全国计算机等级考试二级：如何编写高质量的VB代码 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/166/2021_2022__E5_85_A8_E5_9B_BD_E8_AE_A1_E7_c97_166168.htm 计算机等级考试训练软件《百宝箱》 简介： 本文描述了如何通过一些技术手段来提高VB代码的执行效率。这些手段可以分为两个大的部分：编码技术和编译优化技术。在编码技术中介绍了如何通过使用高效的数据类型、减少外部引用等编程手段来提高代码执行速度，减少代码消耗的系统资源。在编译优化技术中介绍了如何正确地利用VB提供的编译选项对在编译时最后生成的可执行文件进行优化。 前言 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。使用类可以帮助你在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个字节的空间来保存数据，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 amp. 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 amp. 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）。 使用With With frmMain.Text1 .Text = "Learn VB" .Alignment = 0 .Tag = "Its my life" .BackColor = vbBlack .ForeColor = vbWhite End With 或者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 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 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。 错误的代码 For iCount = 1 to 10 执行操作 Next iCount 正确的代码 For iCount = 1 to 10 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。 12. 销毁对象 无论编写的是什么软件，程序员都需要考虑在用户决定终止软件运行后释放软件占用的内存空间。但遗憾的是很多程序员对这一点好像并不是很在意。正确的做法是在退出程序前需要销毁程序中使用的对象。例如： Dim FSO as New FileSystemObject 执行操作 销毁对象 Set FSO = Nothing 对于窗体，可以进行卸载： Unload frmMain 或 Set frmMain = Nothing 13. 变长和定长字符串 从技术上来说，与变长字符串相比，定长字符串需要较少的处理时间和空间。但是定长字符串的缺点在于在很多情况下，你都需要调用Trim函数以去除字符串末的空字符，这样反而会降低代码效率。所以除非是字符串的长度不会变化，否则还是使用变长字符串。 14. 使用类模块，而不是ActiveX控件 除非ActiveX控件涉及到用户界面，否则尽量使用轻量的对象，例如类。这两者之间的效率有很大差异。 15. 使用内部对象 在涉及到使

用ActiveX控件和DLL的时候，很多程序员喜欢将它们编译好，然后再加入工程中。我建议你最好不要这样做，因为从VB连接到一个外部对象需要耗费大量的CPU处理能力。每当你调用方法或存取属性的时候，都会浪费大量的系统资源。如果你有ActiveX控件或DLL的源代码，将它们作为工程的私有对象。 16. 减少模块的数量 有些人喜欢将通用的函数保存在模块中，对于这一点我表示赞同。但是在一个模块中只写上二三十行代码就有些可笑了。如果你不是非常需要模块，尽量不要使用它。这样做的原因是因为只有在模块中的函数或变量被调用时，VB才将模块加载到内存中；当VB应用程序退出时，才会从内存中卸载这些模块。如果代码中只有一个模块，VB就只会进行一次加载操作，这样代码的效率就得到了提高；反之如果代码中有多个模块，VB会进行多次加载操作，代码的效率会降低。 17. 使用对象数组 当设计用户界面时，对于同样类型的控件，程序员应该尽量使用对象数组。你可以做一个实验：在窗口上添加100个PictureBox，每个PictureBox都有不同的名称，运行程序。然后创建一个新的工程，同样在窗口上添加100个PictureBox，不过这一次使用对象数组，运行程序，你可以注意到两个程序加载时间上的差别。 18. 使用Move方法 在改变对象的位置时，有些程序员喜欢使用Width，Height，Top和Left属性。例如：
Image1.Width = 100 Image1.Height = 100 Image1.Top = 0 Image1.Left = 0 实际上这样做效率很低，因为程序修改了四个属性，而且每次修改之后，窗口都会被重绘。正确的做法是使用Move方法： Image1.Move 0,0,100,100 19. 减少图片的使用 图片将占用大量内存，而且处理图片也需要占用很多CPU资

源。在软件中，如果可能的话，可以考虑用背景色来替代图片--当然这只是从技术人员的角度出发看这个问题。 20. 使用ActiveX DLL，而不是ActiveX控件 如果你设计的ActiveX对象不涉及到用户界面，使用ActiveX DLL。 编译优化 我所见过的很多VB程序员从来没有使用过编译选项，也没有试图搞清楚各个选项之间的差别。下面让我们来看一下各个选项的具体含义。 1. P-代码（伪代码）和本机代码 你可以选择将软件编译为P-代码或是本机代码。缺省选项是本机代码。那什么是P-代码和本机代码呢？ P-代码：当在VB中执行代码时，VB首先是将代码编译为P-代码，然后再解释执行编译好的P-代码。在编译环境下，使用这种代码要比本机代码快。选择P-代码后，编译时VB将伪代码放入一个EXE文件中。 本机代码：本机代码是VB6以后才推出的选项。当编译为EXE文件后，本机代码的执行速度比P-代码快。选择本机代码后，编译时VB使用机器指令生成EXE文件。 在使用本机代码进行编译时，我发现有时候会引入一些莫名其妙的错误。在编译环境中我的代码完全正确地被执行了，但是用本机代码选项生成的EXE文件却不能正确执行。通常这种情况是在卸载窗口或弹出打印窗口时发生的。我通过在代码中加入DoEvent语句解决了这个问题。当然出现这种情况的几率非常少，也许有些VB程序员从来没有遇到过，但是它的确存在。 在本机代码中还有几个选项： a) 代码速度优化：该选项可以编译出速度较快的执行文件，但执行文件比较大。推荐使用 b) 代码大小优化：该选项可以编译出比较小的执行文件，但是以牺牲速度为代价的，不推荐使用。 c) 无优化：该选项只是将P-代码转化为本机代码，没有做任何优化。在调试代码时可以使用

。 d) 针对Pentium Pro优化：虽然该项不是本机代码中的缺省选项，但是我通常会使用该选项。该选项编译出的可执行程序在Pentium Pro和Pentium 2以上的机器上可以运行得更快，而在比较老的机器上要稍稍慢一些。考虑到现在用Pentium 2都是落伍，所以推荐大家使用该选项。 e) 产生符号化调试信息：该项在编译过程中生成一些调试信息，使用户可以利用Visual C 一类的工具来调试编译好的代码。使用该选项会生成一个.pdf文件，该文件记录了可执行文件中的标志信息。当程序拥有API函数或DLL调用时，该选项还是比较有帮助的。 2. 高级优化 高级优化中的设置可以帮助你提高软件的速度，但是有时候也会引入一些错误，因此我建议大家尽量小心地使用它们。如果在代码中有比较大的循环体或者复杂的数学运算时，选中高级优化中的某些项会大幅度提升代码的性能。如果你使用了高级优化功能，我建议你严格测试编译好的文件。 a) 假定无别名：可以提高循环体中代码的执行效率，但是在如果通过变量的引用改变变量值的情况下，例如调用一个方法，变量的引用作为方法的参数，在方法中改变了变量的值的话，就会引发错误。有可能只是返回的结果错误，也有可能是导致程序中断运行的严重错误。 b) 取消数组绑定检查、取消整数溢出检查和取消浮点错误检查：在程序运行时，如果通过这些检查发现了错误，错误处理代码会处理这些错误。但是如果取消了这些检查，发生了错误程序就无法处理。只有当你确定你的代码中不会出现上面的这些错误时，你才可以使用这些选项。它们将使软件的性能得到很大的提升。 c) 允许不舍入的浮点操作：选择该选项可以是编译出来的程序更快地处理浮点操作。它唯一的缺点就是在比较两

个浮点数时可能会导致不正确的结果。 d) 取消Pentium FDIV安全检查：该选项是针对一些老的Pentium芯片设置的，现在看来已经过时了。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com