VB 减小程序代码的几种技术

李滋堤　杨应成　唐大全　(海军航空工程学院　264001)

摘要:减小应用程序的大小是一个程序员必须经常面临和考虑的问题,在 Visual Basic 有许多技术可用于编制更紧凑的代码。本文介绍的几种方法除了缩小应用程序所占内存的大小外,大部分优化方法还缩小了 .exe 文件的大小。

关键词:Visual Basic　内存　代码

在优化程序代码大小的诸多技术中,大多包括从代码中删除不必要的元素。在编译应用程序时,Visual Basic 自动删除某些元素。而标识符名称、注释、空行的长度或数量是无须限制的,当应用程序作为一个 .EXE 文件运行时,以上这些元素都不会影响应用程序所占内存的大小。

其他元素,如变量、窗体和过程,确实要占据内存的一部分空间。最好将它们精简以使效率更高。下面几项技术可用于缩小应用程序所需内存和减少代码大小:

1. **减少加载窗体、控件数目和用标签代替文本框**

每一个加载的窗体,无论可视与否,都要占据一定数量的内存(其数量随窗体上控件的类型和数量,以及窗体上位图的大小等的不同而变化)。只在需要显示时才加载窗体,不再需要时卸载窗体(而不是隐藏窗体)。记住,任何对窗体的属性、方法或控件的引用,或对用 New 声明的窗体变量的引用,都会导致 Visual Basic 加载该窗体。

当使用 Unload 方法卸载窗体时,只能释放部分窗体所占空间。要释放所有空间,可用关键字 Nothing 使窗体的引用无效:

```
Set Form = Nothing
```

当设计应用程序时,窗体应尽量少用控件。实际的限制取决于控件的类型和系统,但实际上,含有大量控件的窗体将运行缓慢。一项与之相关的技术是:设计时尽可能地使用控件数组,而不是在窗体上放置大量同类型的控件。控件数组是一组具有共同名称和类型的控件。它们的事件过程也相同。在设计时,使用控件数组添加控件所消耗的资源比直接向窗体添加多个相同类型的控件消耗的资源要少。当希望若干控件共享代码时,控件数组也很有用。标签控件 Label 占用的 Windows 资源比文本框 Textbox 少,因此在可能的情况下,应使用标签代替文本框。例如,当窗体上需要一个隐藏的控件保存文本时,使用标签更有效。

2. **使用磁盘文件或资源和组织模块**

在设计时,直接放入应用程序的数据(象属性或代码中的文字字符串和数值)将增加运行时应用程序占用的内存。运行时从磁盘文件或资源中加载数据可减少占用内存。这对大位图和字符串特别有价值。资源文件实际上是由一系列独立的字符串、位图或者其他项目组成的,其中每一项都有一个唯一的标识符。可以使用类似于在 Microsoft Visual C++(R) 中提供的文本编辑器和资源编译器创建资源文件。编译过的资源文件带有 .res 扩展名。

Visual Basic 只在需要时才加载模块即当代码调用模块中的一个过程时,模块才被加载到内存。如果从未调用一特定模块中的过程,Visual Basic 决不加载该模块。因此,尽量把相关的过程放在同一模块中,让 Visual Basic 只在需要时才加载模块。

3. **考虑替换 Variant 数据类型**

Variant 数据类型使用极其灵活,但是比其他数据类型所占内存大。当要压缩应用程序多余的空间时,应考虑用其他数据类型替代 Variant 变量,特别是替代 Variant 变量数组。

每一个 Variant 占用 16 个字节,而 Integer 占 2 个字节,Double 占 8 个字节。变长字符串变量占用 4 个字节加上字符串中每一个字符占用 1 个字节,但是,每一个包含字符串的 Variant 都要占用 16 个字节加上字符串中每一个字符占用 1 个字节。因为它们太大,因此在用作局部变量或过程的参数时,Variant 变量是特别烦人的,这是因为它们消耗堆栈空间太快。

但在有些情况下,使用其他数据类型替代 Variant,灵活性降低了,为弥补损失的灵活性,不得不增加更多的

代码。结果是大小没有真正的减小。

4. 使用动态数组并在删除时回收内存

使用动态数组代替固定数组。当不再需要动态数组的数据时，用 Erase 或 ReDim Preserve 放弃不需要的数据，并回收数组所用内存。例如，用以下代码可回收动态数组所用空间：

```
Erase MyArray
```

这里，Erase 完全删除数组，ReDim Preserve 则只缩短数组而不丢失其内容：

```
ReDim Preserve MyArray(10, smallernum)
```

删除了固定大小数组，也不能回收该数组所占空间——只是简单地清除数组每一元素中的值。如果元素是字符串，或包含字符串或数组的 Variant 变量，那么删除数组可回收这些字符串或 Variants 所占内存，而不是数组本身所占内存。

5. 回收被字符串或对象变量用过的空间

当过程结束时，可自动回收（非静态）局部字符串和数组变量所用空间。但是，全局和模块级的字符串和数组变量一直存活到整个程序结束。要想应用程序尽量小，就得尽可能回收这些变量所用空间。将零长度字符串赋给字符串变量，可回收其空间：

```
SomeStringVar = "" '回收空间。
```

同样，将对象变量设置成 Nothing 可回收该对象所用的部分（而不是全部）空间。例如，删除一个 Form 对象变量：

```
Global F As New StatusForm
F.Show 1              'Form 加载并以模态显示。
X = F.Text1.Text      '用户按下按钮
                      '隐藏窗体。
Unload F              '删除窗体可视部分。
Set F = Nothing       '回收空间（模块数据）。
```

即使没有使用显式窗体变量，也应注意将不再用的窗体卸载，而不是简单地隐藏。

6. 消除死代码和无用的变量

在开发和修改应用程序时，可能遗留了死代码——代码中的一个完整过程，而它并没有被任何地方调用。也可能声明了一些不用的变量。虽然在创建 .exe 文件中，Visual Basic 确实可删除无用的常数，但不能删除无用的变量和死代码。注意要复查代码，查找并删除无用的变量和死代码。如 Debug.Print 语句，在运行 .exe 时被忽略，可它常常出现在 .exe 文件中。

当创建 .exe 文件时，含有字符串和变量作为参数的 Debug.Print 语句不会被编译。但对于含有函数作为参数的 Debug.Print 语句，其本身被编译器忽略，而函数则被编译。因此，在应用程序运行时，函数被调用，但返回值被忽略。因为在 .exe 文件中，函数作为 Debug.Print 的参数出现时，将占用空间和 CPU 周期时间，所以在生成 .exe 文件前，最好删除这些语句。

在“编辑”菜单中使用“查找”命令搜索特定变量的引用。或者当每个模块都含有 Option Explicit 语句时，通过删除或注释该变量的声明，并运行应用程序，可迅速发现变量是否被使用。若该变量被使用，则 Visual Basic 将出错。若不出错，则该变量没被使用。

参考文献

[1] [美] Richard Mansfield 著（1996）廖卫东 赵军 译 Visual Basic 4.0 编程手册 机械工业出版社

[2] [美] Mike McKelvy & Ronald Martinsen 著（1997）杨继平 欧梅 译 Visual Basic 5 开发使用手册 机械工业出版社

（来稿时间：1998 年 7 月）