优化Microsoft Office宏的方法

姚卫新（无锡 东华大学计算机科学系 214063）

摘要：本文分析了优化宏的必要性后，通过不同方法的对比，提出了一整套优化宏的方法。

关键词：优化 宏 对象

1 优化的必要性

Visual Basic 是一种灵活性非常大的编程语言，可以编出多种多样的程序来完成相同的任务。刚学习编程或仅仅编写一个只运行一次的宏时，只要能完成工作就行了。但当要编写一个经常使用的宏时—如每周一次整理报表的宏、打开文档时就自动运行的宏、被他人使用的宏，就必须考虑优化，使它们运行速度更快、需要的内存更少。

2 通用的优化策略

下列方法可用来优化 Microsoft Excel、Word、PowerPoint 代码(主要以 Excel 为例，同样的原则适用于 Word、PowerPoint)。

2.1 尽量少用 OLE 参照

调用 Visual Basic 的方法或属性都要通过 OLE 接口，这一过程相当费时间，减少调用中方法或属性的使用是提高宏运行速度的最佳方法。Visual Basic 语句中的各个部分用点号(".")分开，计算属性和方法个数的简便办法是统计点“.”的个数。例如下面的语句中包含三个点。

```
Workbooks(1).Sheets(1).Range("c5").Value=10
```

下面的语句中仅包含一个点

```
ActiveWindow.Left=200
```

2.2 使用对象变量

如果需要经常调用相同的对象，可以为该变量设置一个变量，而后在用到该对象的时候改用变量。例如：下列例子中调用了 Workbooks 和 Sheets 方法各两次。

```
Workbooks(1).Sheets(1).Range("c5").Value=10
Workbooks(1).Sheets(1).Range("d10").Value=12
```

通过设置变量的方法，只需要调用 Workbooks 和 Sheets 方法一次。如：

```
Set sheet=Workbooks(1).Sheets(1)
sheet.Range("c5").Value=10
sheet.Range("d10").Value=12
```

2.3 使用 With 语句

也可以使用 With 语句来避免重复调用相同的对象，且不需要设置一个中间对象变量，上面的例子可以改为下列形式：

```
With Workbooks(1).Sheets(1)
    .Range("c5").Value=10
    .Range("d10").Value=12
End With
```

2.4 使用 For Each...Next 循环

使用 For Each...Next 循环比使用索引更快，许多情况下，使用 For Each...Next 可以使宏更小并容易阅读和调试。下面的例子速度较慢，因为在循环中，它每次都要把 r.Rows(i)重新设置为变量 thisRow

```
Set r=Worksheets(1).Range("al:a200")
For i=1 To r.Rows.Count
    Set thisRow=r.Rows(i)
    If thisRow.Cells(1,1).Value<0 Then
      thisRow.Font.Color=RGB(255,0,0)
    End If
Next
```

改用 For Each...Next 循环后，运行速度更快，代码更短，因为在 For Each...Next 中对行数和位置进行跟踪：

```
For Each thisRow In Worksheets(1).Range("a1:a200").Rows
    If thisRow.Cells(1,1).Value<0 Then
      thisRow.Font.Color=RGB(255,0,0)
    End If
Next
```

2.5 将属性和方法放在循环外

代码从变量中获得值的速度要比从属性中获得值的速度快。如果代码要在循环中获得属性的值，首先在循环外将属性值赋给一个变量，而后在循环内就使用该变量，这样速度会更快。下面的例子速度较慢，因为它每次都在循环中取得 Value 属性值。

```
For iLoop=2 To 200
    Cells(iLoop,1).Value=Cells(1,1).Value
Next
```

将上例修改成下面的形式，速度将明显变快，因为在循环开始前把属性值赋给变量 cv，在每一次循环中只要处理一个属性值 Cells(iLoop,1).Value，省略了 Cells(1,1).Value。

```
cv=Cells(1,1).Value
For i Loop =2 To 200
    Cells(iLoop,1).Value=cv
Next
```

如果在循环中使用了对象，最好把它放在循环外。下面的例子在循环中每次调用ActiveWorkbook 属性、Sheets 属性、Cells 属性

```
For c=1 To 1000
    ActiveWorkbook.Sheets(1).Cells(c,1)=c
Next
```

使用 With 语句改写后，将 ActiveWorkbook 属性、Sheets 属性移到循环外。同样也可以使用对象变量。

```
With ActiveWorkbook.Sheets(1)
    For c=1 To 1000
      .Cells(c,1)=c
    Next
End With
```

2.6 使用集合索引值

对大多数对象的方法和属性而言，可以通过名字或数值指定一个特定对象，使用对象的索引值通常要快一点。如果使用对象名，Visual Basic 首先要找到与该名对应的索引值，直接使用索引值就免去了这多余的一步。当然通过名字来指定集中的一个对象也有其优点。其一是代码容易阅读和调试。其二是比较安全，因为在运行过程中对象的索引值会发生变化，如菜单的索引值代表菜单在菜单条上的位置，当菜单条上的菜单增加或删除时，索引值就发生了变化。只有在确信索引值不会变化时才能使用本方法。

2.7 减少激活和选择对象的次数

很多时候，代码可以在不激活对象的情况下对它们进行操作。如果学习过使用宏录制器编写 Visual Basic 程序，可能习惯在对对象操作前先激活或选中该对象。实际不必这样做就可以编写出简单且快速的Visual Basic代码。例如将 Sheet5 上从 C1 到 C20 的单元填写随机数(使用 AutoFill 方法)，下面的方法速度很慢。

```
Sheets("Sheet5").Select
Range("C1").Select
ActiveCell.FormulaR1C1="=RAND()"
Selection.AutoFill Destination:=Range("C1:C20"),
Type:=xlFillDefault
Range("C1:C20").Select
```

实际上所有的 Select 方法都不需要，用 With 语句直接对 Worksheet 操作即可，代码如下：

```
With Sheets("Sheet5")
    .Range("C1").FormulaR1C1="RAND()"
   .Range("C1").AutoFill Destination:=.Range("C1:C20"),_
    Tyep:=xlFillDefault
End With
```

必须记住的是，宏录制器仅仅记录你所做的动作，但它不具备优化功能。上面的例子中用 AutoFill 方法填充随机数，但并不是最高效的方法。其实一条语句就可以完成：

```
Sheets("Sheet5").Range("C1:C20").Formula="=RANDO"
```

优化宏录制器生成的代码时，仔细考虑一下宏的功能，有一些在用户接口处的操作被当作方法(如 AutoFill)记录下来，实际上这些方法在代码优化时可以去掉，在 Visual Basic 中有更快的方法来完成相同的操作。

2.8 去掉不必要的宏记录表达式

宏记录器产生低效代码的另一个原因是，在对话框中改变设置时，它无法辨别。因而在关闭对话框时，宏记录器将所有选项设置一遍。例如，选择B2：B14单元，在“格式化单元”对话框中将字体改为粗体，宏记录器产生如下代码：

```
Range("B2:B14").Select
With Selection.Font
```

```
    .Name="Arial"
    .FontStyle="Bold"
    .Size=10
    .Strikethrough=False
    .Superscript=False
    .Subscript=False
    .OutlineFont=False
    .Shadow=False
    .Underline=xlNone
    .ColorIndex=xlAutomatic
End With
```

上述操作可以在不选择单元的情况下用一条代码即可完成：

```
Range("B2:B14").Font.FontStyle="Bold"
```

或

```
Range("B2:B14").Font.Bold=True
```

2.9 减少使用 Variant 变量

使用 Variant 变量比较方便，但 Visual Basic 处理这种类型的数据所花费时间比处理类型明确的变量所花费的时间长。

2.10 使用特定对象类型

对对象及其方法和属性的参照，即可以在编译宏时确定，也可以在宏运行时确定，编译确定法要比运行确定法快得多。

为了排除二义性并保证变量具备确定的类型，必须正确使用对象对照的格式，并包含库名，如：

```
Dim wb As Excel.Workbook
Dim dc As Word.Document,cb As MSForms.CommandButton
```

2.11 使用常数

常数在代码编译时就被保存下来。变量却不断变化，每次宏运行时，Visual Basic 必须花时间取得变量值。常数也使宏易于阅读和维护。如果一个宏中的字符串或数值恒定不变，最好将它们定义为常数。

2.12 关闭屏幕更新功能

对于改变文档外观的宏(如在大范围内每隔一个单元改变颜色、建立大量图形对象)，将屏幕更新功能关闭掉，将运行得更快。在编写和调试宏时，打开屏幕更新，运行时再关闭屏幕更新。

关闭屏幕更新只要将 ScreenUpdating 属性关闭，

```
Application.ScreenUpdating=False
```

当宏运行完成后，将它设置为 True(旧版本的 Excel 自动设置成 True，但 Excel97 及 Word97 无此功能)。

3 仅适用于 Microsoft Excel 的优化方法

3.1 使用 Worksheet 函数

Microsoft Excel 中的 Worksheet 函数对一定范围内的单元进行操作，比完成同样功能的 Visual Basic 宏的速度要快，如下面的代码速度较慢：

```
For Each c In Worksheets(1).Range("Al:A200")
    totVal=totVal+c.Value
Next
```

改用 SUM 函数后速度将大大提高

```
totVal=Application. WorksheetFuntion. Sum(Worksheets(1).
Range("al:a200"))
```

产生累计结果的函数如 PRODUCT，COUNT，COUNTA，COUNTIF 等，它们的运行效果都比自己编写的 Visual Basic 代码好。

3.2 使用具有特定用途的 Visual Basic 方法

有很多具有特定用途的 Visual Basic 方法可以对一定范围内的单元进行简洁有效的操作。像 Worksheet 系列函数，它们完成同样任务的速度要比普通 Visual Basic 代码快。例如，用下列代码改变 AI 至 A200 单元相对较慢。

```
For Each c In Worksheets(1).Range("al:a200").Cells
    If c.Value=4 Then c.Value=4.5
Next
```

但改用 Replace 方法后，就快得多

```
    Worksheets(1).Range("al:a200").Replace"4","4.5"
```

又如，从A1至A500，为单元值是4的单元增加一个椭圆，以下代码速度较慢。

```
For Each c In Worksheets(1).Range("al:a500").Cells
    If c.Value=4 Then
      With Worksheets(1).Ovals.Add(c.Left,c.Top,c.
Width,c.Height)
       .Interior.Pattern=xlNone
       .Border.ColorIndex=5
      End With
    End If
Next
```

改用 Find 和 FindNext 方法后，速度将大为改观

```
With Worksheets(1).Range("al:a500")
    Set c=.Find(4)
    If Not c Is Nothing Then
      firstAddress=c.Address
    Do
```

```
    With Worksheets(1).Ovals.Add(c.Left,c.
Top,_c.Width,c.Height)
        .Interior.Pattern=xlNone
        .Border.ColorIndex=5
    End With
Set c=.FindNext(c)
Loop While Not c Is Nothing And c.Address<>firstAddress
End If
End With
```

4 仅适用于 Microsoft Word 的优化方法

除了本章已经讨论的内容，可以使用下列方法在 Word 中建立更小更快的宏。

4.1 使用范围对象

Range 对象比 Selection 对象快，允许定义和使用多个 Range 对象，它们对用户来说是不可见的。

4.2 Next 和 Previous

尽可能使用 Next 和 Previous 返回到集合中后面或前面的元素上，例如 myRange.Next Unit:=wdWord 比 myRange.Words(10)快

4.3 避免使用 WordBasic 对象

WordBasic 对象的方法比其他 Visual Basic 对象的方法慢，例如，WordBasic.FileOpen 比 Documents.Open 慢。

4.4 执行内置对话框

With 语句在设置单个对象的多个属性时非常有效，设置多个属性的另一个方法是设置内置对话框的属性，然后执行对话框。执行内置对话框要比使用 With 语句速度快，因为内置对话框首先保存各种属性值，然后一次性地设置它们，而 With 语句每一次设置一个属性。以下的例子用 With 语句设置了多个段落格式属性。

```
With Selection.ParagraphFormat
    .Alignment=wdAlignParagraphCenter
    .KeepWIthNext=True
    .LeftIndent=InchesToPoints(0.5)
End With
```

下面的例子与上个例子一样设置相同的属性，但它运行得更快，因为它执行了一个内置对话框。

```
Set dlg=Dialogs(wdDialogFormatParagraph)
dlg.Alignment=wdAlignParagraphCenter
dlg.KeepWithNext=True
dlg.LeftIndent="0.5"
dlg.Execute
```

■

参考文献

1 [美] Foxall J.D. 著，王建华译，Visual Basic 编程标准，机械工业出版社，2000.6

2 云舟工作室，中文Access 2000 VBA 一册通，人民邮电出版社，2000.1