第 1 章　您好！Visual Basic

从现在开始，我们将学习用 Visual Basic 编写 Windows 应用程序。本章将构造第一个 Visual Basic 应用程序 HELLO。通过 HELLO 程序的构造过程，读者将逐步熟悉 Visual Basic 的基本界面和基本术语，并对 Visual Basic 应用程序的构造过程有一个总体的认识。

1.1　安装和启动 Visual Basic

和大多数 Windows 应用程序一样，Visual Basic 有一个名叫 SETUP.EXE 的安装程序，它负责将 Visual Basic 安装到硬盘上。

按下述步骤安装 Visual Basic：

- 在 MS-DOS 提示符下键入：

  WIN↙

  以启动 Windows。
- 把 1 号磁盘 Disk 1 插入 A 驱动器（或 B 驱动器）。
- 启动 Windows 的 File Manager（文件管理器）从 File 菜单选择 Run 命令。Windows 下是显示 Run 对话框，如图 1.1 所示。
- 如果 Disk 1 插在 A 驱动器中，则在 Run 对话框中键入：

  A:SETUP↙

  如果 Disk 1 插在 B 驱动器中，则在 Run 对话框中键入：

  B:SETUP↙
- 遵循 SETUP 程序给出的指示，完成安装过程

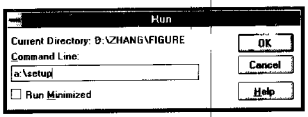

图 1.1　Run 对话框

可用下述两种方法启动 Visual Basic：

1. 在 Windows 的 Program Manager（程序管理器）中双击 Visual Basic 3.0 图标（将鼠标指针放在该图标之上，再快速连按两下鼠标左按钮）。Windows 下是显示 Visual Basic 3.0 应用程序组，如图 1.2 所示。双击 Mircosoft Visual Basic 图标。

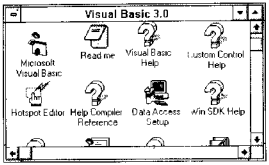

图 1.2 Visual Basic 3.0 应用程序组

2. 在 MS DOS 提示符下，键入：

WIN VB\VB↵

1.2 HELLO 程序

一个 Visual Basic 应用程序叫做一个项目(project)，它由多个文件组成。下面为 HELLO 程序建立一个新项目：

- 启动 Visual Basic。(Visual Basic 的显示如图 1.3 所示。)
- 单击 File 菜单，把鼠标指针移到 File 菜单标题上，按一下鼠标左按钮。Visual Basic 于是打开 File 菜单。
- 单击 New Project 菜单项。

图 1.3 启动后的 Visual Basic

Visual Basic 于是显示各种窗口，其中有一个窗口的标题是 Form1，如图 1.4 所示。这就是在 Visual Basic 中作为用户界面的程序窗口：窗体(Form)。图 1.4 所示的窗体是个空白窗口，我们所要做的是在窗体中加入各种 Visual Basic 工具。

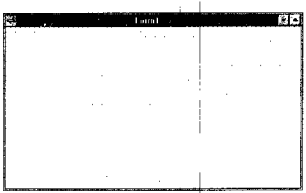

图 1.4 空白窗体

1.2.1 保存新项目

我们迄今还没有做任何实质性的工作，但不妨先保存这个新项目。

前面说过，一个项目由多个文件组成。具体说来它包括以下两种文件：

1. make 文件。make 文件的扩展名为.MAK。一个项目只有一个 make 文件。Visual Basic 根据 make 文件提供的信息构造项目。

2. 窗体文件。窗体文件的扩展名为.FRM。一个项目可有多个窗体文件。HELLO 程序是个单窗体项目，所以只包含一个窗体。

按下述步骤保存 HELLO 项目：

■ 从 File 菜单选择 Save Project As。Visual Basic 于是显示如图 1.5 所示的对话框，询问是否保存窗体文件 Form1.frm。

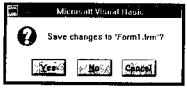

图 1.5 对话框

■ 单击 Yes 按钮，表示想保存窗体。Visual Basic 于是显示 Save File As 对话框，如图 1.6 所示。Save File As 对话框用来设置文件名和文件所在的目录。读者可通过在 Directories 列表框中双击目录来选择合适的目录。File Name 框中是缺省的文件名，由于缺省名 Form1.frm 很不直观，所以应重新命名。

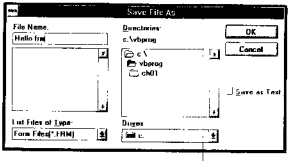

图 1.6 Save File As 对话框

■ 在 File Name 框中键入 HELLO.FRM，然后单击 OK 按钮。在图 1.6 中把 HELLO 项目的窗体文件以 HELLO.FRM 名保存到 C:\vbprog\CH01 目录中了。Visual Basic 接着显示 Save Project As 对话框。Save Project As 对话框用来指定 make 文件的文件名及其所在的目录，如图 1.7 所示。

■ 选择合适的目录（应和窗体文件 HELLO.FRM 所在的目录一致）。

■ 在 File Name 框中键入 HELLO.MAK，以取代原来的缺省名 Form1.mak。

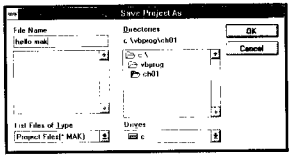

图 1.7 Save Project As 对话框

1.2.2 项目窗口

这样就构造了一个名叫 HELLO.MAK 的项目，它只包含单个窗体文件。

■ 从 Windows 菜单选择 Project、Visual Basic 子是显示如图 1.8 的项目窗口。

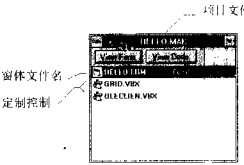

图 1.8 项目窗口

项目窗口的标题是项目文件名 HELLO.MAK，窗口中列出了一系列文件，其中 HELLO.FRM 是项目的窗体文件。项目窗口中有两个按钮：View Form 按钮和 View Code 按钮，分别用于查看窗体和查看代码窗口。

1.2.3 属性窗口

在项目窗口中，窗体文件名 HELLO.FRM 的右边显示着 Form1，Form1 是该窗体的名字，窗体是一个程序窗口，它有自己的名字、长、宽、背景色等性质，这些性质叫做窗体的属性(properties)。可以通过属性窗口设置属性。

■ 从 Windows 菜单选择 Properties。Visual Basic 子是显示属性窗口，如图 1.9 所示。

属性窗口由对象框、设置框和属性列表组成。属性列表的左列是一系列属性，如 AutoRedraw、BackColor 等，右列是各属性的缺省设置，也就是 Visual Basic 自动设置好的值。

对象框用来指示当前显示的是哪个对象的属性列表。"对象"是个面向对象的术语，例如窗体、命令钮、滚动条等都是对象，Visual Basic 程序处理的正是一个个对象。属性用来

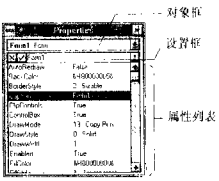

图 1.9 属性窗口

定义对象的外观和行为。例如窗体是一个对象，属性 Caption 指定窗体的标题，该标题显示在窗体窗口的标题栏栏上。BackColor 也是窗体的属性它指定窗体的背景色。

在图 1.9 所示的属性窗口中，对象框中的文本内容是"Form1 Form"，表明这是一个名叫 Form1 的窗体的属性窗口。Form1 正是图 1.4 所示的窗体，该窗体的标题 Form1 显示在窗体窗口上的标题栏上。请注意，Name(名字)属性和 Caption(标题)属性是两个不同的属性，对象框中显示的是名字，属性列表中 Caption 属性右边显示的是标题，Name 属性右边显示的是名字。

下面改变 Form1 窗体的 Caption 属性：

- 单击属性列表中的 Caption 属性。Visual Basic 于是加亮 Caption 属性，并在设置框中显示 Caption 属性的当前值 Form1。设置框用来设置属性的值。
- 在设置框中键入 The Hello Program，按回车键。如图 1.10 所示，Visual Basic 于是在 Form1 窗体的标题栏上显示 The Hello Program。

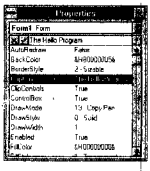

图 1.10 改变 Form1 窗体的 Caption 属性

下面设置 Form1 窗体的 BackColor 属性：

- 单击属性列表中的 BackColor 属性。
- 单击设置框右端的二个圆点，如图 1.11 所示。Visual Basic 于是显示调色板窗口，如图 1.12 所示。
- 在中意的颜色块上单击一下。Visual Basic 于是相应地改变 Form1 窗体的背景色。

图 1.11 选择 BackColor 属性

图 1.12 调色板窗口

设置对象的属性一般有三种方式：

1. 设置框右端的向下箭头为暗淡色时，表明该箭头无效，这时只能直接键入所需的值。例如设置 Caption 属性时，必须直接键入想要的标题文本。

2. 设置框右端显示三个圆点时，一般单击这三个圆点。Visual Basic 将打开一个窗口或对话框，用户可在其中选择所需的设置。

3. 当设置框右端的向下箭头正常显示时，可以通过单击该箭头来打开一个下拉式列表，用户在列表中选择所需的值。如图 1.13 所示。这种情况下也可以直接在设置框中键入所需的值，但键入的值只能是下拉式列表中出现的设置。

前面曾提到 Form1 窗体的 Name 属性，下面改变 Name 属性：

- 在属性窗口中单击属性列表右边的垂直滚动条的向下滚动箭头，直到显示出 Name 属性，如图 1.13 所示。Visual Basic 于是向上滚动属性列表。
- 单击属性列表中的 Name 属性。
- 在设置框中键入 frmHello。

这样就把该窗体改名为 frmHello。

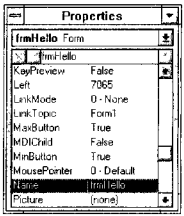

图 1.13 Name 属性的设置

1.3 附加 HELLO 程序的代码

在前面我们认识了 Visual Basic 的程序窗口、窗体，并通过属性窗口设置了窗体的某些属性，这时窗体的外观和最初显示的空白窗体已有所不同了。等会儿运行 HELLO 程序时，读者将发现，运行时的窗体和此时处于设置状态的窗体外观上是一样的。在设计阶段就可以预见到运行时的窗体外观。这就是"可视"(visual)的含义。

Visual Basic 应用程序的构造过程一般分为两个阶段：

1. 可视实现阶段；

2. 附加代码阶段。

在可视实现阶段不需要编写任何代码，所要做的只是在窗体中加入各种 Visual Basic 工具，例如命令钮、文本框等控制。并且通过属性窗口设置这些对象的属性。我们前面所做的工作就是 HELLO 程序的可视实现，只是 HELLO 程序的可视实现非常简单，没有在窗体中加入各种控制。

可视实现阶段结束时，已经可以预见到运行时的窗体外观，"可视实现"由此得名。

在附加代码阶段，编写并键入程序代码。下面为 HELLO 程序附加上的代码。

■ 单击项目窗口的 View Code 按钮。或者在 FrmHello 窗体内双击，Visual Basic 于是显示 HELLO 程序的代码窗口，如图 1.14 所示。有些情况下，项目窗口的 View Code 按钮为暗淡色，这时应先单击项目窗口中的窗体文件名 HELLO. FRM，然后再单击 View Code 按钮。

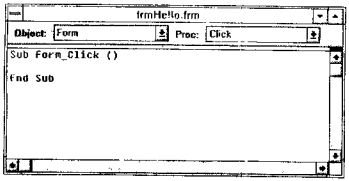

图 1.14 代码窗口

■ Object(对象)框右端的向下箭头，Visual Basic 于时下拉对象列表框。

■ 在打开的下拉式列表框中单击 From 项。Object 框中于是显示 *Form*。

■ 单击 Proc(过程)框右端的向下箭头。Visual Basic 于是下拉事件列表框。

■ 在打开的下拉式列表框中单击 Click 项。Proc 框中于是显示 Click。代码窗口中于是显示事件过程：

```
Sub Form_Click()

End
```

■ 在事件过程 Form_Click()中键入下述代码：

```
Sub Form_Click()
  Print "Hello,Visual Basic"
End Sub
```

■ 从 File 菜单选择 Save Project，保存所做的工作。

1.4 运行 HELLO 程序

- 从 Run 菜单中选择 Start。Visual Basic 开始运行 HELLO 程序，并显示窗体窗口。
- 在窗体内单击一下，Visual Basic 开始显示"Hello，Visual Basic"，如图 1.15 所示。

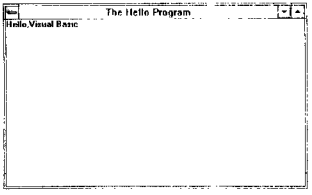

图 1.15 运行时的 HELLO 程序

- 不断单击窗体，Visual Basic 依次相应显示"Hello，Visual Basic"。
- 从 Run 菜单中选择 End，结束 HELLO 程序。

1.5 事件过程

下面分析 HELLO 程序的工作原理。事件过程 From_Click() 如下所示：

```
Sub Form_Click()
  Print "Hello,Visual Basic"
End Sub
```

其中最第一行和最后一行是 Visual Basic 已经编写好了的，读者所键入的只是中间一行。

关键字 Sub 表示着一个过程的开始。From_Click() 是事件过程名，右边() 是空的参数表。其中 Form 和 Click 分别是指"窗体"和"单击"，意思是每当用户单击窗体时，事件过程 From_Click() 就会自动执行。

Visual Basic 应用程序首先是 Windows 应用程序，是由 Windows 操作系统管理并支持的。可以把 HELLO 程序的执行过程用形象化语言描绘如下：刚开始运行时，HELLO 程序就躺着睡大觉，Windows 却精神抖擞地四处查看动静。当用户单击 FrmHello 窗体时，Windows 立刻踢醒 HELLO 程序，通知它："用户单击 frmHello 窗体了！"。HELLO 程序当即清醒过来，并琢磨"frmHello 窗体(From) 被单击(Click)……我应该执行事件过程 From_Click()！"，HELLO 程序于是执行 From_Click()，执行完后又埋头大睡，直到 Windows 再次踢醒它……

用专业用语描述上述过程，开始运行时 HELLO 程序被挂起，当用户单击 frmHello

窗体时，“单击”事件发生，Windows 于是把相应的信息传给 HELLO 程序，触发事件过程 From_Click()的执行。

需要注意的是：在从前的程序设计语言中，函数和过程总是被调用执行，而在 Visual Basic 应用程序中，总是事件触发事件过程的执行。换句话说，事件过程响应已发生的事件而自动执行，这里“自动”是和“被调用”意思相对的。Windows 操作系统在整个过程中做了很多工作。

那么事件过程 From_Click()到底做了什么呢？

From_Click()过程执行语句

```
Print "Hello,Visual Basic"
```

这是一个 Print 语句，在窗体上打印文本 Hello,Visual Basic 双引号标识文本串的开始和结束。

最后一行的 End Sub 表示看该过程的结束，它和第一行的 Sub 相匹配。

还有一个问题需要说明：当程序中有多个窗体时，From_Click()到底是哪个窗体的事件过程呢？如图 1.14 所示，代码窗口的标题栏上已清楚地表明这是窗体文件 HELLO.FRM 的代码，所以该 From_Click()事件过程对应 frmHello 窗体。

1.6 改进 HELLO 程序

HELLO 程序的用户界面仍然是个空窗体，下面改进 HELLO 程序，在窗体中加入各种控制。

1.6.1 工具箱窗口

- 从 Windows 菜单选择 Project。Visual Basic 于是显示项目窗口。
- 单击项目窗口的 View Form 按钮，Visual Basic 于是显示窗体。
- 从 Windows 菜单选择 Toolbox。Visual Basic 于是显示工具箱窗口，如图 1.16 所示。

工具箱窗口中包含各种控制的图标，我们所要做的是选择合适的控制，并把它们放入窗体中。

图标(icon)是一种小的图形图象，可用来代表最小化的程序窗口、一个控制甚至一个命令。控制(control)是 Visual Basic 已经预定义的子窗口，它能完成指定类型的功能。标准版的 Visual Basic 3.0 在工具箱窗口中显示了 20 种控制的图标，如图 1.16 所示。专业版的 Visual Basic 3.0 提供了更多的定制控制，如下所述：

1. 指针(Pointer)　这是工具箱窗口中唯一不能用来绘制控制的工具，只起到指示作用。

2. 图片框(Picture Box)　用于显示图形图象、接收图形方法的输出，还可用来容纳其它控制。

3. 标签(Lable)　用于显示和输出文本信息，但不能用作输入界面。

4. 文本框(Text Box)　用于输入和输出文本。

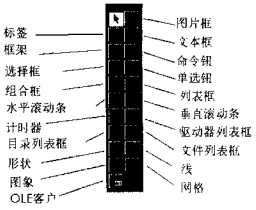

图 1.16 工具箱窗口

5. 框架(Frame) 用于将性质相同的控制集中在一起。必须先绘制框架，然后在框中逐一加入该组控制。

6. 命令钮(Command Button) 用于创建一个按钮。当用户单击命令钮时，即可执行一个命令或动作。

7. 选择框(Check Box) 是供用户选择 True/False 和 Yes/No 的控制。用户可以同时选定多个选择。

8. 单选钮(Option Button) 一般成组使用，用户只能选定其中一个选择。

9. 组合框(Combo Box) 是列表框和文本框的组合。用户要么从列表框中选择表项，要么在文本框中输入值。

10. 列表框(List Box) 用于显示一列表项(item)，供用户选择。当表项太多一次显示不下时，可以通过滚动条滚动列表。

11. 水平滚动条(Horizontal Scroll Bar) 用于快速浏览数据和信息，也是输入和输出的一种手段。

12. 垂直滚动条(Vertical Scroll Bar) 用于快速浏览数据和信息，也是输入和输出的一种手段。

13. 计时器(Timer) 用于捕获每隔指定时间间隔发生的计时器事件。计时器控制在运行时缺省不可见。

14. 驱动器列表框(Drive List Box) 用于显示有效的驱动器名。

15. 目录列表框(Directory List Box) 用于寻找和切换当前驱动器上的目录和路径。

16. 文件列表框(File List Box) 用于显示可供用户打开、保存以及其它操作的文件。

17. 形状控制(Shape) 用于在设计时绘制各种形状，如矩形、圆角矩形、正方形、圆角正方形、椭圆和圆。

18. 线控制(Line) 用于在设计时绘制各种风格的直线。

19. 图象控制(Image) 用于显示各种图形图象，例如位图、图标和元文件。图象控制

只用于静态环境，即不再修改创建后的位图或图标，使用时需要的内存资源比图片框少。图片框适合动态和静态环境，例如在程序运行时移动图片框中的图片，它比图象控制更灵活。

20. 网格控制(Grid)　用于创建象电子表格一样的网格，网格由行、列数据组成。

21. OLE 客户控制(OLE client)　在应用程序之间建立对象的链接和嵌入，它用作客户对象。

我们将在后面的学习中逐一介绍这些控制。

下面在 frmHello 窗体中加入命令钮。

■ 把鼠标指针移到工具箱窗口中的命令钮图标，双击命令钮图标。

Vasual Basic 于是把一个命令钮放入 frmHello 窗体的中央，如图 1.17 所示。

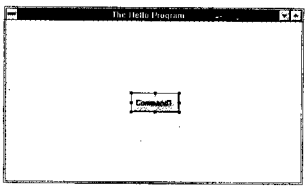

图 1.17　放入的命令钮

■ 拖拉该命令钮。做法是：把鼠标指针移到命令钮上，按住鼠标左按钮不放，然后移动鼠标到窗体的右下角，释放鼠标左按钮。

Vasual Basic 于是把命令钮移到窗体的右下角，如图 1.18 所示。

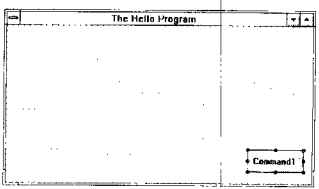

图 1.18　拖拉后的命令钮

下面设置命令钮的属性：

■ 从 Windows 菜单选择 Properties。Vasual Basic 于是显示属性窗口。

■ 单击对象框右端的向下箭头。Vasual Basic 于是下拉对象列表。

■ 在打开的下拉式列表中选择 Command1。对象框中的文本内容变为"Command1 CommandButton",表示属性窗口中当前显示的是命令钮 Command1 的属性列表,如图 1.19 所示。

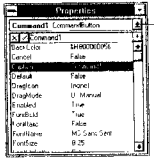

图 1.19 命令钮的属性窗口

■ 单击属性列表中的 Caption 属性,然后在设置框中键入 Exit。Vasual Basic 于是把命令钮上显示的文本改变为 Exit。

命令钮的 Caption 属性是指显示命令钮上的文本内容,即命令钮的标题;而窗体的 Catpion 属性是指显示在窗体标题栏上的文本内容,即窗体窗口的标题。可见,虽然属性名相同,但对不同的对象、同名的属性其具体含义还是有所差别的。

该命令钮的缺省名字也为 Cammand1,下面重新设置它的 Name 属性:

■ 把命令钮的 Name 属性设置为 cmdExit。

用同样的方法加入另一个命令钮,步骤如下:

■ 在工具箱窗口中双击命令钮图标,以加入命令钮 Command2。

■ 把新加入命令钮的 Caption 属性和 Name 属性分别置为 Clear 和 camClear。

■ 这时的 frmHello 窗体中包含两个命令钮 cmdExit 和 cmdClear,它们的标题分别为 Exit 和 Clear,如图 1.20 所示。

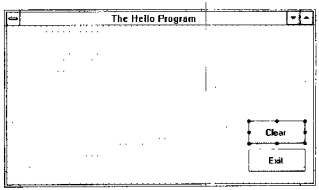

图 1.20 完成后的 frmHello 窗体

改进版 HELLO 程序的属性表可总结如下：

表 1.1 frmHello 窗体的属性表

对　象	属　性	设　置
窗　体	Name	frmHello
	Caption	The Hello Program
	BackColor	蓝为黄色
命令钮	Name	cmdExit
	Caption	Exit
命令钮	Name	cmdClear
	Caption	Clear

1.6.2 附加代码

迄今已完成 HELLO 程序的可视实现，下面为 HELLO 程序附上代码：

- 双击 Exit 按钮。Vasual Basic 于是显示代码窗口。
- 在事件过程 cmdExit_Click()中输入下述代码：

```
Sub cmdExit_Click()
  End                          '终止 HELLO 程序
End Sub
```

每当用户单击 Exit 按钮时，事件过程 cmdExit_Click()就会自动执行。该过程执行 End 语句。End 语句用来终止程序的运行。请注意，cmdExit 是 Exit 按钮的 Name 属性值。

- 在代码窗口中单击 Object 框右端的向下箭头，然后在打开的对象列表中选择 cmdClear。
- 在事件过程 cmdClear_Click()中输入下述代码：

```
Sub cmdClear_Click()
  Cls                          '清除 frmHello 窗体
End Sub
```

每当用户单击 Clear 按钮时，事件过程 cmdClear_Click()就会自动执行。该过程执行 Cls 语句，Cls 语句清除窗体中输出显示的文本和图形。

- 从 File 菜单选择 Save Project，保存所做的工作。

1.6.3 运行改进版的 HELLO 程序

- 从 Run 菜单中选择 Start，运行 HELLO 程序。
- 单击窗体，Vasual Basic 于是显示"Hello, Vasual Basic"。
- 单击 Clear 按钮。Vasual Basic 于是清除窗体上显示的文本串。
- 单击 Exit 按钮。Vasual Basic 于是终止 HELLO 程序。

下面小结 Vasual Basic 应用程序的构造过程：

- 从 File 菜单选择 New Project，打开一个新项目。
- 从 File 菜单选择 Save Project As，依次保存窗体文件（.FRM）和 make 文件（.MAK）。
- 根据窗体的属性表构造窗体。这是 Vasual Basic 程序设计的可视实现阶段。
- 在项目窗口中单击 View Code 按钮，或是双击窗体，窗体中某个控制，以显示代码窗口。
- 在代码窗口中为程序附上代码。这是 Vasual Basic 程序设计的附加代码阶段。
- 从 File 菜单选择 Save Project，保存所做的工作。
- 从 Run 菜单中选择 Start，运行程序。
- 从 Run 菜单中选择 End，终止程序。也可通过程序自身提供的方式(Exit 按钮)终止程序。

为了节省篇幅，以后不再重述上述步骤，但读者每次构造 Vasual Basic 应用程序时，都应该遵循上述步骤。

1.7 建立可执行文件

在前面，我们通过从 Run 菜单选择 Start 来执行 HELLO 程序，这是在 Vasual Basic 环境下以解释的方式执行程序。还可以将设计完成的文件编译成可执行文件(.EXE)，并与 Windows 联结，以后在 Windows 下就可以直接执行该应用程序了。

按下述步骤建立可执行文件：

- 从 File 菜单中选择 Make EXE File。Vasual Basic 于是显示 Make EXE File 对话框，如图 1.21 所示。

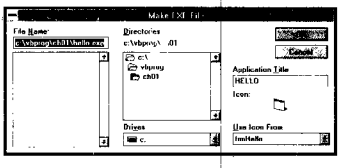

图 1.21 Make EXE File 对话框

- 在 Directories 列表框中选择合适的目录，然后在 File Name 框中键入文件名，其中文件扩展名为.EXE。例如图 1.21 中，在 C:\VBPROG\CH01 目录下建立了可执行文件 HELLO.EXE。

读者可以在 Windows 的 File Manager 中运行 HELLO.EXE

第 2 章　程序设计基础

本章介绍 Visual Basic 常用的数据结构、语法和控制结构，这是 Visual Basic 程序设计的基础，由于本章内容比较枯燥，建议读者较快地浏览全章，理解基本的概念，至于具体的用法会在后面的学习中逐步掌握。

2.1　数 据 类 型

程序由一系列的操作步骤组成，数据是操作的对象。数据类型是数据的表示方法，它指定了数据的取值范围及其操作。例如，整型是一种数据类型，它可以是从－32768 到 32767 之间的任一整数值，整数之间可以进行加、减、乘、除等各种运算。

Visual Basic 的数据类型有两种：标准数据类型和用户自定义数据类型。

2.1.1　标准数据类型

Visual Basic 的标准数据类型如表 2.1 所示。

表 2.1　Visual Basic 的标准数据类型

类型名称	说明	取值范围	所用字节数
Integer	整型	－32768～32767	2 字节
Long	长整型	－2,147,483,648～2,147,483,647	4 字节
String	字符串型	0～65535 个字符	每个字符占 1 字节
Currency	货币型	－922337203685477.5808～922337203685477.5807	8 字节
Single	单精度型	-1.40×10^{-45}～-3.40×10^{38}	4 字节
Double	双精度型	-1.94×10^{-324}～$\pm1.79\times10^{308}$	8 字节
Variant	变体型	上述之一	上述之一

整型是 16 位(字节 8 位)的整数，例如 0，－17，32767 和－32768 都是整数。

长整型是 32 位的整数，例如 0，32767，－32767，4236785，－2147483648 等都是长整数。可见长整型的取值范围比整型大。

字符串型是字符序列，例如"A"，"ABC"等都是字符串。双引号是字符串的字界符，指字符串的开始和结束。有两个特殊的字符串：空串和空格串。空串("")是不含任何字符的字符串，串的长度(即串所含字符数)为 0。而空格串("　")中的字符全是空格符，串长由所含空格符的个数决定，但不为 0。字符串最多可包含 65535 个字符。

货币型是为表示货币值而设计的，小数位固定为 4 位。

单精度型和双精度型分别是 32 位浮点数和 64 位浮点数。浮点数就是通常所说的实数。单精度型浮点数 3.37E+38 表示 3.37×10^{38}，3.37E-38 表示 3.37×10^{-38}，-3.37E-38 表示 -3.37×10^{-38}。双精度型浮点数 3.37D+38 表示 3.37×10^{38}，-1.67D-308 表示 -1.67×10^{-308}。在表 2.1 中，单精度型浮点数的取值范围为 $\pm1.40\times10^{-45}$—$\pm3.40\times10^{38}$意思是正的单精度型浮点数从 1.40×10^{-45}—3.40×10^{38}取值，负的单精度型浮点数从 -1.40×10^{-45}—-3.40×10^{38}取值。

变体型是一种可变的数据类型，可以表示任何值。对程序中未说明数据类型的变量，Visual Basic 把它看作变体型。

2.1.2 用户自定义数据类型

描述一个学生，我们关心其姓名、性别、年龄等数据。如果用 Visual Basic 的标准数据类型，则需要分别用字符串型描述姓名和姓别、整型描述年龄……。显然，这种孤立的描述割断了信息之间的联系，体现不出姓名、性别和年龄等数据都是为描述学生而服务的。由此想到，能不能建立"学生"这种数据类型，姓名、性别等是其中的元素。如下所示：

```
Type Student
    Name    As String
    Sex     As String
    Age     As Integer
End Type
```

这样就建立了用户自定义类型 Student，Name、Sex 和 Age 是它的元素，分别表示姓名、性别和年龄。Name AS String 表明元素 Name 是字符串型，Age As Integer 表明元素 Age 是整型。

格式

```
Type 自定义数据类型的名称
    元素名称    AS    数据类型
    元素名称    AS    数据类型
    ……
    ……
End  Type
```

[例]

定义用户自定义类型 Dog，用来描述狗：

```
Type Dog    Name        AS String * 9   '名字
    Sex                 AS String * 7   '性别
    Birthday            As Double       '生日
    Color               As String * 7   '颜色
    Owner               As String * 20  '主人的名字
```

```
End Type
```

其中，Name As String * 9 表示 Name 是字符串型，且这个字符串最多容纳 9 个字符。单引号后面的文字是注释。

2.2 常量和变量

数据在程序中以常量和变量的形式出现。在程序运行过程中，其值不能改变的量称为常量。例如整型常量 0、11、-16，单精度型常量 1.23E+4、-2.67E-12，字符串常量 "HELLO VISUAL BASIC"。也可以用一个标识符代表一个常量：

```
Const True        -1  '定义常量 Ture
Const False       0   '定义常量 False
```

这样就定义了常量 True 和 False，其值分别为 -1 和 0。

格式

```
Const    常量名=常量表达式
```

需要指出的是，在程序中不允许改变常量的值。

[例]

```
Const    Max=100
Const    Min=0
Const    Mid=Max+Min-50
```

上面说明了常量 Max，Min 和 Mid。Max，Min 和 Mid 都是代表常量的标识符。Visual Basic 对这种标识符有一定的规定，这就是常量和变量的命名规则。

2.2.1 命名规则

在给 Visual Basic 常量和变量命名时，必须遵循下述规则：

1. 名字必须以字母字符开头，
2. 名字中的字符只能是字母、数字或下划线字符(_)。
3. 名字不多于 40 个字符。
4. 不能用关键字作为常量名或变量名。

关键字是 Visual Basic 中规定了特殊意义的标识符。例如 Visual Basic 用 Const 表示常量说明语句的开始，用 Type…End Type 表示用户自定义数据类型，因此，Const，Type，End 等都是关键字，不能作常量名或变量名。

Visual Basic 不区分变量字母的大小写，因此，MyName，myName 和 myname 是同一变量名。

[例]