CHAPTER 3

# 第 3 章

# VB语言的基本控制结构

计算机能够完成很多任务，实质上这些工作都是按照人们事先编写好的程序执行的。程序是计算机的灵魂，而给计算机以灵魂的是程序设计人员。熟悉一门语言的程序控制结构是程序开发人员必备的一种技能。本章简要介绍各种算法的基本描述方法，重点介绍 Visual Basic 的控制结构——顺序结构、分支结构、循环结构，以及常用的算法。

**本章学习要点**

- 了解算法的概念、特征及描述方法。
- 理解结构化程序设计思想。
- 掌握三种程序控制结构。
- 领会常见算法的 VB 实现。
- 单选按钮、复选框及框架的属性、方法和事件。

## 3.1 算法与结构化程序设计

### 3.1.1 算法概述

**1. 算法的概念**

所谓算法是指对解题方案的准确而完整的描述。算法分为数值型算法与非数值型算法两种。非数值型算法又称为符号处理。

**2. 算法的基本要素**

算法由两种基本要素组成：一是对数据对象的运算和操作；二是算法的控制结构。

**3. 算法设计基本方法**

计算机解题的过程实际上是在实施某种算法，这种算法称为计算机算法。基本算法有如下几种。

(1) 列举法：基本思想是根据提出的问题，列举所有可能的情况，并用问题中给定的条件检验哪些是需要的，哪些是不需要的。

(2) 归纳法：基本思想是通过列举少量的特殊情况，经过分析，最后找出一般的关系。

(3) 递推：指从已知的初始条件出发，逐次推出所要求的各中间结果和最后结果。其中，初始条件或是问题本身已经给定，或是通过对问题的分析与化简而得到确定。

(4) 递归：人们在解决一些复杂问题时，为了降低问题的复杂程度总是将问题逐层分解，最后归结为一些最简单的问题。这种将问题逐层分解的过程，实际上并没有对问题进行求解，而只是当解决了最后那些最简单的问题后，再沿着原来分解的逆过程逐步进行综合，这就是递归的基本思想。

(5) 减半递推技术：所谓“减半”，是指将问题的规模减半，而问题的性质不变。所谓“递推”，是指重复“减半”的过程。

(6) 回溯法：有些实际问题很难归纳出一组简单的递推公式或直观的求解步骤，并且也不能进行无限的列举。对于这类问题，一种有效的方法是“试”。通过对问题的分析，找出一个解决问题的线索，然后沿着这个线索逐步试探。对于每一步的试探，若试探成功，就得到问题的解；若试探失败，就逐步回退，换别的路线再进行试探。这种方法称为回溯法。回溯法在处理复杂数据结构方面有着广泛的应用。

### 4. 算法的特征

一个算法应该具有以下 5 个重要的特征。

(1) 有穷性：一个算法必须保证执行有限步之后结束。

(2) 确定性：算法的每一步骤必须有确切的定义，执行何种动作应无二义性，目的明确。

(3) 输入：一个算法有 0 个或多个输入。

(4) 输出：一个算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

(5) 可行性：算法理论上能够精确地运行，而且人们用笔和纸做有限次运算后即可完成。

### 5. 算法的描述

人的思想要用语言来表达。算法是人求解问题的思想方法，是对解题过程的精确描述，同样也需要用语言来表示。表示算法的语言主要有自然语言、流程图、伪代码、程序语言等。

(1) 用自然语言表示。自然语言是人们日常所用的语言，如汉语、英语、德语等，可用于描述问题求解的算法。使用这些语言不用专门训练，所描述的算法也通俗易懂。

然而，用自然语言描述法也存在明显的缺点。

① 由于自然语言的歧义性，容易导致算法执行的不确定性。

② 自然语言的语句一般太长，从而导致了用自然语言描述的算法太长。

③ 由于自然语言表示是按照步骤的标号顺序执行的，因此当一个算法中循环和分支较多时就很难清晰地表示出来。

④ 自然语言表示的算法不便翻译成计算机程序设计语言。

**【例 3-1】** 设计一个算法，求 100 以内能被 4 整数的数。

**解题分析：** 设能被 4 整除的数为 $x$，令 $x=1,2,3,\cdots,100$。如果 $x$ 能被 4 整除，则输出 $x$，否则，检查下一个，直到 $x>100$ 为止。

自然语言算法可表示如下：

① 令 $x=1$；

② 如果 $x$ 能被 4 整除，则输出 $x$；

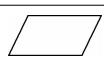
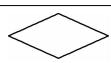
③ $x$ 在原值的基础上累加 1，$x+1\Rightarrow x$；

④ 如果 $x\leqslant 100$，则返回第②步；

⑤ 结束算法。

(2) 用流程图表示。流程图是算法的一种图形化表示方法，是描述算法的常用工具。表 3-1 列出了美国国家标准化协会 ANSI 规定的一些常用的流程图符号。使用流程图描述算法，与自然语言相比算法流程清晰简洁，容易表达分支结构；它不依赖于任何具体的计算机和计算机程序设计语言，从而有利于不同环境的程序设计；同一问题的流程图描述不唯一。

**表 3-1　常用流程图符号**

| 图　形 | 名　称 | 功　能 |
|---|---|---|
|  | 开始结束 | 表示算法的开始或结束 |
|  | 输入输出 | 表示算法中变量的输入或输出 |
|  | 处理 | 表示算法中变量的计算与赋值 |
|  | 判断 | 表示算法中的条件判断 |
|  | 流程线 | 表示算法中的流向 |
|  | 连接点 | 表示算法中的转接 |

**【例 3-2】** 用流程图描述算法：输入三个数，然后输出其中最大的数，如图 3-1 所示。

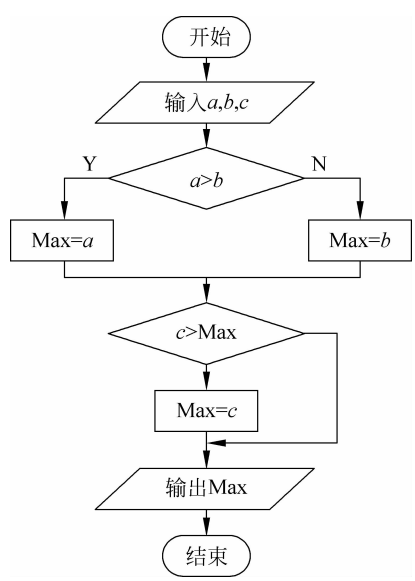



图 3-1　例 3-2 算法流程图

(3) 用伪代码表示。使用伪代码描述算法没有严格的语法限制，书写格式也比较自由，只要把意思表达清楚就可以了，较自然语言格式紧凑，也容易理解。它更侧重于对算法本身的描述，同时便于向计算机语言表示的算法过渡。但由于语言的种类繁多，伪代码的语句不容易规范，有时候会产生理解上的不一致。

**【例 3-3】** 用伪代码描述算法：给定一个四位数的年份，判断它是否为闰年。

**解题分析**：闰年也就是我们通常所说的：四年一闰，百年不闰，四百年再闰。判断闰年的条件是：该年份能被 4 整除但不能被 100 整除，或者能被 400 整除，则该年为闰年。

算法描述：

```
输入年份→year
```

```
If (year 能被 4 整除,但不能被 100 整除)或者(year 能被 400 整除) Then
    输出"是闰年"
Else
    输出"不是闰年"
End If
```

(4) 用计算机语言表示算法。计算机无法直接识别自然语言、流程图和伪代码形式的算法,只有用计算机语言编写的程序才能被计算机识别和处理,因此用自然语言、流程图和伪代码形式描述的算法,最终还要将它转换成计算机语言描述的程序。

**【例 3-4】** 用 VB 语言描述实现例 3-3 的判断闰年问题。

其描述代码如图 3-2 所示。

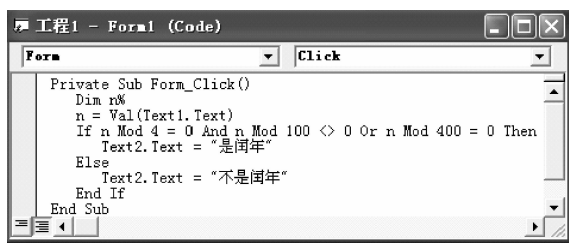



图 3-2 例 3-4 描述代码

## 3.1.2 结构化程序设计

### 1. 结构化程序设计思想

荷兰学者 E. W. Dijikstra 在 1965 年提出了"结构化程序设计"的思想,它的主要观点是采用自顶向下、逐步求精的程序设计方法,使用三种基本控制结构构造程序。任何算法都可以由顺序、选择、循环三种基本控制结构构造。在构造算法时,以顺序、选择、循环三种结构作为基本单元,同时规定基本结构之间可以并列和互相包含,不允许交叉和从一个结构直接转到另一个结构的内部去,使程序具有合理的结构,以保证和验证程序的正确性。这种方法要求程序设计者不能随心所欲地编写程序,而要按照一定的结构形式来设计和编写程序,使程序具有良好的结构,易于设计、理解和调试修改,以提高设计和维护程序工作的效率。

### 2. 结构化程序设计特征

结构化程序设计的特征主要有以下几点。

(1) 整个程序采用模块化结构,即达到"模块化"、"逐步求精"。

(2) 以三种基本结构的组合来描述程序,这三种结构都是单入口/单出口的程序结构,结构间接口简单,逻辑清晰。

(3) 有限制地使用转移语句。goto 语句的存在使程序的静态书写顺序与动态执行顺序十分不一致,导致程序难读难理解,容易存在潜在的错误,所以要有限制地使用。

(4) 结构中每一部分都应当有被执行到的机会。

(5) 采用结构化程序设计语言书写程序,并采用一定的书写格式使程序结构清晰,易于阅读。

### 3. 模块化与自顶而下的设计方法

结构化程序设计的总体思想是采用模块化结构，自上而下，逐步求精。即先设计第一层，把一个复杂的大问题分解为若干相对独立的小问题。小问题仍可以继续分解成若干子问题，直到小问题或子问题简单到能够直接用程序设计语言明确地描述出来为止。然后，对应每一个小问题或子问题编写出一个功能上相对独立的程序块来，这种像积木一样的程序块被称为模块。每个模块各个击破，最后再统一组装，这样，对一个复杂问题的解决就变成了对若干个简单问题的求解。这就是自上而下，逐步求精的程序设计方法，优点如下。

(1) 复杂系统化大为小，化繁为简。

(2) 便于维护。

(3) 提高系统设计效率(多人并行开发)。

## 3.1.3　三种程序控制结构

算法的实现过程是由一系列操作组成的，这些操作之间的执行次序就是程序的控制结构。1996 年，计算机科学家 Bohm 和 Jacopini 证明了这样的事实：任何简单或复杂的算法都可以由顺序结构、选择结构和循环结构这三种基本结构组合而成。所以，这三种结构就被称为程序设计的三种基本结构，也是结构化程序设计必须采用的结构。

### 1. 顺序结构

顺序结构表示程序中的各个操作是按照它们出现的先后顺序执行的，其流程如图 3-3(a)所示。图中的 S1 和 S2 表示两个处理步骤，这些处理步骤可以是一个非转移操作或多个非转移操作序列，甚至可以是空操作，也可以是三种基本结构中的任一结构。整个顺序结构只有一个入口 a 和一个出口 b。这种结构的特点是：程序从入口 a 开始，按顺序执行所有操作，直到出口处 b，所以称为顺序结构。

### 2. 选择结构

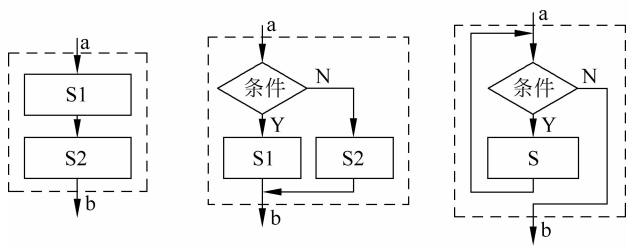
在程序设计中，经常需要根据不同情况，选择不同的算法，判断语句是必须的。如图 3-3(b)所示，选择结构的执行顺序是：当条件值为真时，执行语句序列 S1；当条件值为假时，执行语句序列 S2。

当 S1 和 S2 中的任意一个处理为空时，说明结构中只有一个可供选择的分支。例如 S2 为空，如果条件满足执行 S1 处理，否则顺序向下到流程出口 b 处。也就是说，当条件不满足时，什么也没执行，所以称为单选择结构。

### 3. 循环结构

同样，循环型结构也只有一个入口点 a 和一个出口点 b，如图 3-3(c)所示，循环终止是指流程执行到了循环的出口点。图中所表示的 S 处理可以是一个或多个操作，也可以是一个完整的结构或一个过程。

图中，先计算条件的值，为真时执行语句 S，然后返回再计算条件的值，为真时再执行语句序列 S，重复循环直到表达式的值为假时，则退出循环，转向执行下面的语句，退出循环结构。




(a) 顺序结构　(b) 选择结构　(c) 循环结构

图 3-3　三种程序控制结构

结构化程序要求每一基本结构具有单入口和单出口的性质是十分重要的,这是为了便于保证程序的正确性。设计程序时将一个个结构块组合起来形成整体,整个程序结构层次分明,在需要修改程序时,可以将某一基本结构单独提出来进行修改,而不致影响到其他的结构。

## 3.2　顺序结构

顺序结构的程序严格按照程序中语句的出现次序来执行,其执行流程就像瀑布一样飞流直下。顺序结构是最基本的程序结构,可以说,所有的程序都会包含这种程序结构。

### 3.2.1　常用基本语句

#### 1. 赋值语句

赋值语句是在 Visual Basic 程序设计中使用最频繁的语句之一。指的是将一个数据赋给一个变量或者是赋给某一带属性的对象。在前面的例子中已经多次出现这样的语句。赋值语句的形式为:

```
变量名 = <表达式>
对象.属性 = <表达式>
```

功能为计算赋值号“=”右边表达式的值,并将结果赋值给左边的变量名或者指定对象的属性。例如:Form1.Width = 500, temp=temp+50。

**说明:**

(1) 赋值语句中的“=”不是数学中的符号,它是赋值符号,其作用是先计算右边表达式或变量的值,再赋给左边的变量或带有属性的对象,例如:

```
temp = 5                                    '将数值 5 赋值给变量 temp
temp = temp + 5                             '先计算右边表达式的值,再将其赋值给 temp(为 10)
```

另外,赋值号不满足等号的交换率,即 5=temp 是错误的赋值语句。

(2) 赋值号左边只能为变量或带有属性的对象,而不能是其他量,如常量或表达式等。例如:

```
C = A + B                                   '将变量 A、B 的值的和赋值给变量 C
```

如果写成 A ＋ B ＝ C 就是错误的。下面的语句也是错误的：

```
10 = X                                    '赋值号左边是常量
sqr(X) = 20                               '赋值号左边包含了函数调用
```

(3) 在一个赋值语句里不能同时给多个变量赋值。例如：

```
x = y = z = 5                             '不能连续赋值,应拆开分别赋值
```

系统将最左边的“＝”看成赋值号，而将后边的“＝”作为关系运算处理。

(4) 赋值号两边的数据类型应当相同或只有当表达式是一种与变量兼容的数据类型时，该表达式的值才可以赋给变量，否则会出错。例如：当表达式是非数字字符串，左边变量为数值类型，则出错。

### 2. 注释语句

一个程序放了一段时间后再重新阅读难免会忘了某一段的功能，或者在分工编程时为了让其他人员阅读方便，经常需要在程序中使用注释。注释是不会被程序执行的，它仅仅对程序的某些部分进行说明，可放在过程、模块的开头作为标题，也可以放在执行语句的后面。注释语句的格式为：

```
Rem <注释内容>  或  '<注释内容>
```

**说明：**

(1) 注释内容包含任何注释文本。

(2) 注释语句作为一个独立行，可放在过程、模块的开头作为标题，也可以放在执行语句的后面。Rem 关键字通常用在行首，如果在其他语句行后面使用 Rem 关键字，必须用冒号(:)与语句隔开。例如：

```
Const PI = 3.1415926                      '声明符号常量 PI
S = 2 * PI * r                            : Rem 计算圆直径
```

### 3. With 语句

当对同一个对象或同一个自定义类型变量执行多个操作，如引用对象的多个属性、调用对象方法或改变自定义类型变量元素的值时，使用 With 语句可以不必重复指出对象名或自定义类型变量名，使代码更容易编写、阅读和更有效地运行。With 语句的语法为：

```
With 对象名或自定义类型变量名
    <语句块>
End With
```

例如：下面多条语句可以设置窗体的外观，程序中反复引用了对象名 Form1。

```
Form1.Width = 1000
Form1.Height = 2000
Form1.Caption = "学习语句"
Form1.Move 0, 0
```

使用 With 语句之后,程序段变为:

```
With Form1
    .Width = 1000                          '前面的"."不能省略
    .Height = 2000
    .Caption = "学习语句"
    .Move 0, 0
End With
```

虽然程序增加了两行,但变得更易阅读与调试。另外,需要注意一个 With 语句只能用于同一个对象或自定义类型变量。

#### 4. 结束语句

使用 End 语句可以结束程序的运行,使用形式为:

```
End
```

**说明:**

(1) 程序遇到 End 语句,应用程序结束运行,关闭以 Open 语句打开的文件,清除变量释放所占用的内存,返回操作系统或 Visual Basic 集成开发环境。

(2) End 语句只是强制终止代码,不调用 Unload、QueryUnload 或 Terminate 事件,以及任何其他的 Visual Basic 代码。

(3) End 语句还能同 VB 的一些关键字联合使用,用以结束一段程序块,包括以下情况。

End Sub:结束一个 Sub 过程。

End Function:结束一个 Function 过程。

End If:结束一个 If 语句块。

End Type:结束记录类型的定义。

End With:结束一个 With 语句。

End Select:结束情况语句。

#### 5. 暂停语句

Stop 语句是用来暂停程序执行的,其格式为:

```
Stop
```

**说明:**

Stop 语句的功能是把解释程序置为中断模式(Break),以便对程序进行检查和调试。Stop 语句可以放置在程序的任何地方,当执行 Stop 语句时,系统将自动打开立即窗口。

### 3.2.2 输入输出对话框

VB 应用程序的输入与输出,除了使用文本框的 Text 属性、标签 Caption 属性(作为输出)以及 Print 方法(在窗体上输出)等之外,VB 系统还提供了标准对话框作为应用程序的输入、输出数据的界面。

### 1. InputBox 函数

InputBox 函数产生一个对话框供用户输入信息，其使用格式如下：

```
变量名 = InputBox[ $ ](<提示信息>[,<标题>][,<默认值>][,<x 坐标>][,<y 坐标>])
```

该函数的功能是打开一个如图 3-4 所示的对话框，等待用户输入信息，当用户按回车键或者单击“确定”按钮时，函数将输入的内容作为字符串返回。

**说明：**

(1) 提示信息：为字符型参数，用来显示对话框中的提示信息，字符总长度不得超过 1024 个字符。如果内容太长，一行放不下，只能用程序方法换行，即将要显示的信息通过字符连接符“＋”或“&”组成字符串，在换行处加回车 Chr(13)和换行 Chr(10)控制符或使用系统常量 vbCrLf，如图 3-4 所示。


图 3-4　输入对话框

(2) 标题：为字符型参数，用来显示对话框的标题内容，如图 3-4 中的标题“输入对话框”。如果该项省略，则把应用程序名放入标题栏中。

(3) 默认值：为字符型参数，用来在数据输入区作为默认值显示。如同意此值可直接按回车键或单击“确定”按钮即可，如不同意则重新输入。如果无此参数，数据输入区显示空白，等待用户输入。如图 3-4 中此参数的值为 100。

(4) x 坐标：是数值型表达式，指定对话框的左边与屏幕左边的水平距离，如果该项省略，则对话框在水平方向上居中显示。

(5) y 坐标：是数值型表达式，指定对话框的上边与屏幕上边的垂直距离，如果该项省略，则对话框会被放置在垂直方向上距离下边大约 1/3 处。

(6) 各项参数次序要一一对应，除第一个参数是必需的外，其余参数都可以省略。如果处于中间位置的参数省略了，那么要求逗号不能省略，要保留。

例如：

```
Score = InputBox("请输入成绩", , 80)      '中间参数标题省略,但逗号保留
```

(7) InputBox 函数总会返回一个字符串，即使用户没有输入任何值，它也会返回一个长度为 0 的空字符串。所以，必须将该函数的值赋给一个变量，否则就会编译出错。

例如，图 3-4 所示的输入对话框是通过下面的语句来实现的：

```
Data$ = InputBox("欢迎使用本系统!" + Chr(13) + Chr(10) + _
    "请输入数据:", "输入对话框", 100)
```

**【例 3-5】** 利用 InputBox 函数输入一华氏温度，将其转换成摄氏温度。转换公式为：C＝5/9＊(32－F)。

**解题分析：**通过调用 InputBox 函数来输入华氏温度，利用转换公式计算摄氏温度，再通过 Print 方法将其显示在窗体上。

**操作步骤：**

(1) 启动 VB 环境，选择“标准 EXE”，单击“确定”按钮，进入 VB 主窗口。

(2) 在代码编辑器中,编写 Form 的 Click 事件代码,如图 3-5(a)所示。

(3) 按 F5 功能键或单击工具栏中的启动▶按钮,单击窗体,运行结果如图 3-5(b)所示。图 3-5(b)是调用 InputBox 函数后产生的输入华氏温度的输入对话框。输入相应信息,单击"确定"按钮,程序的运行结果如图 3-5(c)所示。


(a) 例3-5程序代码


(b) 调用InputBox函数产生输入对话框　　(c) 例3-5运行结果

图 3-5　例 3-5 程序代码及运行结果

### 2. MsgBox 函数和 MsgBox 语句

使用办公自动化软件时,当关闭一个未经保存的 Word 文档时经常会出现如图 3-6 所示的确认窗口,询问是不是要进行关闭操作。这是一个常见的消息对话框,用于显示一些程序执行的信息供用户选择,然后根据用户选择的结果作为程序继续执行的依据。在 Visual Basic 中可以通过 MsgBox 函数来调用标准信息对话框。


图 3-6　消息对话框

调用 MsgBox 函数的语法如下:

```
变量名 = MsgBox(<提示信息>[,<对话框样式>][,<标题>])
```

MsgBox 函数的功能是在对话框中显示信息,等待用户单击按钮,并返回一个整数告诉系统用户单击的是哪一个按钮。MsgBox 过程与 MsgBox 函数的功能相同,但它没有返回值,常常用于显示某些信息,而不作程序流程的选择控制。MsgBox 过程使用形式如下:

```
MsgBox <提示信息>[,<对话框样式>][,<标题>]
```

**说明:**

(1) 提示信息是显示在对话框上的提示信息,与 InputBox 函数中的对应参数相同,该项不能省略。

(2) 对话框样式是可选项,最多可由 4 项数值相加组成,用来说明在对话框中显示的按钮数目及形式、图标类型、默认按钮以及对话框的强制返回值。若省略此项,取其默认值 0。