第 3 章

VB. NET 程序设计基础

3.1 VB. NET 基础

ASP. NET 是一个程序设计环境(技术),要在这个环境中进行程序设计,除了使用 HTML 标记、ASP. NET 的对象外,还必须选择一种程序设计语言。. NET Framework 支持 VB 等非脚本的高级语言。

本章介绍一些 Visual Basic. NET 的初步知识。Visual Basic. NET 是继 Visual Basic 6. 0 之后新发布的 Visual Basic 的全新版本,这是一种简单易学、功能强大的开发环境,全面支持面向对象编程思想的程序设计语言。由于它语法简单,更接近于自然语言表示,所以有更多的使用者和应用产品;又由于它引入了类与对象等概念,建立在. NET 平台之上,所以具备更强的系统功能和可扩展性,因此也就成了 ASP. NET 所能支持的最常用的编程语言之一。使用 VB. NET 语言开发 ASP. NET 应用,具有简单、快速、高效等特点。

如果读者对 Visual Basic. NET 的编程有较多的认识和了解,可以跳过这一章,从第 4 章开始学习。

3.2 VB. NET 程序代码约定

3.2.1 注释和续行

1. 注释

注释是程序设计员阅读程序代码的说明,在注释符号后面的数据全部视为注释。注释并不会影响程序的执行。

在程序中加入注释可以让他人较容易阅读,也可以帮助自己在日后阅读程序时更容易回忆起来。VB. NET 有两种标注注释的方式,一种是使用单引号('),另一种则是使用 Rem 关键字。

例如:

```
<%@ Import Namespace = "System.Datetime" %>          '这是一个注释语句,引用系统命名空间
```

也可用 Rem 语句表示注释,语法形式如下:

```
Rem 这是一个注释语句
```

2. 续行

在编写程序时,可能遇到一行代码太长,甚至超过一个页面的宽度,这种冗长的程序代码在阅读程序时需要拉动滚动条,并不是很方便。VB.NET 使用下画线(_)作为续行字符,此时可以利用续行字符将太长的程序代码拆成比较容易阅读的方式。

例如:

```
Dim Mystring As String_
    = "ASP.NET 网络程序设计"
```

如果要在一行内显示多个语句,则在各个语句间用冒号分隔。例如:

```
A = A + 1: B = B + A: C = C + B: Rem 在一行内显示了 3 个语句
```

3.2.2 编码约定

当在程序设计中要使用变量时,要为变量取一个名称。变量的命名不但要容易理解,而且还要变量名称的一惯性,尤其是要使程序员或别人日后阅读或维护程序的时候容易理解。为变量取一个清楚且有意义的名称在程序设计中特别重要,变量的命名和其他对象一样必须遵循下列的命名规则。

(1) 变量必须以英文字母开头。

(2) 变量不能包含空格,但可包含数字、字母。

(3) 同一代码段内变量是唯一的。

(4) 变量不能包含标点或系统使用的类型定义字符,但可以包含下画线。

(5) 变量不能使用 VB.NET 的保留关键字或系统对象名称。

(6) 变量长度不能超过 255 个字符。

3.3 数据类型

计算机要处理各种不同类型的数据,如整数、小数和时间等,这些就称为数据类型。

VB.NET 主要有字节(Byte)、数值(Numeric)、字符串(String)、日期时间(Date)、布尔(Boolean)、对象(Object)等数据类型,当程序运行时,就会为不同类型的数据安排相应大小的内存空间。数据类型可以划分为数值类型、文本类型和其他类型三大类。表 3-1 列出了 VB.NET 的数值型数据类型。

用于存放文本的数据类型有两个。VB.NET 的文本型数据类型如表 3-2 所示。

需要说明如下几点。

(1) 对于 String 类型,可以存放任何形式的字符串,它可以是纯文本和数字的组合或者是数字、日期等。如“今天是 2009-12-20,天气真暖和”和“这本书的价格是 25.5 元”。对于字符串类型的数据,可以进行相关的字符串操作,如连接、截取、定位、查找等。

表 3-1　VB. NET 的数值型数据类型

数据类型	表示方式	取值范围	说　明
整型	Integer	－2147483648～2147483647	用于表示简单整数
字节型	Byte	0～255	用于简单算术运算。由于这个类型的变量可以在一个字节中存储，所以运算速度最快
短整型	Short	－32768～32767	是整型的一种形式，相对表示范围较小
长整型	Long	－9223372036854775808～9223372036854775807	是整型的一种形式，相对表示范围较大
单精度型	Single	－3.402823E38～－1.401298E－45(对于负数) 1.401298E－45～3.402823E38(对于正数)	用于存放单精度浮点数
双精度型	Double	－1.7986931 3486232E308～－4.94065645841247E－324(对于负数) 和 4.94065645841247E－324～1.79869313486232E308(对于正数)	用于存放双精度型浮点数
小数	Decimal	当小数位为 0 的时候，为： －79228162514264337593543950335～79228162514264337593543950335； 当小数位为 28 的时候，为： －7.9228162514264337593543950335～7.9228162514264337593543950335	常用于存储货币值

表 3-2　VB. NET 的文本型数据类型

数据类型	表示方式	说　明
字符串型	String	用于存放任何形式的字符串，包括一个字符或者多行字符
字符型	Char	用于存放一个字符

(2) 对于 Char 类型，可以存储的只是一个字符。其余的数据类型还有：Date 数据类型、布尔数据类型和 Object 数据类型。表 3-3 列出了 VB. NET 的其他数据类型。

表 3-3　VB. NET 的其他数据类型

数据类型	表示方式	说　明
日期型	Date	必须用 mm/dd/yyyy 的格式表示，也可以存储时间(可以存储 00:00:00～23:59:59 之间的任何时间)
布尔型	Boolean	取值为 true 和 false
对象型	Object	

当 Boolean 类型数据转换为数值类型时，会把 true 当成 1 来处理，把 false 当成 0 来处理。

3.3.1　VB. NET 常量

常量拥有固定的数值，它可以代表字符串、数字和日期。常量一经声明，其值将不能再更改。声明常量的意义就在于可以在程序的任何部分用该常量来代表特定的数值，从

而方便了编程。例如,在计算程序中常用 PI 来表示 3.1415926,这样既不容易出错,也会使程序更加简洁。

1. 声明常量

可以使用 Const 语句,例如:

```
Const PI As Single = 3.1415926                    '表示声明的单精度常量 PI
Const M_name As String = "Tom"                    '字符串型常量用""
Const m_date As Date = #2005-10-29#               '声明的日期型常量用##
```

声明过 PI 这个常量,在程序的其他地方就可以用 PI 来表示 3.1415926 了。例如:

```
Const PI As Single = 3.1415926
S = PI * R^2          '计算半径为 R 的圆的面积
```

2. 常量的命名规则

通常情况下,常量命名规则如下。

(1) 可以使用字母、数字、下画线等字符。

(2) 不可以使用空格、斜杠、逗号、句号、加减号等特殊字符。

(3) 第一个字母必须是英文字母。

(4) 长度不能超过 255 个字符。

(5) 不能使用 VB.NET 中的关键字。所谓关键字,就是 Dim、Sub、End 等 VB.NET 使用的一些特殊字符串。

(6) 在编程时要养成一个良好的习惯,采用一个科学的命名规则。

① 可以将数据类型的缩写作为常量名的前缀,这样一看就知道这是个常量。

② 命名不仅要让自己以后能看明白,而且要让别人能看明白,所以使用有意义的名称,但不要太长。

③ 使用字母的大小写或下画线使常量更容易理解,如 ConuNumber 或 conu_number。

VB.NET 常量根据作用域的不同也可分为过程常量(或局部常量)和全局常量。常量的作用域由声明它的位置决定。如果是在一个子过程或函数里声明的常量,则只在该过程里有效。

图 3-1 页面 3_1.aspx 的运行结果

【例 3-1】 声明常量的作用域,第一个常量 m_string 是全局常量,第二个常量 n_string 是局部常量,页面运行结果如图 3-1 所示。

源程序(aspnetjc(vb)/WebSites/WebSite1/ch03/3_1.aspx):

```
<%@ Page Language = "VB" CodeFile = "3_1.aspx.vb" Inherits = "ch03_3_1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns = "http://www.w3.org/1999/xhtml" >
<head runat = "server">
```

```
    <title>常量的定义与应用</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="message1" runat="server" Width="320px"></asp:Label>
        <p><asp:Label ID="message2" runat="server" Width="320px"></asp:Label></p>
    </div>
    </form>
</body>
</html>
```

源程序(aspnetjc(vb)/WebSites/WebSite1/ch03/3_1.aspx.vb)：

```
Partial Class ch03_3_1
    Inherits System.Web.UI.Page
    Const m_string As String = "第一个常量是全局常量 m_string"
    Sub page_load(ByVal obj As Object, ByVal e As EventArgs)
        Const n_string As String = "第二个常量是局部常量 n_string"
        message1.Text = m_string
        message2.Text = n_string
    End Sub
End Class
```

程序注释：

(1) 常量 m_string 是全局常量，常量 n_string 是局部常量，只能在 Page_Load 过程中有作用。

(2) 不同过程中可以使用相同的常量名称，其值和作用只在本过程中有效。

(3) 若全局常量名和某一过程的常量名相同，在该过程中的常量起作用。

3. 使用常量的注意事项

在 VB.NET 中，使用常量时注意以下几点。

(1) Nothing：在 VB.NET 中，把一个表示对象的变量赋值为 Nothing 时，就表示这个对象不再使用，VB.NET 会释放这个对象所占用的内存空间。

例如：

```
My_object = Nothing
```

(2) Null：当一个变量的值是 Null 时，它表示这个变量的值不是有效数据。如果没有给一个变量赋任何值，VB.NET 会赋给它一个初始值，如果用户定义了一个整型的变量，那么在没有使用它之前，它的值是 0，而 Null 则表示这个变量的值是一个无效值。

(3) true：表示真。

(4) false：表示假。

true 和 false 是布尔值。

在定义常量的时候使用 Const 关键词，常量出现在程序中时，它的值在程序的执行过程中是不能改变的。

为了提高程序的运行效率,建议用户不要定义不需要使用的常量,因为所有的常量都要占用内存空间。一旦定义了一个常量,系统就要在它的整个生存周期内负责维护这个常量。对于大型的程序,往往会定义一个常量文件,把项目使用的所有常量都定义在这个文件中,在需要使用的时候把这个文件包含进来。这种方法虽然对于降低代码的复杂度很好,但是,某一个程序并不会使用包含在文件中的所有常量。因此,很多常量是没有用的,但是这些常量却仍然要占用服务器的内存空间。由于在网络的环境中,客户的需求不好估计,往往很大,所以在定义常量时要考虑清楚,保证程序的运行效率。

3.3.2 VB. NET 变量

变量用来存储程序中需要处理的数据,在所有的程序设计语言中,几乎都要求程序设计人员在使用变量以前定义数据类型,因为不同数据类型的变量所需要的内存空间不一样,如字节型的变量需要 8 位空间,短整型变量需要 16 位空间等,所以为一种数据定义的变量就不能存放另一种数据类型的值。

变量的值在定义后可以随时改变。创建变量的方式有两种:一种是显式定义的方法;另一种是使用隐式定义的方法,也就是在用的时候直接写出这个变量并为它赋值。

定义变量可以使用 Dim 语句或 Private 语句,在过程中定义变量必须用 Dim 语句。

例如:

```
Dim m_x As Integer                          '定义一个整型变量
Dim m_book As String , m_today As Date      '定义多个变量
Dim m_id, m_pwd                             '定义了两个对象型的变量,没有定义类型
M_id = 'peter'                              '给对象 M_id 赋值 String 型
M_pwd = 86912828                            '给对象 M_pwd 赋值 Integer 型
```

注意:

VB. NET 和其他语言一样,变量名称必须以字母开头,只能包含字母、数字和下画线,并且不是 VB. NET 的关键字。在为变量取名时,建议不要使用像 a 或者 b 这样的让人无法理解的变量名,应该采用小写前缀加上有特定描述意义的名字作为变量命名,这种命名方法被称为 Hungarian 法,如表 3-4 所示。变量名的前 3 个字母用于说明数据类型,第四个字母大写以表示变量的实际含义。

表 3-4 Hungarian 法变量前缀

数据类型	前 缀	示 例	数据类型	前 缀	示 例
Boolean	bln	blnYes	Integer	int	intTotal
Byte	byt	bytByte	Long	lng	lngLong
Char	chr	chrChar	Single	sng	sngSingle
Date	dat	datDate	Short	sho	shoShort
Double	dbl	dblDouble	String	str	strString
Decimal	dec	decDecimal	Object	obj	objObject

例如：

```
Dim chrBook As String
Dim intTotal As Integer
```

对于程序员来说，这种命名约定并不是强制性的，仍然可以使用 a 和 b 这样的变量名，也可以用 intTotal 变量来存储一个字符串。很显然，如果程序员都采用 Hungarian 法这种为变量命名的方式并在程序设计过程中遵循这样的好习惯，可以使得程序易读和易维护，并减少出错的机会。

变量名的命名应使变量的用途明确，使每个变量的数据类型和可见范围清晰明了，使代码中的过程易于理解，使程序易于调试，使变量的存储和处理更为有效。

3.4　运算符与表达式

程序中对数据的加工是通过运算符完成的，如两数的加、减操作等，对数据的复杂加工要通过语句组成一段程序才能完成，如一组数据的排序等。

在 VB. NET 中，常用的运算符与其他的语言并没有大的不同。常用的运算符有赋值运算符、算术运算符、字符串连接运算符、比较运算符和逻辑运算符等。

表达式是一个或多个运算符的组合。VB. NET 的表达式与其他语言的表达式没有显著的区别。每个符合 VB. NET 规则的表达式的计算都是一个确定的值，对于常量、变量的运算和对于函数的调用都可以构成最简单的表达式。根据运算符的不同，有各种类型的表达式，表达式总是有确定的值的，根据运算符的优先级进行计算。

1. 赋值运算符

赋值运算的作用是执行一次赋值运算，将右边表达式的值送入等号左边的变量。

赋值表达式：变量名 ＝ 表达式。

赋值表达式功能是先计算表达式的值，再将计算结果送入变量。表达式又可以是赋值表达式。赋值表达式本身是一个运算表达式，它也有值，其值就是给左边变量赋的值。

赋值运算符是最常用的运算符，它就是熟悉的等号(＝)。需要注意的是，虽然它表面上是一个等号，但它并不是一般数学意义上的等号，它的意思是把等号右边的值送入等号左边的变量。

例如：

```
Dim intMumber as Integer
intNumber = 10
intNumber = intNumber * 2
```

intNumber 的值是 20。

2. 算术运算符

VB. NET 中的算术运算符有＋(加)、－(减)、＊(乘)、/(除)、\(整数除)、Mod(取模)和^(幂)等。

需要说明的是，/(除)和\(整数除)有一定的区别。/(除)表示的是通常意义的除法，

如(4.5/3)的结果是 1.5,而\(整数除)表示把除数和被除数四舍五入以后再计算除法得到整数结果,在计算(4.5\3)时,把 4.5 四舍五入为 5,再进行运算,得到的整数结果是 1,这种运算在特定的应用中会十分有用。例如,有 107 条消息需要发布,又不希望所有的 107 条信息都在一页中显示,而希望分页显示,而每页中只能够显示 20 条信息。那么(107\20)而得到的 5,就表示有 5 页是填满了 20 条信息的,而剩下的一页中有 7 条信息。

3. 比较运算符

VB.NET 中的比较运算符有 =(等于)、<>(不等于)、<(大于)、<=(小于等于)、>(大于)、>=(大于等于)。这些运算符对于数值、字符、日期表达式的比较都是有效的。结果是布尔类型的 true 或 false。

需要说明的是,字符串之间也是可以进行比较的,比较的方法是根据字母表的排列顺序。例如,“but”比“book”大,而“A”小于“a”。

4. 字符串连接运算符

VB.NET 中对于两个字符串类型的变量,使用“+”或“&”运算符做字符串的连接运算。但是这时运算符“+”的含义不是加法,而是字符串的连接。

例如:

```
Dim strBook,strName,strBookname as String        '定义变量为字符型
StrBook = "ASP.NET"
strName = "网络程序设计"
strBookname =  StrBook  + strName                '结果是“ASP.NET 网络程序设计”
'strBookname =  StrBook &strName                 '结果是“ASP.NET 网络程序设计”
```

5. 逻辑运算符

VB.NET 中常用的逻辑运算符有 Not(非)、And(与)、Or(或)、Xor(异或)。运算的结果仍然是布尔类型的 true 或 false。

例如:

```
Dim bollogc1,bollogc2, bollogc As Bollean        '定义变量为布尔型
bollogc1 = true
bollogc2 = false
bollogc =  bollogc1 And bollogc2                 'true And false 是 false
bollogc =  bollogc1 Or bollogc2                  'true Or false 是 true
bollogc =  bollogc1 Xor bollogc2                 'true Or false 是 true
bollogc =  Not bollogc1                          'Not true 是 false
```

6. 运算符的优先级

VB.NET 优先级高的运算先执行,若优先级一样,则先执行左边的运算。不过要记忆这些运算级的顺序似乎不很容易,所以可以使用小括号来强制指定运算顺序的优先级。VB.NET 遇到小括号先执行,倘若小括号中还有小括号,则最内层的小括号先执行。例如(15×(1+4))/(7-2),其运算顺序为先执行 1+4,得到 5 后再乘以 15,得到 75 后再除以(7-2),最后就得到正确的答案 15。运算符的优先级如表 3-5 所示。

表 3-5　运算符的优先级

优　先　级	运　算　符	名　　称	范　　例	结　　果
高	()	括号	A＝(8＋5) * 3	A＝39
↓	^	幂	B＝5^2＋10	B＝35
↓	*	乘	C＝12＋3 * 2	C＝18
↓	/	除	D＝35－9/3	D＝32
↓	\	整数除法	D＝11\3＋5	D＝8
↓	Mod	余数	E＝11 Mod 3	E＝2
低	＋ －	加减法	F＝18＋4－10	F＝12

7. VB. NET 新增的运算符

VB. NET 支持下列新的运算符，新增的运算符如表 3-6 所示。

表 3-6　新增的运算符

运　算　符	范　　例	说　　明
＋＝	shtA ＋＝ 1	等于 shtA ＝ shtA ＋ 1
－＝	shtB －＝ 1	等于 shtB ＝ shtB － 1
* ＝	shtC * ＝ 2	等于 shtC ＝ shtC * 2
/＝	shtD /＝ 2	等于 shtD ＝ shtD / 2
\＝	shtE \＝ 3	等于 shtE ＝ shtE \ 3
^＝	shtF ^＝ 2	等于 shtF ＝ shtF ^ 2
&＝	strG &＝ strH	等于 strG ＝ strG & strH

3.5　数组

进行数据处理时，经常碰到要求处理一批同类型相关数据，如数据的排序、矩阵运算等。如果按照前面所学的知识，用简单变量来处理，一个变量只能存放一个数据，这样不仅程序繁复、冗长，体现不出各数据间的关系。数组是一种可以存储多个数据的特殊数据结构，这个结构由相同数据类型的元素组合而成。当想要将一些数据类型相同的数据用一个变量来管理的时候，数组是一个非常方便的结构。

数组是一批同类型相关数据的有序集合，每一个数据称为元素，这些元素有一个共同的名字即数组名，不同元素由其在数组中的序号即下标(从 0 开始编号)来标识。

1. 一维数组

与普通变量一样，在使用数组前要对它加以定义。数组定义的主要目的是确定数组的名称、确定数组的大小、确定数组的类型。由一个下标确定元素的数组称为一维数组。

下面介绍一维数组的定义及赋值，例如：

```
Dim arrNumber(3) As Integer
arrNumber(0) = 200
arrNumber(1) = 5
```

```
arrNumber(2) = 2
arrNumber(3) = arrNumber(0)/arrNumber(1) + arrNumber(2)
```

在上面的 Dim 语句中定义了一个一维数组 arrNumber,有 4 个元素 arrNumber(0)、arrNumber(1)、arrNumber(2)、arrNumber(3),从例子可以看出数组的下标是从 0 开始的,其类型是整型。

2. 多维数组

最常见的是二维数组和三维数组。二维数组是“数组的数组”,它是最简单的多维数组,数据有行列之分,用两个下标来标识一个数组元素。

下面介绍二维数组的定义,例如:

```
Dim arrNumber(1,2) As Integer                    '定义了一个 2×3 的数组
```

arrNumber(1,2)数组的下标范围第一维也称为行,从 0 到 1 有 2 行,第二维也称为列,是从 0 到 2 有 3 列,元素从 arrNumber(0,0)到 arrNumber(1,2)共 6 个单元。

VB. NET 中的数组维数最多可以定义到 60。

3. 动态数组

在 VB. NET 中还可以定义动态数组,即长度不确定的数组,可以在程序开始的时候定义一个动态数组。例如:

```
Dim arrData()As Integer
```

在上面的定义中,既没有指定数组的长度,也不知道它的维数,因此这个数组在程序中是不能直接使用的。因为所有的数组都需要有连续的内存空间存放信息,如果没有定义数组的维数和每一维的长度,就意味着无法为数组分配内存空间,所以 VB. NET 要求在使用动态数组之前用 ReDim 语句为这个数组指定维数和每一维的长度。

例如:

```
ReDim arrData(5)
```

如果一个数组在定义的时候不是动态数组,就不能使用 ReDim 语句对数组重新定义。另外,对于已使用 ReDim 语句重定义过的动态数组,可以使用 ReDim 语句重新再定义这个数组的长度,但是不能使用 ReDim 语句来重新定义数组的维数。也就是说,如果已经定义了一个动态数组 Dim arrData(),可以在后面的程序中把它重新定义为一个一维的数组 ReDim arrData(5),而不容许再把它定义为一个二维的数组 ReDim arrData(3,5)。

4. 对象型数组

前面介绍的数组元素的数据类型必须要和在定义数组时所指定的数据类型一样。VB. NET 的数组数据类型有对象(Object)型,这个数组可以存储各种不同类型的数据。

例如:

```
Dim objStudent(3) As Object
objStudent(0) = "Lin Ping"                       '姓名
objStudent(1) = "hangzhou"                       '出生地
objStudent(2) = 35                               '年龄
```

```
objStudent(3) = #10/03/1970#                    '生日
```

这个示例定义了一个名为 objStudent 并可存储 4 个元素的对象型数组，分别把姓名、出生地、年龄以及生日填入数组中。

多维数组也可以定义为对象型数组，如 Dim objStudent(3,4) As Object，可以存储 4 个人的信息。

3.6　控制语句

程序由 3 种基本程序结构组成，即顺序结构、选择结构和循环结构。

(1) 顺序结构是最基本、最简单的程序结构，它由若干语句块组成，按照各块的语句排列顺序依次执行，如图 3-2(a)所示。

(2) 选择结构又称分支结构，是根据给定的条件，从两条或者多条路径中选择下一步要执行的操作路径，如图 3-2(b)所示。图中表达式是给定的条件，当条件成立时，选择语句组 1 操作，否则选择语句组 2 操作。

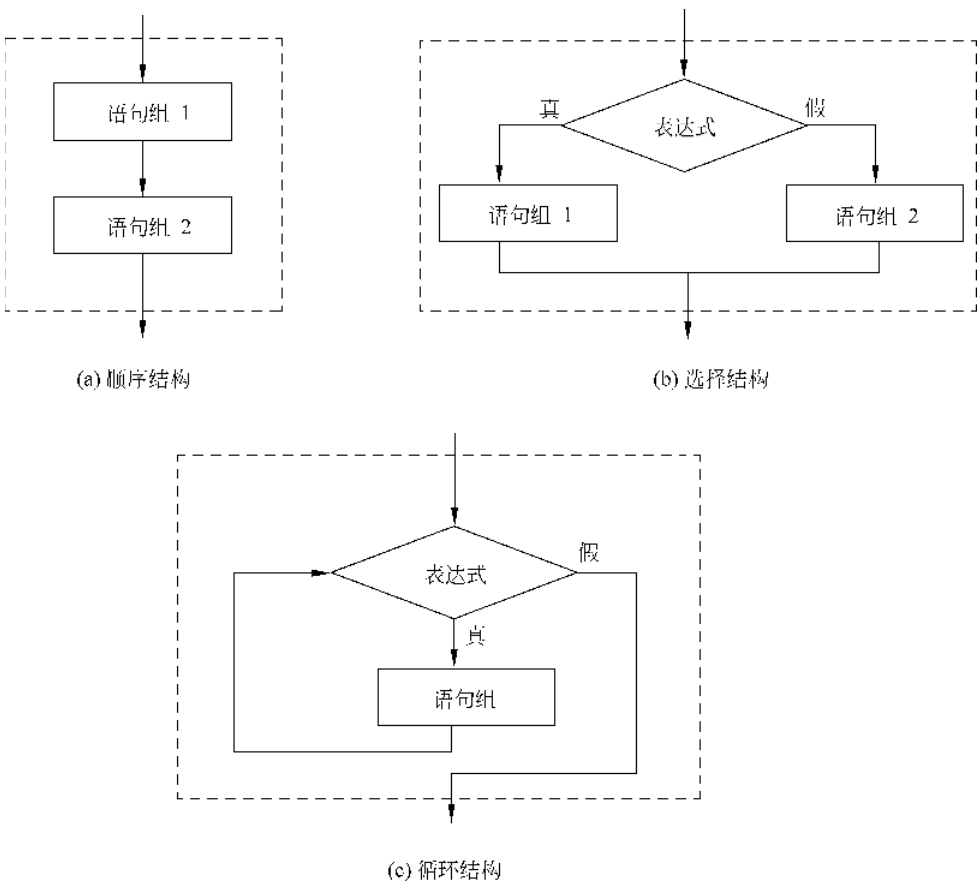

图 3-2　程序的 3 种基本结构