem 变色为黑色

```
Sub ChangeColorB()
Dim i As Long, l As Long
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
ReDim Pic_Pixel(0 To FrmMain.Pic_Get.Width - 1, 0 To FrmMain.Pic_Get.Height - 1)

If FrmMain.Pic_Reslt.Width <= FrmMain.Pic_Get.Width And FrmMain.Pic_Reslt.Height <=
FrmMain.Pic_Get.Height Then
          For i = 0 To FrmMain.Pic_Reslt.Width - 1
                    For l = 0 To FrmMain.Pic_Reslt.Height - 1
                                    Pic_Pixel(i, l) = GetPixel(FrmMain.Pic_Get.hdc, i, l)
                                    Red = (Pic_Pixel(i, l) And &HFF)
                                    Green = (Pic_Pixel(i, l) And &HFF00&) / 256
                                    Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL
数据转化成 RGB
                                If FrmMain.Pic_Choice.BackColor = RGB(Red, Green, Blue) Then
                                FrmMain.Pic_Reslt.PSet (i, l), RGB(0, 0, 0)
                                Else
```

```
                    FrmMain.Pic_Reslt.PSet (i, l), RGB(Red, Green, Blue)
                    End If
                Next l
        Next i
Else
ReDim Pic_Pixel(0 To FrmMain.Pic_Reslt.Width - 1, 0 To FrmMain.Pic_Reslt.Height - 1)
          For i = 0 To FrmMain.Pic_Reslt.Width - 1
                For l = 0 To FrmMain.Pic_Reslt.Height - 1
                                Pic_Pixel(i, l) = GetPixel(FrmMain.Pic_Reslt.hdc, i, l)
                                Red = (Pic_Pixel(i, l) And &HFF)
                                Green = (Pic_Pixel(i, l) And &HFF00&) / 256
                                Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL
数据转化成 RGB
                    If RGB(255, 255, 255) = RGB(Red, Green, Blue) Then
                    FrmMain.Pic_Reslt.PSet (i, l), RGB(0, 0, 0)
                    Else
                    FrmMain.Pic_Reslt.PSet (i, l), RGB(Red, Green, Blue)
                    End If
                Next l
        Next i
End If

End Sub
```

54.GetColor(图片像素搜素)

详细说明：

函数或方法名称	作用
GetColor_2	获得 FrmMain.Pic_Get 的各个像素显示到 FrmMain.Pic_Reslt
ChangeColorB	变色为黑色
GetColorPixel	获取坐标像素
MoveToCenter	移动到中间 GetColorBlock 子程序
GetColorBlock	获得选取图片块
AddPlace	增加显示空间
GetColorBlockFlip	翻转显示图像块

代码：

```
Option Explicit
Option Base 1
'********************************************************************************
' Class:      MSZ_GetColor
' Filename: MSZ_GetColor.cls
' Author:      zhannghong2002
' Date:          2018 11 24
'********************************************************************************
```

```
Private Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long)
As Long
```

'GetColor_2 获得 FrmMain.Pic_Get 的各个像素显示到 FrmMain.Pic_Reslt

Rem 获得 FrmMain.Pic_Get 的各个像素显示到 FrmMain.Pic_Reslt

```
Sub GetColor_2()
Dim i As Long, l As Long
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
ReDim Pic_Pixel(0 To FrmMain.Pic_Get.Width - 1, 0 To FrmMain.Pic_Get.Height - 1)
FrmMain.Pic_Reslt.Width = FrmMain.Pic_Get.Width
FrmMain.Pic_Reslt.Height = FrmMain.Pic_Get.Height
For i = 0 To FrmMain.Pic_Get.Width - 1
    For l = 0 To FrmMain.Pic_Get.Height - 1
```

```
            Pic_Pixel(i, l) = GetPixel(FrmMain.Pic_Get.hdc, i, l)
            Red = (Pic_Pixel(i, l) And &HFF)
            Green = (Pic_Pixel(i, l) And &HFF00&) / 256
            Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
            FrmMain.Pic_Reslt.PSet (i, l), RGB(Red, Green, Blue)
        Next l
Next i
End Sub
```

'GetColorPixel 获取坐标像素

```
Sub GetColorPixel(X, Y)
Dim ChoiceColor As Long
Dim Red As Variant, Green As Variant, Blue As Variant
ChoiceColor = GetPixel(FrmMain.Pic_Get.hdc, X, Y)
Red = (ChoiceColor And &HFF)
Green = (ChoiceColor And &HFF00&) / 256
Blue = (ChoiceColor And &HFF0000) / 65536    'GETPIXEL 数据转化成 RGB
FrmMain.Pic_Choice.BackColor = RGB(Red, Green, Blue)
End Sub
```

Rem MoveToCente 移动到中间 GetColorBlock 子程序

```
Sub MoveToCenter(X, Y, ChoiceBlock)
If ((FillX / FillNumY) > FrmMain.Pic_Block(ChoiceBlock).Width) Then
X = X + ((FillX / FillNumY) - FrmMain.Pic_Block(ChoiceBlock).Width) / 2
End If
If ((FillY / FillNumX) > FrmMain.Pic_Block(ChoiceBlock).Height) Then
Y = Y + ((FillY / FillNumX) - FrmMain.Pic_Block(ChoiceBlock).Height) / 2
End If
End Sub
```

'GetColorBlock 获得选取图片块

```
Sub GetColorBlock(ChoiceBlock)
Dim i As Long, l As Long
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
Dim X As Long, Y As Long
ReDim Pic_Pixel(0 To FrmMain.Pic_Block(ChoiceBlock).Width - 1, 0 To
FrmMain.Pic_Block(ChoiceBlock).Height - 1)
            FillCount = FillCount + 1
            Call AddPlace(X, Y) '行列显示
            Call MoveToCenter(X, Y, ChoiceBlock) '居中显示
```

```
For i = 0 To FrmMain.Pic_Block(ChoiceBlock).Width - 1
    For l = 0 To FrmMain.Pic_Block(ChoiceBlock).Height - 1
        Pic_Pixel(i, l) = GetPixel(FrmMain.Pic_Block(ChoiceBlock).hdc, i, l)
        Red = (Pic_Pixel(i, l) And &HFF)
        Green = (Pic_Pixel(i, l) And &HFF00&) / 256
        Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536   'GETPIXEL 数据转化成 RGB
        FrmMain.Pic_Reslt.PSet (i + X, l + Y), RGB(Red, Green, Blue)
    Next l
Next i
End Sub
```

Rem AddPlace 增加显示空间 GetColorBlock 子程序

```
Sub AddPlace(X, Y)
Rem  列
Dim BlockRow As Long, BlockLine As Long
Dim i As Long
  BlockLine = Int((FillCount - 0.1) / FillNumY) + 1
  BlockRow = FillCount - ((BlockLine - 1) * FillNumY)
X = (BlockRow - 1) * (FillX / FillNumY)
Y = (BlockLine - 1) * (FillY / FillNumX)
End Sub
```

'GetColorBlockFlip 翻转显示图像块

```
Sub GetColorBlockFlip(AddBlock, ChoiceBlock)
Dim i As Long, l As Long
Dim Red As Variant, Green As Variant, Blue As Variant
Dim Pic_Pixel() As Long
ReDim Pic_Pixel(0 To FrmMain.Pic_Block(ChoiceBlock).Width - 1, 0 To
FrmMain.Pic_Block(ChoiceBlock).Height - 1)
For i = 0 To FrmMain.Pic_Block(ChoiceBlock).Width - 1 Step 1
    For l = 0 To FrmMain.Pic_Block(ChoiceBlock).Height - 1 Step 1
        Pic_Pixel(i, l) = GetPixel(FrmMain.Pic_Block(ChoiceBlock).hdc,
FrmMain.Pic_Block(ChoiceBlock).Width - 1 - i, l)
        Red = (Pic_Pixel(i, l) And &HFF)
        Green = (Pic_Pixel(i, l) And &HFF00&) / 256
        Blue = (Pic_Pixel(i, l) And &HFF0000) / 65536   'GETPIXEL 数据转化成 RGB
        FrmMain.Pic_Block(AddBlock).PSet (i, l), RGB(Red, Green, Blue)
    Next l
Next i
End Sub
```