

1 Visual Basic

Visual Basic 应用软件开发规范的目的，是使应用程序的结构和编码风格标准化，以便于阅读和理解，使代码更健壮，同时易于后期维护。

本开发规范主要规定了编写 Visual Basic 应用程序时，在命名约定、代码格式、代码注释、错误处理和界面设计方面的规范。这些规范不是一成不变的，我们也不可能样样都要标准化，但保持原则上的代码结构清晰，便于自己和他人的阅读理解，就足够了。

1.1 命名约定

1、变量命名约定

(1) 变量应该被定义在尽可能小的范围内。全局 (Public) 变量可以导致极其复杂的状态，并且使一个应用程序的逻辑非常难于理解。全局变量也使代码的重用和维护更加困难。

| 范围 | 声明位置 | 可见位置(引用位置) |
|-----|---------------------------------------|----------------|
| 过程级 | 过程、子过程或函数过程中的 Private、Dim | 在声明它的过程中 |
| 模块级 | 窗体或代码模块 (frm、bas) 的声明部分中的 Private、Dim | 窗体或代码模块中的每一个过程 |
| 全局 | 代码模块 (bas) 的声明部分中的 Public | 应用程序中的每一处 |

(2) 随着工程大小的增长，划分变量范围的工作也迅速增加。在类型前缀的前面放置单字母范围前缀表明了这种增长，但变量名的长度并没有增加很多。如果一个变量在标准模块或窗体模块中被声明为 Public，那么该变量具有全局范围。如果一个变量在标准模块或窗体模块中被分别声明为 Private，那么该变量有模块级范围。此用法一般用在大型程序中。

| 范围 | 前缀 | 示例 |
|-----|----|--------------|
| 过程级 | 无 | dblPayee |
| 模块级 | m | mInProceedID |
| 全局 | g | gInRollID |

(3) 变量数据类型，其前缀由三个字母组成，均需小写。而且前缀可以被扩展，用来指明变量范围。变量命名一律使用前缀+用途的命名方式。

| 变量类型 | 前缀 | 描述 | 示例 |
|----------|-----|------|---------------|
| Array | arr | 数组 | arrDyna |
| Boolean | bln | 布尔值 | blnFound |
| Byte | byt | 字节型 | bytRasterData |
| Currency | cur | 货币型 | curProduct |
| Date | dat | 日期型 | datStartRun |
| Double | dbl | 双精度 | dblPayee |
| Error | err | 错误对象 | errLoadData |
| Handle | hnd | 句柄 | hndForm |
| Integer | int | 整型 | intCounter |
| Long | lng | 长整型 | lngDistance |
| Object | obj | 对象 | objCurrent |
| Single | sng | 单精度 | sngAverage |
| String | str | 字符型 | strFirstName |
| Time | tim | 时间 | timEnd |



WWW.MUGUA.NET

网址 www.mugua.net 《Visual Basic 6.0 完全自学手册》热销中

2、控件命名约定

控件命名约定与变量命名相似，一律使用前缀+用途的命名方式。

| 控件类型 | 前缀 | 描述 |
|-----------------------------|-----|-------|
| Combobox | cmb | 下拉列表框 |
| Checkbox | chk | 检查框 |
| Commandbutton | cmd | 命令按钮 |
| Commdialog | dlg | 公共对话框 |
| DBGrid/MSHFlexGrid/DataGrid | grd | 网格 |
| Frame | Fra | 框架 |
| Form | frm | 窗体 |
| Image | img | 图像 |
| Label | lab | 标签 |
| Line | lin | 直线 |
| Listbox | lst | 列表框 |
| ListView | lv | 列表视图 |
| MaskedTextBox | msk | 掩码编辑框 |
| Menu | mnu | 菜单 |
| Optionbutton | opt | 选项框 |
| Picture | pic | 图片框 |
| Report | rpt | 报表 |
| Scrollbar | sbr | 滚动条 |
| Shape | shp | 图形 |
| StatusBar | st | 状态条 |
| Timer | tmr | 定时器 |
| ToolBar | tb | 工具条 |
| Textbox | txt | 文本框 |
| TreeView | tv | 树型视图 |

说明：对于上面没有列出的控件，应该用唯一的由两个或三个字符组成的前缀使它们标准化，以保持一致性。只有当需要澄清时，才使用多于三个字符的前缀。例如，对于派生的或定制的控件象上述那样扩展其前缀，使得在真正使用了哪一个控件的问题上避免混淆。

3、数据库对象命名约定

数据库对象命名约定与变量命名相似，一律使用前缀+用途的命名方式。

| 数据库对象 | 前缀 | 示例 |
|-----------|-----|------------------|
| Container | con | conReports |
| Database | db | dbAccounts |
| DBEngine | dbe | dbeJet |
| Document | doc | docSalesReport |
| Field | fld | fldAddress |
| Group | grp | grpFinance |
| Index | ix | idxAge |
| Parameter | prm | prmJobCode |
| QueryDef | qry | qrySalesByRegion |
| Recordset | rec | recForecast |
| Relation | rel | relEmployeeDept |
| TableDef | tbd | tbdCustomers |
| User | usr | usrNew |
| Workspace | wsp | wspMine |

4、常量命名约定



WWW.MUGUA.NET

网址 www.mugua.net 《Visual Basic 6.0 完全自学手册》热销中

常量名建议全部大写，使用下划线作为单词间的分隔符，单词尽量使用全名称。

对于常量名，应遵循与变量相同的约定，并加上 Const 关键字。如果是全局常量，应该使用 Public 而不是早期版本的 Global 来声明变量，对一些常用词应该使用简写。

```
Public Const SC_CLOSE = &HF060
Public Const SC_MINIMIZE = &HF020
Public Const SC_MAXIMIZE = &HF030
```

5、函数命名约定

此处函数包括 Sub 和 Function，以下将这两种过程统称为函数。

函数表示的是一个动作，所以它的结构应该是动词+名词，动词必须小写，后面的名称首字母大写，如：

```
getRollInfoState
loadProceedPayeeData
setUserPass
```

函数命名尽量不要使用缩写，而且它的名称应该使人一目了然，必须能够反映函数的作用，能够从名称就知道这个函数的功能，不要使用无意义的函数名称，如：getCode，readData。

另外，函数命名必须能准确表达所要实现的功能，不至于引起歧义。当函数名称不足以表达其功能时，使用在函数头部加上让调用者足够明白的注释。

参数的命名：参数命名的原则是全部小写，如果参数包括两个或以上的单词时，首单词字母小写，其它单词首字母大写，如 showRows、isUpdate。

1.2 代码格式

1、代码格式目的

对代码进行格式化时，所要达到的目的：

- 通过将代码分割成功能块和便于理解的代码段，使代码更容易阅读和理解。
- 减少为理解代码结构而需要做的工作。
- 使代码的读者不必进行假设。
- 使代码结构尽可能做到格式清楚了。

2、代码格式约定

(1) 不要将多个语句放在同一行上

当多个语句出现在同一个代码行上，各个语句之间用冒号分开时，就很难确定一个语句在何处结束，另一个语句在何处开始，这无形之中就增加了代码的复杂性。

下面这段代码给人一个冗长的感觉

```
intXLeg = m_rectBound.Right - m_rectBound.Left: intYLeg = _
m_rectBound.Bottom - m_rectBound.Top
```

应该写成下面这个格式

```
intXLeg = m_rectBound.Right - m_rectBound.Left
intYLeg = m_rectBound.Bottom - m_rectBound.Top
```

(2) 使用空白行将相关的语句组织在一起

- 在以下的情况，要插入空白行：



- ❑ 每个 If...Then 构造的前面和后面（尤其是 If 语句前的注释的前面）。
- ❑ 每个 Select Case 构造的前面。
- ❑ 每个循环的前面和后面。
- ❑ 变量块的说明的后面。
- ❑ 执行统一任务的两个语句组的中间。
- ❑ 应该在两个过程或函数之间插入两个空行。

(3) 对很长的代码行使用行接续符

在语句中选定一个适当位置，将行接续符 & 置于该位置，并在行接续符的前面放置一个空格。接续符用于指明下一个代码行是当前代码语句的组成部分。当用行接续符将很长的 SQL 语句分割成多个代码行时，它的好处是很明显的，例如：

```
If Not SqlRst("SELECT rol_proceed.id,proceed AS 公证事项名称 " _
    & "FROM rol_proceed " _
    & "INNER JOIN not_proceed " _
    & "ON proceedid=not_proceed.id " _
    & "WHERE rollid=2006060638") Then Exit Sub
```

(4) SQL 语句格式

代码中书写的 SQL 语句要求 SQL 关键字全部大写，表名和字段名小写。例如：

```
"SELECT name FROM user_info WHERE accept=1"
```

2、代码缩进约定

缩进代码行后，阅读代码的人就能够直观地了解执行统一任务的各个语句的组织结构。当遇到以下情况时，代码要缩进一个 TAB 制表位（在本书一个制表位为 4 个字符）：

(1) 当使用 End If 时，在 If 语句后缩进

```
If intCurRecord > 1 Then
    intCurRecord = intCurRecord - 1
End If
```

(2) 在 Else 语句后缩进

```
If intCurRecord > 1 Then
    intCurRecord = intCurRecord - 1
Else
    MsgBox "已经是第一条记录!", vbInformation
Exit Sub
End If
```

(3) 在 Select Case 语句后缩进

```
Select Case objTool.Name
    Case Is = "Save"
    Case Is = "Exit"
End Select
```

(4) 在 Case 语句后缩进

```
Select Case objTool.Name
    Case Is = "Save"
        SaveData
    Case Is = "Exit"
        Unload Me
```



```
End Select
```

(5) 在 For 语句后缩进

```
For intCount = 1 To intLastRecord
    If intCount <> intTempRecord Then
        Get #intFileNum, intCount, Cust
        Put #2, , Cust
    End If
Next
```

(6) 在 Do 语句后缩进

```
Do While Not Rst.EOF
    cmbNotman.AddItem Rst(0)
    Rst.MoveNext
Loop
```

(7) 在 With 语句后缩进

```
With Gridtask
    .TextMatrix(0, 1) = "案卷号"
    .TextMatrix(0, 2) = "公证类别"
    .TextMatrix(0, 3) = "承办人"
    .TextMatrix(0, 4) = "流程状态"
    .ColWidth(0) = 0
    .ColWidth(1) = 1000
    .ColWidth(2) = 900
    .ColWidth(3) = 700
    .ColWidth(4) = 900
End With
```

(8) 在调用 Recordset 对象的 Edit 或 AddNew 方法后缩进。Update 或 ConcelUpdate 方法的缩进层次应该与 Edit 或 AddNew 语句相同。

```
rstRollInfo.Edit
rstRollInfo(0) = strID
rstRollInfo(1) = strName
rstRollInfo.Update
```

(9) 在调用 BeginTrans 方法后缩进

```
Workspaces(0).BeginTrans
Do While Not rstRollInfo.EOF
    intCounter = intCounter + 1
    rstRollInfo.Edit
    rstRollInfo(0) = strID
    rstRollInfo(1) = strName
    rstRollInfo.Update
    rstRollInfo.MoveNext
Loop
Workspaces(0).CommitTrans
```

(10) 对自定义数据结构的代码进行缩进

```
Private Type Customer          '定义客户资料记录类型
    Customer_ID                As String * 3
    Customer_Name              As String * 10
```



WWW.MUGUA.NET

网址 www.mugua.net 《Visual Basic 6.0 完全自学手册》热销中

```

Customer_Job      As String * 10
Customer_Company As String * 20
End Type

```

(11) 对枚举说明的主体进行缩进

```

Public Enum RollInfo      '定义案卷属性信息枚举
    not_Kind = 0          '类别
    not_Country = 1      '使用地
    not_Purpose = 2        '用途
    not_Language = 3     '语种
    not_Name = 4         '操作员
End Enum

```

1.3 代码注释

1、代码注释目的

- 使用代码注释时，应该达到下列目的：
- 用文字说明代码的作用（即为什么要编写该代码，而不是如何编写）。
- 明确指出该代码的编写思路和逻辑方法。
- 使人们注意到代码中的重要转折点。
- 使代码的阅读者不必在他们的头脑中仿真运行代码的执行过程。

2、代码注释约定

为了增强程序的可读性并达到代码注释的目的，注释应遵循以下约定：

- 注释尽量使用中文。
- 每个程序开始部分必须有以下注释。
- 程序名称、程序描述、实现功能、设计人员、设计日期。
- 程序代码部分必须有足够的代码片段的注释，表达清楚该程序片段的编写意图和实现原理，文字既简炼而准确。
- 应该在编写代码前进行注释。
- 用注释来说明何时可能出错和为什么出错。
- 避免形成注释框，因为这只会增加不必要的额外工作量。
- 一些显而易见、人人都能看得懂的代码，不必要加注释。

3、代码注释示例

下面是代码注释示例，全面反应了变量注释、函数注释、逻辑分支及循环条件的注释。

```

=====
'目的：返回指定用户在 UserList 数组中第一次出现的位置。
'输入：strUserList(): 所查找的用户列表。
'      strTargetUser: 要查找的用户名。
'返回：strTargetUser 在 strUserList 数组中第一次出现时的索引。
'编写：程序员甲
'修改：程序员乙
'时间：2006-06-30
=====

```

木瓜工作室

WWW.MUGUA.NET

网址 www.mugua.net 《Visual Basic 6.0 完全自学手册》热销中

```
Function intFindUser (strUserList(), strTargetUser)
    Dim intCounter As Integer           '循环计数器。
    Dim blnFound As Boolean             '发现目标的标记。
    intFindUser = -1
    intCounter = 0                       '初始化循环计数器。
    '开始循环
    Do While intCounter <= Ubound(strUserList) and Not blnFound
        If strUserList(iCounter) = strTargetUser Then
            blnFound = True             '标记设为 True。
            intFindUser = intCounter    '返回值设为循环计数器。
        End If
        intCounter = intCounter + 1     '循环计数器加 1。
    Loop
    '结束循环
End Function
```

1.4 错误处理

1、错误处理目的

运用错误处理程序要达到的目的是：

- 防止程序崩溃。
- 可能时恰当地纠正错误。
- 发生错误时将情况通知用户，以便纠正错误。

2、错误处理约定

(1) 使用 On Resume Next 以忽略错误

对错误进行处理的最简单（和最危险）的方法是使用 On Error Resume Next 语句。On Error Resume Next 语句规定，代码中的错误将完全被忽略，存在错误的代码行被跳过，然后继续执行下一个语句。例如，下面这个过程存在一个运行期错误（即一个被 0 除的错误），它由 On Error Resume Next 错误处理程序来处理：

```
Private Sub cmdRun_Click()
    On Error Resume Next
    Dim intNum As Integer           '定义整型变量
    Dim lngNum As Long              '定义长整型变量

    lngNum = 60000
    intNum = lngNum                 '将 lngNum 赋值给 int
End Sub
```

2) 使用 On Error GoTo 转移执行的代码流

转移执行的代码用行标注来指定，行标注是一个文本串，用于标识一个单行代码。行标注可以是任何字符的组合，它以一个字母开始，以冒号结尾。

例如，下面这个过程包含一个标记。当出现被 0 除的错误时，On Error GoTo 语句转移到标注为 errPrint 的代码行上去执行：

```
Private Sub cmdRun_Click()
```



WWW.MUGUA.NET

网址 www.mugua.net 《Visual Basic 6.0 完全自学手册》热销中

```

On Error GoTo errHandler
Dim intNum As Integer           '定义整型变量
Dim lngNum As Long             '定义长整型变量

lngNum = 60000
intNum = lngNum                 '将 lngNum 赋值给 int
Exit Sub
errHandler:                     '错误处理段标号
MsgBox Err.Description, vbCritical, "错误信息" '错误处理段
End Sub

```

1.5 界面设计

1、界面设计原则

用户界面设计必须保持一致，界面设计的一致性要达到的目的是：

- 创建程序中统一的界面，并且创建不同应用程序中统一的界面；
- 使得用户能够充分利用他们现有的技巧（即使用可复用的知识）；
- 减少用户在操作中的混乱和困难；
- 形成醒目的界面，建立用户的信心，使用户感到满意。
- 用简明扼要的专业性语言告诉用户他们必须知道的消息。

2、界面设计基础

(1) 为每个窗体赋予相应的边框样式

新建一个窗体时，Visual Basic 自动将窗体的 BorderStyle 属性设置为 Sizable（可缩放）。大多数情况下，这个设置值是不适用的。如果窗体的大小改变时，它不改变其内容的大小，那么边框就不应该缩放。

窗体通常用于创建对话框，即用于搜集用户输入的信息的窗口。这样的对话框并不具有可缩放的边框。所以，BorderStyle 属性要视情况而设置。

(2) 在规定情况下使用最佳界面组件。

- 如果 TextBox 控件仅用于显示数据的，请将此控件换成 Label 控件。
- 使用滚动条 ScrollBar 来指明数量和速度。
- 只有在绝对必要时才选用 Picture 控件，建议使用 Image 控件来显示图片。
- 当允许用户选定和取消选项项目时，应该尽可能使用复选框 CheckBox。
- 用选项按钮 OptionButton 来显示包含 5 个或少于 5 个项目的静态列表。
- 使用列表框 ListBox 显示带有 5 个以上项目的动态列表和静态列表。
- 若运行一段程序需要长时间的等待，使用进度条 ProgressBar 可以缓和用户的紧张。

(3) 提供便于理解和使用的菜单

按照与其他流行的 Windows 应用程序相一致的方式，对菜单进行格式化和组织。统一的菜单组织结构不仅表现在顶层菜单项的顺序和样式上，也要与其他许多流行的 Windows 程序保持一致性。

例如，“新建”菜单项总是“文件”主菜单上的第一个项目，但决不会在“编辑”主菜单上。

将快捷键赋予常用菜单项，用户不必访问菜单，就可以启动菜单命令。

木瓜工作室

WWW.MUGUA.NET

网址 www.mugua.net 《Visual Basic 6.0 完全自学手册》热销中

例如，我们在开发程序时，习惯按 F5 键进行程序的编译和运行，或者按 Ctrl + F 5 键开始全面编译。

(4) 精心设置所有窗体的 Tab 键顺序

Tab 键是 Windows 在窗体上使焦点向前（或使用 Shift + Tab 键向后）移过控件时所用的标准操作键。若要为窗体定义 Tab 次序，只要为第一个接收焦点的控件赋予 0 作为其 TabIndex 值，下一个控件的 TabIndex 值为 1，依此类推。

(5) 创建对话框中的默认命令按钮和取消命令按钮

大多数应用程序都会显示一个模式对话框。这些对话框在关闭前用于搜集用户输入的信息。一般来说，这类对话框至少包含两个命令按钮，即 OK（确定）和 Cancel（取消）。

- 若要将 Esc 键赋予一个命令按钮，请将该命令按钮的 Cancel 属性设置为 True。
- 若要将 Enter 键赋予一个命令按钮，可以将该按钮的 Default 属性设置为 True。

(6) 设置与数据域相关联的控件的 MaxLength 属性

数据库表格中的大多数字段都有一个它们能够存放的最大字符数。如果试图将超出允许范围的字符数目存放到数据表中，就会产生错误。所以，有必要限制用户输入的最大长度字符数。

(7) 使用功能良好的消息框

消息框 MsgBox 函数非常灵活，运用该函数，可以提出问题，或者对某些情况进行说明。还可以控制显示的图标类型以及用户可以使用的按钮。使用 MsgBox 函数要注意以下几个情况：

- 编写消息时，要使用正式的语气，不要使用夸大性的词汇，也不要使用缩写词。文字应该简明易懂，不要使用过分花哨的用语。请记住，消息框不是炫耀文字技巧的场所。它只是用来向用户传达简明扼要的消息。当消息框必须提出问题时，问题必须准确达意，使用能够采取恰到好处的决断。另外，消息框的标题不能省略，可以使用“系统提示”“系统错误”等文字。
- 为既定的情况选择适当的消息框。消息的类型有四种。坚决禁止不带有任何图标的消息框，那是一种不负责任的做法。
- 提供真正有意义的按钮。消息框能够显示许多不同的按钮，这取决于你选择在消息框中显示哪个按钮或哪些按钮。

| 常数 | 值 | 描述 |
|--------------------|------|-------------------------------------------------------------------------------------------------------------|
| vbOKOnly | 0 | 只显示OK（确定）按钮 |
| vbOKCancel | 1 | 显示OK（确定）及Cancel（取消）按钮 |
| vbAbortRetryIgnore | 2 | 显示Abort（终止）、Retry（重试）及Ignore（忽略）按钮 |
| vbYesNoCancel | 3 | 显示Yes（是）、No（否）及Cancel（取消）按钮 |
| vbYesNo | 4 | 显示Yes（是）、No（否）按钮 |
| vbRetryCancel | 5 | 显示Retry（重试）及Cancel（取消）按钮 |
| vbCritical | 16 | 显示Critical Message图标  |
| vbQuestion | 32 | 显示Warning Query图标  |
| vbExclamation | 48 | 显示Warning Message图标  |
| vbInformation | 64 | 显示Information Message图标  |
| VbDefaultButton1 | 0 | 第一个按钮是默认值 |
| VbDefaultButton2 | 256 | 第二个按钮是默认值 |
| VbDefaultButton3 | 512 | 第三个按钮是默认值 |
| VbDefaultButton4 | 768 | 第四个按钮是默认值 |
| VbApplicationModal | 0 | 应用程序强制返回；应用程序被挂起，直到用户对消息框作出响应才继续工作 |
| VbSystemModal | 4096 | 系统强制返回；全部应用程序都被挂起，直到用户对消息框作出响应才继续工作 |

木瓜工作室

WWW.MUGUA.NET

网址 www.mugua.net 《Visual Basic 6.0 完全自学手册》热销中

值的说明:

- 第一组值(0-5)描述了消息框中所显示按钮的类型与数目;
- 第二组值(16,32,48,64)描述了图标的样式;
- 第三组值(0,256,512,768)指定哪一个按钮是默认值;
- 第四组值(0,4096)则决定消息框的强制返回性。可以将这些数字相加来生成一个组合的对话框样式参数值,但是在相加的时候。只能从每组值中取用一个数字。

注意: 建议采用表中的常数名称来代替数值,可以增加程序的可读性。不必担心使代码的输入工作变得繁琐,代码编辑器的语句完成功能已经提供此便利。

当执行到一个 MsgBox 语句或 MsgBox 函数时,屏幕上就出现一个消息对话框,显示标题、提示信息、图标及按钮,并等待用户单击按钮。如果是 MsgBox 函数,用户单击按钮后会返回一个整型值,让程序了解用户单击的是哪一个按钮,以便作出不同的处理。

下表列出了 MsgBox 函数的返回值及其含义。

| 常数 | 值 | 用户单击的按钮 |
|----------|---|-------------|
| vbOK | 1 | OK (确定) |
| vbCancel | 2 | Cancel (取消) |
| vbAbort | 3 | Abort (终止) |
| vbRetry | 4 | Retry (重试) |
| vbIgnore | 5 | Ignore (忽略) |
| vbYes | 6 | Yes (是) |
| vbNo | 7 | No (否) |