

vb代码:如何编写高质量的VB代码(1)

疯狂代码 <http://CrazyCoder.cn/> j:<http://CrazyCoder.cn/VisualBasic/Article21038.html>

什么是一个高效的软件？一个高效的软件不仅应该比实现同样功能的软件运行得更快，还应该消耗更少的系统资源。这篇文章汇集了作者在使用VB进行软件开发时积累下来的一些经验，通过一些简单的例子来向你展示如何写出高效的VB代码。其中包含了一些可能对VB程序员非常有帮助的技术。在开始之前，先让我陈清几个概念。

让代码一次成型：在我接触到的程序员中，有很多人喜欢先根据功能需求把代码写出来，然后在此基础上优化代码。最后发现为了达到优化的目的，他们不得不把代码再重新写一遍。所以我建议你在编写代码之前就需要考虑优化问题。

把握好优化的结果和需要花费的工作之间的关系：通常当完成了一段代码，你需要检查和修改它。在检查代码的过程中，也许你会发现某些循环中的代码效率还可以得到进一步的改进。在这种情况下，很多追求完美的程序员也许会立马修改代码。我的建议是，如果修改这段代码会使程序的运行时间缩短一秒，你可以修改它。如果只能带来10毫秒的性能改进，则不做任何改动。这是因为重写一段代码必定会引入新的错误，而调试新的代码必定会花掉你一定的时间。程序员应该在软件性能和开发软件需要的工作量之间找一个平衡点，而且10毫秒对于用户来说也是一个不能体会到的差异。

在需要使用面向对象方法的时候尽量使用它；VB提供的机制不完全支持面向对象的设计和编码，但是VB提供了简单的类。大多数人认为使用对象将导致代码的效率降低。对于这一点我个人有些不同的意见；考察代码的效率不能纯粹从运行速度的角度出发，软件占用的资源也是需要考虑的因素之一。使用类可以帮助你整体上提升软件的性能，这一点我会在后面的例子中详细说明。

当你编写VB代码的时候，希望你能把上面几点作为指导你编码的原则。我把文章分为两个部分：如何提高代码的运行速度和编译优化。

如何提高代码的运行速度

下面的这些方法可以帮助你提高代码的运行速度：

1. 使用整数 (Integer) 和长整数 (Long)

提高代码运行速度最简单的方法莫过于使用正确的数据类型了。也许你不相信，但是正确地选择数据类型可以大幅度提升代码的性能。在大多数情况下，程序员可以将Single，Double和Currency类型的变量替换为Integer或Long类型的变量，因为VB处理Integer和Long的能力远远高于处理其它几种数据类型。

在大多数情况下，程序员选择使用Single或Double的原因是因为它们能够保存小数。但是小数也可以保存在Integer类型的变量中。例如程序中约定有三位小数，那么只需要将保存在Integer变量中的数值除以1000就可以得到结果。根据我的经验，使用Integer和Long替代Single，Double和Currency后，代码的运行速度可以提高将近10倍。

2. 避免使用变体

对于一个VB程序员来说，这是再明显不过的事情了。变体类型的变量需要16个字节的空間来保存数据，而一个整数 (Integer) 只需要2个字节。通常使用变体类型的目的是为了减少设计的工作量和代码量，也有的程序员图个省事而使用它。但是如果一个软件经过了严格设计和按照规范编码的话，完全可以避免使用变体类型。

在这里顺带提一句，对于Object对象也存在同样的问题。请看下面的代码：

```
Dim FSO
Set FSO = New Scripting.FileSystemObject
    或
```

```
Dim FSO as object
Set FSO = New Scripting.FileSystemObject
```

上面的代码由于在申明的时候没有指定数据类型，在赋值时将浪费内存和CPU时间。正确的代码应该象下面这样：

```
Dim FSO as New FileSystemObject
```

3. 尽量避免使用属性

在平时的代码中，最常见的比较低效的代码就是在可以使用变量的情况下，反复使用属性（Property），尤其是在循环中。要知道存取变量的速度是存取属性的速度的20倍左右。下面这段代码是很多程序员在程序中会使用到的：

```
Dim intCon as Integer
For intCon = 0 to Ubound(SomVar())
Text1.Text = Text1.Text & vbCrLf & SomeVar(intCon)
Next intCon
```

下面这段代码的执行速度是上面代码的20倍。

```
Dim intCon as Integer
Dim sOutput as String
For intCon = 0 to Ubound(SomeVar())
sOutput = sOutput & vbCrLf &
SomeVar(intCon)
Next
Text1.Text = sOutput
```

4. 尽量使用数组，避免使用集合

除非你必须使用集合（Collection），否则你应该尽量使用数组。据测试，数组的存取速度可以达到集合的100倍。这个数字听起来有点骇人听闻，但是如果你考虑到集合是一个对象，你就会明白为什么差异会这么大。

5. 展开小的循环体

在编码的时候，有可能遇到这种情况：一个循环体只会循环2到3次，而且循环体由几行代码组成。在这种情况下，你可以把循环展开。原因是循环会占用额外的CPU时间。但是如果循环比较复杂，你就没有必要这样做了。

6. 避免使用很短的函数

和使用小的循环体相同，调用只有几行代码的函数也是不经济的--调用函数所花费的时间或许比执行函数中的代码需要更长的时间。在这种情况下，你可以把函数中的代码拷贝到原来调用函数的地方。

7. 减少对子对象的引用

在VB中，通过使用.来实现对象的引用。例如：

```
Form1.Text1.Text
```

在上面的例子中，程序引用了两个对象：Form1和Text1。利用这种方法引用效率很低。但遗憾的是，没有办法可以避免它。程序员唯一可以做就是使用With或者将用另一个对象保存子对象（Text1）。

注释：使用With

```
With frmMain.Text1
.Text = "Learn VB"
.Alignment = 0
.Tag = "Its my life"
.BackColor = vbBlack
.ForeColor = vbWhite
End With
```

或者

注释：使用另一个对象保存子对象

```
Dim txtTextBox as TextBox
Set txtTextBox = frmMain.Text1
TxtTextBox.Text = "Learn VB"
TxtTextBox.Alignment = 0
TxtTextBox.Tag = "Its my life"
TxtTextBox.BackColor = vbBlack
TxtTextBox.ForeColor = vbWhite
```

注意，上面提到的方法只适用于需要对一个对象的子对象进行操作的时候，下面这段代码是不正确的：

```
With Text1
.Text = "Learn VB"
.Alignment = 0
.Tag = "Its my life"
.BackColor = vbBlack
.ForeColor = vbWhite
End With
```

很不幸的是，我们常常可以在实际的代码中发现类似于上面的代码。这样做只会使代码的执行速度更慢。原因是With块编译后会形成一个分枝，会增加了额外的处理工作。

8. 检查字符串是否为空

大多数程序员在检查字符串是否为空时会使用下面的方法：

```
If Text1.Text = "" then
```

注释：执行操作

```
End if
```

很不幸，进行字符串比较需要的处理量甚至比读取属性还要大。因此我建议大家使用下面的方法：

```
If Len(Text1.Text) = 0 then
```

注释：执行操作

```
End if
```

9. 去除Next关键字后的变量名

在Next关键字后加上变量名会导致代码的效率下降。我也不知道为什么会这样，只是一个经验而已。不过我想很少有程序员会这样画蛇添足，毕竟大多数程序员都是惜字如金的人。

注释：错误的代码

```
For iCount = 1 to 10
```

注释：执行操作

```
Next iCount
```

注释：正确的代码

```
For iCount = 1 to 10
```

注释：执行操作

```
Next
```

10. 使用数组，而不是多个变量

当你有多个保存类似数据的变量时,可以考虑将他们用一个数组代替。在VB中，数组是最高效的数据结构之一。

11. 使用动态数组，而不是静态数组

使用动态数组对代码的执行速度不会产生太大的影响，但是在某些情况下可以节约大量的资源。

2008-10-19 1:41:49

疯狂代码 <http://CrazyCoder.cn/>