第 2 章　Visual Basic. NET 编程基础

在用 Visual Basic. NET 进行基本的编程之前，还需要了解它的一些基础知识，如数据类型、常量和变量的概念及用法、运算符、表达式、内部函数和程序语句的使用等。

2.1　数 据 类 型

程序在执行过程中，会对一些原始数据进行处理和运算，最后产生计算结果。编程语言中的数据是各式各样的，不同类型的数据在计算机中具有不同的表示方法，占有不同的存储空间。Visual Basic. NET 定义了多种数据类型，用户在编写对数据处理的程序时，可选择相应的数据类型表示数据。

Visual Basic. NET 的数据类型可分为基本数据类型和自定义数据类型。

2.1.1　基本数据类型

Visual Basic. NET 的基本数据类型可分为以下 5 类：数值数据类型、字符数据类型、布尔数据类型、日期数据类型和对象数据类型，如表 2-1 所示。每种标准数据类型都拥有一个固定名字，在程序中可以直接使用这些标准数据类型。

表 2-1　Visual Basic. NET 的基本数据类型

数据类型		关 键 字	占用内存（字节）	取值范围
数值数据类型	字节型	Byte	1	0～255
	符号字节型	SByte	1	－128～127
	短整型	Short	2	－32 768～32 767
	整型	Integer	4	－2 147 483 648～2 147 483 647
	长整型	Long	8	－9 223 372 036 854 775 808～9 223 372 036 854 775 807
	无符号短整型	UShort	2	0～65 535
	无符号整型	UInteger	4	0～4 294 967 295
	无符号长整型	ULong	8	0～18 446 744 073 709 551 615
	小数型	Decimal	16	整数：－79 228 162 514 264 337 593 543 950 335～＋79 228 162 514 264 337 593 543 950 335 小数：－7. 922 816 251 426 433 759 354 395 033 5～＋7. 922 816 251 426 433 759 354 395 033 5（28 位小数）

续表

数据类型		关 键 字	占用内存(字节)	取值范围
数值数据类型	单精度浮点型	Single	4	负数范围:-3.402 823E38～-1.401 298E-45 正数范围:1.401 298E-45～3.402 823E38
	双精度浮点型	Double	8	负数范围:-1.797 693 134 862 32E308～-4.940 656 458 412 47E-324 正数范围:4.940 656 458 412 47E-324～1.797 693 134 862 32E308
字符数据类型	字符型	Char	2	0～65 535
	字符串型	String	根据实际情况	0～2^{31}个 Unicode 字符
布尔数据类型		Boolean	2	True(-1 或非 0 值)或 False(0)
日期数据类型		Date	8	0001 年 1 月 1 日 0:00:00 到 9999 年 12 月 31 日 23:59:59
对象数据类型		Object	4	可存储任何类型的数据

1.数值数据类型

数值数据类型包括字节型(Byte)、短整型(Short)、整型(Integer)、长整型(Long)、小数型(Decimal)、单精度浮点型(Single)、双精度浮点型(Double)等,下面详细介绍。

1)字节型

字节型表示的是无符号整型数据类型,一个字节型数据占用一个字节的存储空间,用于存储 0～255 的整数。例如,将 100 赋值给字节型变量 BytVar:

```
Dim BytVar As Byte
BytVar=100
```

2)短整型

短整型用于存储-32 768～32 767 的带符号整数,占用两个字节的存储空间。例如,将-2 000 赋值给短整型变量 ShoVar:

```
Dim ShoVar As Short
ShoVar=-2000
```

与短整型相比,无符号短整型用于存储 0～65 535 的整数,也占用两个字节的存储空间,用法和短整型相似。

3)整型

整型用于存储-2 147 483 648～2 147 483 647 的带符号整数,占用 4 个字节的存储空间。例如,将 220 000 赋值给整型变量 IntVar:

```
Dim IntVar As Integer
IntVar=220000
```

与整型相比,无符号整型用于存储 0～4 294 967 295 的整数,也占用 4 个字节的存储空间,用法和整型相似。

4)长整型

长整型以 8 个字节的存储空间来存储带符号的整数,可存储-9 223 372 036 854 775 808～9 223 372 036 854 775 807 的整数,存储范围比整型大,可根据数据大小选择合适的数据类

型。例如，将－13 450 000 000 赋值给长整型变量 LogVar：

```
Dim LogVar As Long
LogVar＝－13450000000
```

与长整型相比，无符号长整型用于存储 0～18 446 744 073 709 551 615 的整数，占用 8 个字节的存储空间。

5)小数型

小数型用 16 个字节来存储小数，属于非整型数据类型。当小数位数为 0，可存储的整数范围为－79 228 162 514 264 337 593 543 950 335～＋79 228 162 514 264 337 593 543 950 335，当存储小数时，可存储 28 位小数，存储的小数范围为－7.922 816 251 426 433 759 354 395 033 5～＋7.922 816 251 426 433 759 354 395 033 5。例如，将 0.000 01 赋值给小数型变量 DecVar：

```
Dim DecVar As Decimal
DecVar＝0.00001
```

小数型比较适合财务类数据的计算，即需要记录的数的位数很多，但又不允许出现四舍五入。

6)单精度浮点型

单精度浮点型以 4 个字节(32 位)来存储单精度浮点数，其中符号占 1 位，指数占 8 位，其余 23 位表示尾数。单精度浮点型存储的负数范围为－3.402 823E38～－1.401 298E－45，正数范围为 1.401 298E－45～3.402 823E38。例如，将－1.4E5 赋值给单精度浮点型变量 SngVar：

```
Dim SngVar As Single
SngVar＝－1.4E5
```

7)双精度浮点型

双精度浮点型以 8 个字节(64 位)来存储双精度浮点数，其中符号占 1 位，指数占 11 位，其余 52 位表示尾数。双精度浮点型比单精度浮点型存储的数据范围更广，其存储的负数范围为－1.797 693 134 862 32E308～－4.940 656 458 412 47E－324，存储的正数取值范围为 4.940 656 458 412 47E－324～1.797 693 134 862 32E308。例如，将 1.400 000 1E100 赋值给双精度浮点型变量 DblVar：

```
Dim DblVar As Double
DblVar＝1.4000001E100
```

2.字符数据类型

字符数据类型包括字符型(Char)和字符串型(String)两种数据类型。

1)字符型

字符型数据类型是无符号的单个双字节(16 位)Unicode 字符，以 16 位无符号的数值形式存储。其中 Unicode 是一种以数字形式表示符号的字符编码规范，是一种重要的交互和显示的通用字符编码标准。

字符型数据类型一般用于存储单个字符，例如，将字符"B"赋值给字符型变量 CharVar：

```
Dim CharVar As Char
CharVar＝"B"
```

如果想在字符型数据类型和数值数据类型之间转换，可以通过 AscW 和 ChrW 函数

实现。

2)字符串型

字符串型数据类型是 0 个或多个 Unicode 字符的序列。一个字符串可以存储 0～2^{31} 个 Unicode 字符。

字符串类型的数据放在一对西文双引号里面,其中长度为 0(不含任何字符)的字符串称为空字符串。下面举例说明:

```
Dim StrVar1 As String
Dim StrVar2 As String
Dim StrVar3 As String
StrVar1="Visual Basic.NET"
StrVar2=""
StrVar3="计算机"
```

上面程序中将字符串"Visual Basic.NET"赋值给字符串变量 StrVar1,将空字符串赋值给字符串变量 StrVar2,将字符串"计算机"赋值给字符串变量 StrVar3。

3.布尔数据类型

布尔型(Boolean)数据占用两个字节的存储空间来存储逻辑值,逻辑值为 True 或 False。如果将逻辑值转换成数值,则 True 转换成－1,False 转换成 0。如果将数值转换成布尔值,则 0 转换成 False,非 0 值转换成 True。

一般用布尔数据类型记录两个状态,布尔变量的取值不是 True 就是 False,在程序设计中起到判断条件的作用。下面程序根据布尔变量 BolVar 的取值决定字符变量 CharVar 的赋值:

```
Dim CharVar As Char
Dim BolVar As Boolean
BolVar=False
If BolVar=False Then
  CharVar="A"
Else
  CharVar="B"
End If
```

上面程序 CharVar 的值为字符"A"。

4.日期数据类型

日期型(Date)数据以 8 个字节双精度数值的形式进行存储,存储时整数部分表示日期,小数部分表示时间,变量声明为日期型之后可以保存日期和时间。

Visual Basic.NET 日期数据类型以 mm/dd/yyyy(月/日/年)的格式定义,以“＃”作为定界符括起来,如 2010 年 3 月 1 日,表示为＃03/01/2010＃。如果存储的是时间数据,必须以 hh:mm:ss(小时:分钟:秒)的格式定义,如＃18:55:06＃。如果同时存储日期和时间数据,必须以“mm/dd/yyyy hh:mm:ss”(月/日/年 小时:分钟:秒)格式定义。日期型表示的数据范围为 0001 年 1 月 1 日 00:00:00 到 9999 年 12 月 31 日 23:59:59。下面举例说明日期数据类型的用法:

```
Dim DatVar As Date
DatVar＝＃03/01/2010＃
DatVar＝＃18:55:06＃
DatVar＝＃03/01/2010 18:55:06＃
DatVar＝Now()
```

以上都是有效的赋值语句，其中 Now()函数返回当前日期和时间。

5. 对象数据类型

对象型(Object)数据以 4 个字节的形式存储对象引用，使用时可以为对象型变量分配其他数据类型的数据。对象型和 Visual Basic 6.0 里面的变体型有点相似，但在 Visual Basic. NET 中不允许使用变体型变量，为了能够处理任何类型的变量，所有变量都需要先声明为对象型变量。下面例子说明对象数据类型的用法：

```
Dim ObjVar As Object
ObjVar＝500
ObjVar＝"ABC"
ObjVar＝＃03/01/2010＃
```

2.1.2　自定义数据类型

除了 2.1.1 中介绍的基本数据类型之外，Visual Basic. NET 还包括一些自定义数据类型，如枚举类型、结构类型等，将在 2.7～2.10 节详细介绍。

2.2　常量与变量

在程序中需要处理各种类型的数据，这些类型的数据，有的数据值在程序中是可以变化的，有的值是常数，在程序运行过程中不可改变。根据值可变不可变进行分类，可把数据分为常量和变量。

2.2.1　常量

常量也叫常数，是指在程序运行过程中其值保持不变的数据，常量可以是任何数据类型。在 Visual Basic. NET 中常量分为一般常量和符号常量。

1. 一般常量

一般常量指的是在程序代码里直接出现的各种类型的数据，常量的数据类型由书写格式自动区分，不需要声明和定义。一般常量包括数值常量、字符常量、布尔常量和日期常量。

1)数值常量

数值常量包括整数常量和浮点数常量两类。

(1)整数常量主要包括整型、短整型、长整型、无符号整型、无符号短整型、无符号长整型、字节型和符号字节型。整数常量使用的时候大多数都是使用十进制数，用户也可以根据需要采用八进制、十六进制的形式书写和表示。

• 十进制形式：如 100、－256、0。

• 八进制形式:使用前缀 &O(字母 O,而不是数字 0)表示,如 &O245。

• 十六进制形式:使用前缀 &H 表示,如 &H123、&H1A。

(2)浮点数常量包括单精度浮点数常量和双精度浮点数常量。浮点数可以用小数形式和指数形式表示,当用指数形式书写时,由尾数、指数符号、指数 3 部分组成。单精度浮点数的指数符号是 E(e),双精度浮点数的指数符号是 D(d)。例如,123.45E3(123.45e+3)是单精度浮点数,相当于 123.45×10^3,123.456 78D3(123.456 78d+3)为双精度浮点数,相当于 $123.456\ 78\times10^3$。其中,123.45 或 123.456 78 是尾数部分,E3、D3 是指数部分,E 和 D 是指数符号。

在 Visual Basic. NET 中,输入的数值常数的数据格式决定它的数据类型。默认情况下,把整数常量作为整型数据类型处理,把小数常量作为双精度浮点型数据类型处理。

常量的数据类型可以显式地指定,在 Visual Basic. NET 中通过提供的值类型字符来指定常量的数据类型,方法是在常量的后面加上类型字符。表 2-2 列出了值类型字符及相应用法。

表 2-2 值类型字符及用法

值类型字符	数据类型	示 例
C	Char	Var1="A"C
S	Short	Var1=123S
I	Integer	Var1=123I
L	Long	Var1=123L
D	Decimal	Var1=123D
F	Single	Var1=123F
R	Double	Var1=123R
US	UShort	Var1=123US
UI	UInteger	Var1=123UI
UL	ULong	Var1=123UL

2)字符常量

字符常量可分为字符型常量和字符串型常量,是由西文双引号引起来的一个或一串字符,其中的字符可以是 ASCII 码字符、汉字和其他可以打印的字符,而字符或字符串常量两端的双引号不是字符常量的一部分,只起到定界的作用。如"abc"、"张三"、"A"都是合法的字符常量。

3)布尔常量

布尔常量只有真和假两个值,在 Visual Basic. NET 中真值用 True 表示,假值用 False 表示。

4)日期常量

日期常量在书写时用定界符"#"把表示日期和时间的值括起来,如#03/01/2010#、#03/01/2010 18:55:06#、#January 1,2001#都是正确的日期常量。

2. 符号常量

在 Visual Basic. NET 中可以定义一个符号来代表一个常量,这就是符号常量。

定义符号常量的语法格式如下：

```
[Public|Private] Const 常量名 [As 类型]=表达式 [,常量名 [As 类型]=表达式]…
```

参数说明如下：

- Public 或 Private：可选项。Public 表示所定义的符号常量在整个项目中都有效；Private 表示所定义的符号常量只在当前所声明的模块中有效。如果缺省 Public 和 Private，表示所定义的符号常量在当前过程中有效。
- 常量名：必选项。它代表所定义的常量的名称，命名规则必须符合 Visual Basic. NET 的标识符命名规则。
- As 类型：可选项。类型可以是 Visual Basic. NET 所支持的任何一种数据类型，每个常量都必须用单独的 As 子句，若省略该项，则默认为对象数据类型。如下语句表示 Pi 为单精度类型的符号常量：

  ```
  Const Pi As Single=3.1415
  ```

可以在符号常量名的后面加上类型说明符来表示符号常量的类型，也可以在表达式值的后面加上值类型说明符来表示符号常量的类型，如下语句也是表示 Pi 为单精度类型的符号常量：

```
Const Pi!=3.1415        '! 为类型说明符，2.2.2 中详细介绍
Const Pi=3.1415F
```

- 表达式：必选项。表达式由字符常量、算术运算符、逻辑运算符等组成。

需要说明的是，如果在一行中同时定义多个符号常量，则它们之间要用逗号进行分隔。例如：

```
Const Pi As Single=3.1415,A As Double=1.2345
```

符号常量代表一个指定的数值或字符串，在定义的有效范围内可多次重复使用，这样大大简化了程序，对用户来说非常方便。

2.2.2　变量

变量是指在程序运行过程中其值可变的量。变量具有名称和数据类型，实际上变量代表了内存中的一块存储空间，通过变量名可以访问该空间里存放的数据，该存储空间里的数据即为给变量赋的值，变量名代表了存储空间的地址，变量的数据类型决定了该变量存储数据的方式。

1. 变量的命名规则

为了有效地在程序中使用变量，变量的命名必须遵循下面的规则：

(1)变量名只能由字母、数字、下划线“_”或汉字组成，不能包含空格、句号以及其他各种符号。例如，“A%B”、“A. B”、“H ow”都是不符合规则的变量名。

(2)变量名必须以字母、汉字或下划线“_”开头，如果以下划线“_”开头，至少还应包含一个合法字符；变量名长度不能超过 1 023 个字符；所有字母不区分大小写。

(3)变量名不能和 Visual Basic. NET 的关键字同名。例如，If、Loop、Abs、Mod 等都是关键字，不能作为变量名。另外，变量名也不能是末尾带有类型说明符的关键字，如 Double# 不能作为变量名。

(4)同一范围内的变量名必须是唯一的。

实际上，在 Visual Basic. NET 中过程名、数组名、符号常量名、结构类型名都必须遵循上面的规则。

2. 变量的声明

变量具有两个方面的特性，即名称和数据类型，因此声明变量也需要注意这两个方面。任何变量都具有一定的数据类型，用户可以通过声明来定义变量的数据类型。变量的声明分为“显式声明”和“隐式声明”。

1)显式声明

显式声明是在使用变量之前，先用声明语句声明变量。显式声明的语法格式如下：

声明关键字 ＜*变量名 1*＞ [As ＜*类型 1*＞][＝*初始值 1*][,＜*变量名 2*＞ [As ＜*类型 2*＞][＝*初始值 2*]]…

具体说明如下：

(1)语句中的声明关键字通常是 Dim、Private、Static、Public、Protected、Friend 或 Shared 中的一个，通常用 Dim。

(2)在声明变量的同时可以给变量赋初始值，即初始化变量，方法是在变量后面加上“＝初始值”。例如：

```
Dim IntVar As Integer＝100
Dim CharVar As Char＝"A"
Dim BolVar As Boolean＝False
```

(3)“变量名”必须符合命名规则，“类型 1”、“类型 2”等用来定义被声明的变量的数据类型，如果省略“As 类型”子句，若有初始值，则变量的数据类型为初始值的数据类型，若没有初始值，则变量的数据类型为对象类型。例如：

```
Dim IntVar＝100          '变量 IntVar 为整型
Dim IntVar               '变量 IntVar 为对象类型
```

(4)在一个声明语句中可以声明多个变量，变量的数据类型也可以不同。若变量是相同的数据类型，只需使用一个 As 子句就可以了，变量之间需要用逗号分开。例如：

```
Dim IntVar As Integer,CharVar As Char,BolVar As Boolean
Dim IntX,IntY,IntZ As Integer
Dim IntVar As Integer＝100,CharVar As Char＝"A"
```

需要注意的是，在声明多个同类型变量的同时不能对它们初始化。例如，下面的代码都是错误的：

```
Dim IntX＝100,IntY＝200,IntZ＝300 As Integer
Dim IntX,IntY,IntZ As Integer＝100
```

(5)声明变量时，使用不同的“声明关键字”定义的变量，其作用范围是不一样的，有关作用范围将在第 3 章详细介绍。

2)隐式声明

隐式声明指的是在使用变量之前不需要事先声明该变量，而是在变量名后面加上一个类型说明符来说明该变量的数据类型。如 LngVar&、StrVar $、SngVar! 分别表示变量 LngVar、StrVar、SngVar 为长整型、字符串型、单精度浮点型变量。表 2-3 给出了常见的数据类型和类型说明符的对应关系。

表 2-3　数据类型和类型说明符

数据类型	类型说明符	类型名称
Integer	%	整型
Long	&	长整型
Single	!	单精度浮点型
Double	#	双精度浮点型
Decimal	@	小数型
String	$	字符串型

默认情况下，Visual Basic. NET 编译器要求强制显式声明变量，即要求在每个变量使用前必须显式声明，否则会出错。如果用户去掉这个限制，将会允许隐式声明变量。方法如下：

(1)通过 Option Explicit 语句控制编译器的工作方式。在代码的开头写上 Option Explicit 语句，该语句的语法格式如下：

```
Option Explicit [On|Off]
```

参数 On 表示显式声明变量；参数 Off 表示允许隐式声明变量。如果省略参数，则默认为 On。例如：

```
Option Explicit Off
IntVar% =100
StrVar$ ="Visual Basic"
```

上面代码中，变量 IntVar 被隐式声明为整型变量，值为 100；变量 StrVar 被隐式声明为字符串型变量，值为"Visual Basic"。

注意：Option Explicit 必须放在所有声明语句之前。

(2)在解决方案资源管理器窗口中右击项目名称，在弹出的快捷菜单中选择“属性”命令，在打开的界面中单击“编译”选项卡，在 Option explicit 下拉列表框里选择 Off 选项，如图 2-1 所示，这样编译器允许隐式声明变量。

注意：尽管隐式声明比较方便，但却有很多弊端，例如，在程序中将变量名拼错，就会导致难以查找的错误，因此建议用户显式声明变量。

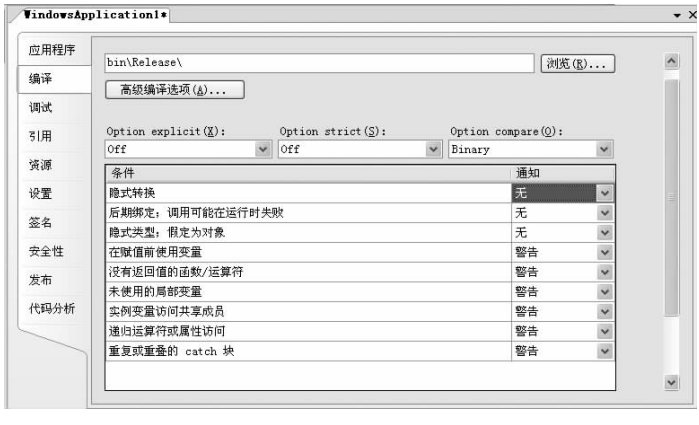

图 2-1　设置 Option Explicit

2.3 运算符与表达式

在程序里,常量和变量所代表的数据都是被操作的对象,运算符是操作这些数据的符号,表明对数据实施什么样的运算,表达式是用运算符和数据连接而成的式子,单个变量或常量也可以看成是一个表达式。Visual Basic. NET 中运算符有算术运算符、关系运算符、逻辑运算符以及字符串运算符等。

2.3.1 算术运算符与算术表达式

1. 算术运算符

算术运算符是用于数值计算的运算符。Visual Basic. NET 的算术运算符如表 2-4 所示。

表 2-4 算术运算符

运 算 符	名 称	算术表达式示例
+	加法	a+b
−	减法	a−b
*	乘法	a * b
/	浮点数除法	a/b
\	整数除法	a\b
^	乘方	a^b
Mod	求余	a Mod b
−	取负	−a

对于算术运算符,有两点说明:

(1)在上述算术运算符中,取负(−)运算符是单目运算符,其他运算符都是双目运算符。

(2)浮点数除法(/)和整数除法(\)的区别在于浮点数除法执行标准的除法运算,运算结果为浮点数,整数除法执行整除运算,运算结果为整数。

2. 算术表达式

算术表达式是由常量、变量、算术运算符、函数等连接而成的合法的运算式子。算术表达式返回的值是数值型的。例如:

```
Var1=10^2          '返回 100
Var2=10/4          '返回 2.5
Var3=10\4          '返回 2
Var4=10Mod 4       '返回 2
```

2.3.2　关系运算符与关系表达式

1. 关系运算符

关系运算符也称比较运算符，用于对两个相同数据类型表达式的值进行比较，被比较的数据的类型可以是数值型、字符型或日期型，甚至可以是两个对象类型。关系运算符两侧参加运算的表达式的数据类型必须完全一致。关系表达式返回的结果是逻辑值，只能是 True 或 False。Visual Basic. NET 的关系运算符如表 2-5 所示。

表 2-5　关系运算符

运 算 符	比较关系	关系表达式示例
=	等于	a=b
>	大于	a>b
>=	大于或等于	a>=b
<	小于	a<b
<=	小于或等于	a<=b
<>	不等于	a<>b
Like	根据模式比较字符串	"a" Like "abc"
Is	比较对象	A Is B
IsNot	比较对象，功能与 Is 相反	A IsNot B

2. 关系表达式

用关系运算符把算术表达式、字符串表达式等同类型的表达式连接起来构成的合法式子称为关系表达式，下面对关系运算符及表达式的使用进行说明：

(1)如果关系运算符比较的是两个数值表达式，这样的比较属于数值比较，常用表 2-5 里前 6 种关系运算符。两个表达式如果都是 Byte、Boolean、Integer、Long、Single、Double、Decimal 或 Date 类型，都能比较数值的大小。如果是日期型数据，则较早的日期小于较晚的日期。下面举例说明：

```
25>48                              '关系表达式的值为 False
#02/01/2000#<#02/05/2001#          '关系表达式的值为 True
a>b+2                              '如果 a 的值大于 b+2 的值，结果为 True，否则为 False
```

(2)如果关系运算符比较的是两个字符表达式，则结果取决于 Option Compare 语句的状态。Option Compare 语句的格式如下：

```
Option Compare {Binary|Text}
```

当为 Option Compare Binary 语句时，表示字符比较的规则是依据 ASCII 码大小进行比较，字符区分大小写，如关系表达式"A"<"a"的结果为 True。

当为 Option Compare Text 语句时，表示字符是以不区分字符大小写的文本排序顺序进行比较，这时"A"="a"的结果为 True。

默认情况下，Option Compare 的设置状态为 Binary，即字符比较按 ASCII 码区分大小写，本书中如不特别说明，都按默认设置处理。

(3)如果关系运算符比较的是两个字符串表达式时，规则和上面比较字符的规则一样，需要看 Option Compare 的设置状态，默认状态下，比较两个字符串大小时，从左边字符串的第一个字符开始，逐一与右边字符串的对应位置上的字符进行比较(比较 ASCII 码)，其中 ASCII 码值大的字符所在的字符串较大。如果第一个位置上的字符相等，则比较第二个位置上的字符，以此类推，直到比较出结果。

(4)Like 运算符用来比较两个字符串的模式是否匹配，即判断一个字符串是否符合某一模式。模式的表示可以通过以下通配符来表示：

- *，表示该位置字符可以是任意多个任意字符。
- ?，表示该位置字符可以是任意单个字符。
- #，表示该位置字符可以是任意单个数字。
- [list]，允许使用 list 列表里任意单个字符。
- [!list]，允许使用 list 列表外任意单个字符。

下面针对以上 4 种情况，举部分例子加以说明：

```
"a">"AB"                    '结果为 True
"aaa">"aa"                  '结果为 True
"56"<"9"                    '结果为 True
"aaa" Like "a*"             '结果为 True
"a2a" Like "a#a"            '结果为 True
"A" Like "[B-Z]"            '结果为 False
"A" Like "[!B-Z]"           '结果为 True
```

(5)如果关系运算符比较的是两个对象，则使用 Is 运算符来判断两个对象是否引用同一个对象，如果是，则返回结果为 True，不是则返回结果为 False；IsNot 的功能和 Is 的功能正好相反，它用来判断两个对象引用是否代表不同对象。下面举例说明：

```
Dim a As TextBox
Dim b As New TextBox
Dim c As Boolean
a=b
c=a Is b                    'c 的值为 True
```

上面程序，第四行代码的作用是使变量 a 和 b 引用同一个 TextBox 对象，因此 a Is b 返回的结果为 True。若去掉第四行代码，a 和 b 引用的对象不同，a Is b 返回的结果为 False。

2.3.3 逻辑运算符与逻辑表达式

1. 逻辑运算符

逻辑运算也称布尔运算，逻辑运算符是用来执行逻辑运算的运算符。Visual Basic. NET 的逻辑运算符如表 2-6 所示。

表 2-6　逻辑运算符

逻辑运算符	名　称	示　例	说　明
And	逻辑与	A And B	A、B 同时为 True，结果为 True，否则结果为 False
Or	逻辑或	A Or B	A、B 同时为 False，结果为 False，否则结果为 True
Xor	异或	A Xor B	A、B 取的逻辑值相同，则结果为 False，否则结果为 True
Not	逻辑非	Not A	对 A 的逻辑值取反
AndAlso	短路与	A AndAlso B	类似 And 运算
OrElse	短路或	A OrElse B	类似 Or 运算

2. 逻辑表达式

逻辑表达式由逻辑运算符、关系表达式、逻辑常量、变量和函数组成，运算结果为逻辑值。例如，Not(2>3)的值为 True，(5>3)And(3>5)的值为 False，(5>3)Or(3>5)的值为 True。

AndAlso、OrElse 运算符分别与 And、Or 运算符的功能相似。区别在于为了加快运算速度，当第一个表达式的值为 False 时，AndAlso 运算符将不再计算第二个表达式的值，而是直接确定整个表达式的值为 False；而当第一个表达式的值为 True 时，OrElse 运算符将不再计算第二个表达式的值，而是将整个表达式的值确定为 True。

2.3.4　字符串运算符与字符串表达式

1. 字符串运算符

字符串运算符适用于连接两个字符串的运算。字符串连接运算符有“＋”和“&”。例如：

```
"abcd"＋"efgh"                    '结果为 abcdefgh
"Visual Basic" & "程序设计"       '结果为 Visual Basic 程序设计
```

下面说明运算符“＋”和“&”的区别：

(1)“＋”运算符：两个操作数均应为字符串类型，若其中一个为数字字符型(如"100")，另一个为数值型，则自动将数字字符型转换成数值型，然后进行算术加法运算；若其中一个为非数字字符型(如"abc")，另一个为数值型，则出错。例如：

```
"100"＋100          '结果为 200
"100"＋"100"        '结果为 100100
"abc"＋100          '出错
```

(2)“&”运算符：两个操作数既可为字符型也可为数值型，当为数值型时，系统自动先将其转换为数字字符型，然后进行连接操作。例如：

```
"100" & 100         '结果为 100100
100 & 100           '结果为 100100
"abc" & "100"       '结果为 abc100
"abc" & 100         '结果为 abc100
```

注意：使用运算符“&”时，变量和“&”之间应加上一个空格，这是因为“&”还是长整型

的类型说明符，如果变量与“&”连在一起书写，系统会把“&”作为类型说明符处理，会出现语法错误。

2. 字符串表达式

一个字符串表达式就是由字符串常量、字符串变量、字符串函数、字符串运算符和括号连接形成的一个有意义的运算式子，如"Visual Basic" & "程序设计"。

2.3.5 运算符优先级比较

一个表达式中往往会同时出现多种运算符，Visual Basic. NET 规定了各种运算符的运算优先顺序，以便能正确计算出表达式的值。优先级高的运算符先运算，即首先进行算术运算，然后处理字符串运算，再处理关系运算，最后是逻辑运算；运算符优先级相同时，从左向右进行运算；括号内的运算优先进行；处在最内层括号里的运算优先进行，然后从内层向外层进行运算。

在 Visual Basic. NET 中，各运算符的优先级如表 2-7 所示。

表 2-7 运算符的优先级

优 先 级	运算符类型	运 算 符
1	算术运算符	^指数运算
2		－取负运算
3		*、/ 乘法和浮点除法运算
4		\整除运算
5		Mod 求余(模)运算
6		＋、－加法和减法运算
7	字符串运算符	＋、&
8	关系运算符	＝、<>、>、>＝、<、<＝、Like、Is、IsNot
9	逻辑运算符	Not
10		And、AndAlso
11		Or、OrElse
12		Xor

2.4 常 用 函 数

. NET 框架的两个重要组成部分是公共语言运行库(CLR)和. NET 基础类库。CLR 提供一些公共服务，如提供 Visual Basic. NET 使用的函数库。. NET 基础类库由数百个已定义的类(包括结构)和它们的成员组成。

编写程序过程中，用户可以直接使用 Visual Basic. NET 的函数库或. NET 框架基础类库中的函数来完成相应的运算。这些由系统本身提供的、用户可以直接使用的函数统称为

内部函数(库函数、标准函数)。

使用函数的格式是:

函数名([参数列表])

说明如下:

(1)Visual Basic. NET 函数库的命名空间和模块。Visual Basic. NET 命名空间内包含构成 Visual Basic. NET 运行库的类和模块,它们提供了丰富的函数,比较常用的模块有如下几种:

- VbMath:提供数学函数。
- Strings:提供字符串函数。
- DateAndTime:提供日期、时间函数。
- Conversion:提供类型转换函数。

(2). NET 基础类库中的命名空间和类。System 是. NET 基础类库的根命名空间,根据功能分为若干个子命名空间,各命名空间内含有若干类。如 Math 类,存在于 System 根命名空间内,该类中提供了数学函数,包括三角函数、对数函数、指数函数等。

由此可见,Visual Basic. NET 中常用的内部函数大多数已分别迁移到相应的命名空间中,使用时需要引入相应的命名空间或直接在函数名前写出相应的命名空间和所属的类(即函数的完全限定名),即

命名空间名. 函数名([参数列表])

2.4.1　数学函数

数学函数是系统提供给用户进行数学计算的函数,其中常用的数学函数及功能如表 2-8 所示。

表 2-8　数学函数及功能

函 数 名	功　　能	示　　例
Abs(x)	求 x 的绝对值	Abs(－3)结果是 3
Atn(x)	求角度 x 的反正切值	Atn(0)结果是 0
Cos(x)	求角度 x 的余弦值	Cos(0)结果是 1
Exp(x)	求 e^x 的值	Exp(0)结果是 1
Fix(x)	求 x 的整数部分。x 为正数时,去掉小数部分返回整数部分;x 为负数时返回大于等于 x 的最小整数	Fix(－99.8)结果是－99
Int(x)	求 x 的整数部分。x 为正数时,去掉小数部分返回整数部分;x 为负数时返回小于等于 x 的最大整数	Int(－99.8)结果是－100
Log(x)	求 x 的自然对数	Log(10)结果是 1
Rnd(x)	得到一个大于等于 0 小于 1 的随机数	(6 * Rnd)结果是 0～6 之间的随机数
Sgn(x)	根据 x 为正、零、负分别返回 1、0、－1	Sgn(14)结果是 1
Sin(x)	求 x 的正弦值	Sin(0)结果是 0
Sqrt(x)	求 x 的平方根	Sqrt(4)结果是 2
Tan(x)	求角度 x 的正切值	Tan(0)结果是 0

续表

函 数 名	功　　能	示　　例
Val(x)	求字符串 x 的数值	Val("24 and 1")结果是 24
Asc(x)	求字符串 x 首字母的 ASCII 码值	Asc("a")结果是 97
Chr(x)	返回 ASCII 码指定的字符	Chr(65)结果是 A
Str(x)	把数值 x 转换成字符串	Str(135)结果是"135"
Hex(x)	求十进制数 x 对应的十六进制数	Hex(15)结果是 F
Oct(x)	求十进制数 x 对应的八进制数	Oct(8)结果是 10

Visual Basic. NET 中的数学函数已迁移到 System 命名空间下的 Math 类中。调用形式如下：

System. Math. 函数名([参数列表])

由于 Math 类处于 System 命名空间下，所以可以在模块的开头处加入 Imports 语句：

Imports System. Math

这样，在程序中使用数学函数时，就可以省略“System. Math”，而直接写函数名即可。

【例 2-1】 制作一个万年历，用来查看某年的元旦是星期几。

计算某年元旦是星期几的计算公式如下：

F＝Int((Y－1)＊(1＋1/4－1/100＋1/400)＋1)

K＝F－Int(F/7)＊7

上面两个式子中 Y 为某年公元年号，计算出 K 为星期几，K＝0 为星期日，K＝1 为星期一，以此类推。

功能要求：

在文本框(txtYear)中输入年份，单击“查看”按钮(Button1)，在文本框(txtDay)中显示星期，运行界面如图 2-2 所示。

图 2-2 【例 2-1】运行界面

程序代码如下：

```
Private Sub Button1_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles Button1.Click
    Dim Y As Integer,F As Integer,K As Integer
    Y＝Val(txtYear.Text)
```

```
    F=Int((Y-1)*(1+1/4-1/100+1/400)+1)
    K=F-Int(F/7)*7
    txtDay.Text=K
End Sub
```

程序分析：

文本框的 Text 属性是字符型，而在计算中要使用的变量 a 是数值型，因此在计算中必须运用 Val(x)函数对文本框的 Text 属性值进行转换。

对数值型变量取整，可以用 Fix(x)或 Int(x)函数，当函数参数 x 是正数时，Fix(x)和 Int(x)结果相同；当参数 x 是负数时，则 Int 函数返回小于等于 x 的第一个负整数，而 Fix 函数则返回大于等于 x 的第一个负整数。

【例 2-2】 求方程 $ax^2+bx+c=0$ 的根。求解公式如下：

$$x1=(-b+\sqrt{b^2-4ac})/2a$$

$$x2=(-b-\sqrt{b^2-4ac})/2a$$

方程的根有以下几种可能：

(1)a=0，有一个实根。

(2)$b^2-4ac=0$，有两个相等的实根。

(3)$b^2-4ac>0$，有两个不等的实根。

(4)$b^2-4ac<0$，有两个共轭的复根。

界面设计：

界面由 5 个文本框、5 个标签和 1 个按钮组成，输入文本框为 txtA、txtB 和 txtC。界面安排如图 2-3 所示。

图 2-3　【例 2-2】设计界面

功能要求：

在文本框中输入 a、b、c 的值，单击“计算”按钮(btnStart)，计算方程根并将运行结果显示在文本框 txtX1 和 txtX2 中。

程序代码如下：

```
Imports System.Math
Public Class Form1
Private Sub btnStart_Click(ByVal sender As System.Object,ByVal e As _
```

```
System.EventArgs) Handles btnStart.Click
        Dim a,b,c,Disc,x1,x2,Rpart,Ipart As Single
        a=Val(txtA.Text)
        b=Val(txtB.Text)
        c=Val(txtC.Text)
        If Abs(a)<=0.000001 Then
            txtX1.Text=-c/b
            txtX2.Text="无解"
        Else
            Disc=b*b-4*a*c
            Rpart=-b/(2*a)
            If Abs(Disc)<=0.000001 Then
                txtX1.Text=Rpart
                txtX2.Text=Rpart
            ElseIf Disc>0.000001 Then
                x1=(-b+Sqrt(Disc))/(2*a)
                x2=(-b-Sqrt(Disc))/(2*a)
                txtX1.Text=x1
                txtX2.Text=x2
            Else
                Ipart=Sqrt(-Disc)/(2*a)
                txtX1.Text=Rpart & "+" & Ipart & "i"
                txtX2.Text=Rpart & "-" & Ipart & "i"
            End If
        End If
    End Sub
    End Class
```

程序分析：

在判断 b^2-4ac 是否等于 0 时，要注意一个问题，由于变量 Disc(即 b^2-4ac)是实数型，而实数在计算和存储时会有一些微小的误差，所以不能直接用语句“If Disc=0 Then…”进行判断，因为这样会出现本来是 0 的量由于上述误差而被判别为不等于 0 的情况，导致结果出错。统一采用的办法是判断 Disc 的绝对值(Abs(Disc))是否小于一个很小的数，如 0.000 001，如果小于此数则认为 Disc=0。

当计算的根为两个复数时，将实部和虚部用“&”组合成字符串显示在文本框中。

2.4.2 字符串函数

字符串函数用于进行字符串的处理。Visual Basic. NET 中的字符串函数已迁移到 Microsoft. VisualBasic 命名空间下的 Strings 模块中，其中常用的字符串函数及功能如表2-9所示。

表 2-9　字符串函数

函 数 名	功　　能	示　　例
Ltrim(字符串)	去掉左边空格	Ltrim("　hi　")结果是"hi　"
Rtrim(字符串)	去掉右边空格	Rtrim("　hi　")结果是"　hi"
Trim(字符串)	去掉前后空格	Trim("　hi　")结果是"hi"
Left(字符串,长度)	从左取指定长度的字符串	Left("hello",2)结果是"he"
Right(字符串,长度)	从右取指定长度的字符串	Right("hello",2)结果是"lo"
Mid(字符串,开始,[长度])	从开始位置取指定长度字符串	Mid("hello",2,2)结果是"el"
Len(字符串)	字符串长度	Len("hello")结果是 5
String(长度,字符)	重复字符	String(3,"*")结果是"***"
Space(长度)	重复空格	Space(5)结果是"　　"
Lcase(字符串)	变成小写字母	Lcase("Hello")结果是"hello"
Ucase(字符串)	变成大写字母	Ucase("Hello")结果是"HELLO"

【例 2-3】 将输入的字符串反向显示。

界面包括 2 个标签、1 个按钮(btnReverse)和 2 个文本框(txtInput 用于输入,txtReverse 用于显示反向后的字符串)。运行界面如图 2-4 所示。

图 2-4　【例 2-3】运行界面

功能要求:

从文本框 txtInput 输入原始的字符串,单击"反向"(btnReverse)按钮,在文本框 txtReverse 中显示反向后的字符串。

程序代码如下:

```
Private Sub btnReverse_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles btnReverse.Click
    Dim Str As String,StrReverse As String
    Dim i As Integer
    StrReverse=""
    Str=txtInput.Text
    For i=1 To Len(Str)
        StrReverse=Mid(Str,i,1) & StrReverse
    Next i
```

```
        txtReverse.Text=StrReverse
    End Sub
```

程序分析：

For 循环的次数是由字符串的长度 Len(Str)决定的，用 Mid 函数每次从字符串中第 i 个位置取一个字符。

【例 2-4】 译电文。为了使电文保密，往往按一定规律将其转换为密码，收报人再按规律译回原文。例如，按照以下规律：将字母 A 变成 E，即变成其后的第四个字母，相应的最后 4 个字母 W 变成 A，X 变成 B，Y 变成 C，Z 变成 D，小写字母也是同样。

图 2-5 【例 2-4】运行界面

界面设计：

界面由 2 个文本框(txtInput 和 txtOutput)、1 个命令按钮 btnStart 和 2 个标签组成，运行界面如图 2-5 所示。

功能要求：

在文本框(txtInput)中输入原文 world，单击“开始”(btnStart)按钮后，则会在文本框(txtOutput)中出现译过的密码 asvph。

程序代码如下：

```
    Private Sub btnStart_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles btnStart.Click
        Dim String1,String2,c As String
        Dim StrL,i As Integer
        string1=txtInput.Text
        String2=""
        StrL=Len(string1)
        i=1
        Do While i <=StrL
            c=Mid(string1,i,1)
            If(c >="a" And c <="z")Or(c >="A" And c <="Z")Then
                c=Chr(Asc(c)+4)
                If c>"Z" And Asc(c)<=Asc("Z")+4 Or c>"z" Then
                    c=Chr(Asc(c)-26)
                End If
            End If
            String2=String2 & c
            i=i+1
        Loop
        txtOutput.Text=String2
    End Sub
```

程序分析：

在 Do 循环中嵌套了 If 分支结构。Len 函数得到输入字符串的长度，循环次数由长度决定。Mid 函数取字符串中的一个字符，每个字符单独转换后再组合成字符串。Asc 函数得到字母的 ASCII 码，Chr 函数将 ASCII 码转换为字符。

2.4.3　日期和时间函数

日期和时间函数用于对日期和时间进行处理。Visual Basic. NET 中的日期和时间函数已迁移到 Microsoft. VisualBasic 命名空间下的 DateAndTime 模块中，其中常用的日期和时间函数及功能如表 2-10 所示。

表 2-10　日期和时间函数

函 数 名	功　　能	示　　例
Day(日期)	返回日期	Day(＃2010/3/10＃)结果是 10
Month(日期)	返回月份	Month(＃2010/3/10＃)结果是 3
Year(日期)	返回年份	Year(＃2010/3/10＃)结果是 2010
Weekday(日期)	返回星期几	Weekday(＃2010/3/10＃)结果是 3
TimeOfDay	返回系统时间	TimeOfDay 结果是 15:05:33
Date	返回系统日期	Date 结果是 2010-3-10
Now	返回系统日期和时间	Now 结果是 2010-3-10 15:05:33
Hour(时间)	返回小时	Hour(Now)结果是 15
Minute(时间)	返回分	Minute(Now)结果是 5
Second(时间)	返回秒	Second(Now)结果是 33

【例 2-5】　使用日期和时间函数在窗体上显示日期和时间。

界面设计：

共 9 个标签，其中 5 个标签显示如图 2-6 所示的文字，4 个空白标签 labYear、labMonth、labDay 和 labTime 分别显示年、月、日和时间。运行界面如图 2-7 所示。

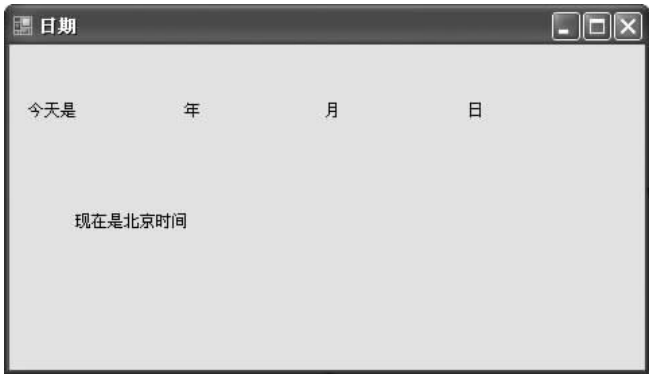

图 2-6　【例 2-5】设计界面

图 2-7 【例 2-5】运行界面

窗体加载事件(Load)程序代码如下：

```
Private Sub Form1_Load(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles MyBase.Load
    labYear.Text＝Year(Now)
    labMonth.Text＝Month(Now)
    labDay.Text＝Day(Now)
    labTime.Text＝Hour(Now)&":" & Minute(Now) & ":" & Second(Now)
End Sub
```

程序分析：

Text 为标签的属性，表示标签的显示文本。函数 Year(Now)返回当前系统时间的年份，Month(Now)返回当前系统时间的月份，Day(Now)返回当前系统时间的日期，Hour(Now)、Minute(Now)和 Second(Now)分别返回当前系统时间的小时、分和秒。

2.4.4 类型转换函数

类型转换函数用于转换函数参数的类型。Visual Basic. NET 中的类型转换函数已迁移到 Microsoft. VisualBasic 命名空间下的 Conversion 模块中，其中常用的类型转换函数及功能如表 2-11 所示。

表 2-11 类型转换函数

函 数 名	功 能	示 例
Cbool(x)	将表达式的值强制转换为 Boolean 类型	Cbool(－3)结果为 True
Cbyte(x)	将表达式的值强制转换为 Byte 类型	Cbyte(5.5)结果为 6
Cdate(x)	将表达式的值强制转换为 Date 类型	Cdate("12-30-2009")结果为＃12/30/2009＃
CDbl(x)	将表达式的值强制转换为 Double 类型	CDbl(33)结果为 33
Cint(x)	将表达式的值强制转换为 Integer 类型	Cint(7.5)结果为 8
Cshort(x)	将表达式的值强制转换为 Short 类型	Cshort(4.5)结果为 4
CStr(x)	将表达式的值强制转换为 String 类型	CStr(123)结果为"123"

续表

函 数 名	功　　能	示　　例
Str(x)	将数值转换为字符串	Str(123.45)结果为"123.45"
Val(x)	将字符串转换为数值	Val("－2.3")结果为－2.3
Ctype(x,y)	对指定的变量作数值类型转换	Ctype(n,Single)将变量 n 转换为 Single 类型

Val 函数的功能是将字符串转换为数值，如果被转换的字符串是英文字母，则该函数的函数值为 0；如果在字符串转换过程中遇到非数字字符(除指数符号、小数点和正负号外)，则转换停止，非数字字符不转换；如果被转换的数字字符中含有多个小数点，则转换到左数第二个小数点停止转换。例如，Val("23.56")的值为 23.56，Val("－45" & ".23")的值为－45.23，Val("5E－2")的值为 0.05，Val("－90.545.12")的值为－90.545，Val("88ab34.45")的值为 88，Val("abcxyz")的值为 0。

在用 Str 函数将数值转换为字符串时，如果被转换的数值是正数，则转换后对应的字符串包含一个前导空格，暗示为加号。如果为负数，则返回的字符串包含减号(－)，没有前导空格。例如，表达式 Str(23.56)的值为字符串" 23.56"，而 Str(－90.12)的值为字符串"－90.12"。

关于类型转换函数，注意以下几点：

(1)转换函数的参数值必须对目标数据类型有效，否则会发生错误。例如，如果把 Long 型数转换成 Integer 型数，Long 型数必须在 Integer 数据类型的有效范围之内。

(2)所有数据变量都可以相互赋值，在将浮点数赋予整数之前，VB. NET 要将浮点数的小数部分四舍五入，而不是将小数部分去掉。

(3)当将数值类型转换为 Boolean 型时，0 会转换成 False，而其他非 0 值则为 True。当将 Boolean 型转换为数值数据类型时，False 会转换成 0，而 True 会转换成－1。

(4)当其他数值类型转换成 Date 型时，小数点左边的值表示日期信息，小数点右边的值表示时间。

2.4.5　输入函数 InputBox

InputBox 函数在 Microsoft. VisualBasic 命名空间下，用于接收用户从键盘输入的数据，也称输入框。运行时，自动产生一个对话框，用户可以在其中输入数据。格式如下：

InputBox(对话框字符串 s[,标题 s][,文本框默认值 s][,横坐标值 n][,纵坐标值 n])

参数说明如下：

- 对话框字符串是对话框中显示的提示字符串，最大长度是 1 024 字符。如果提示字符串中包含多行，则可在各行之间用回车符(Chr(13))、换行符(Chr(10))或回车换行符的组合(Chr(13) & Chr(10))来分隔各行。
- 标题指对话框标题栏的字符串，如果省略，则标题栏中为应用程序名。
- 文本框默认值指文本框中显示的默认字符串，如果省略，则文本框为空。
- 横、纵坐标值指对话框在屏幕上的左上角位置(数值表达式)。

在调用 InputBox 函数时会出现一个对话框，包含一个文本框、“确定”和“取消”按钮。

对话框等待用户在文本框输入内容，如果单击“取消”按钮，则返回一个零长度字符串。

【例 2-6】 通过 InputBox 函数输入文件的路径并显示在窗体上。

界面设计：

界面包含两个标签(labFile 和 labFileName)和两个按钮(btnStart 和 btnEnd)，各控件的属性设置如下：labFile 和 labFileName 标签的 Text 属性值分别为“文件名：”和“”，btnStart和 btnEnd 按钮的 Text 属性值分别为“输入文件名”和“退出”。

运行情况：

程序运行主界面如图 2-8 所示，单击“输入文件名”按钮弹出“输入文件”对话框，如图 2-9(a)所示，单击“确定”按钮后程序运行结果如图 2-9(b)所示。

程序代码如下：

```
Private Sub btnStart_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles btnStart.Click
    Dim FileName As String
    FileName＝InputBox("请输入文件名：","输入文件","D:\Program Files\ _
              DEV-CPP\DevCPP.exe")
    If FileName <> "" Then
        labFileName.Text＝FileName
    Else
        labFileName.Text＝""
    End If
End Sub
```

图 2-8 【例 2-6】运行主界面

(a)InputBox函数运行界面　　(b)程序运行结果界面

图 2-9 【例 2-6】运行结果

程序分析：

在“输入文件”对话框中输入文件路径，赋值给 FileName 变量，并把结果显示在窗体的 labFileName 标签上。

2.4.6　输出函数 MsgBox

MsgBox 函数在 Microsoft. VisualBasic 命名空间下，用于向用户发布提示消息。在运行时，它会自动产生一个对话框，也称消息框，用来显示提示消息。对话框还包含命令按钮，要求用户通过单击按钮作出必要的响应，以提供程序继续执行的依据。格式如下：

MsgBox(消息文本 s[,显示按钮 s][,标题 s])

参数说明如下：

- 消息文本是在对话框中作为消息显示的字符串，最大长度是 1 024 字节，用于提示消息。如果提示字符串中包含多行，则可在各行之间用回车符(Chr(13))、换行符(Chr(10))或回车符与换行符的组合(Chr(13) & Chr(10))来分隔各行。
- 标题指对话框标题栏的字符串，默认时为空白。
- 显示按钮是一个枚举类型 MsgBoxStyle 值，用来控制在对话框内显示的按钮、图标的种类及数量。

需要注意的是，由 InputBox 和 MsgBox 函数生成的对话框要求用户在应用程序继续执行之前作出响应，即不允许对话框未关闭就进入程序的其他部分。例如，“MsgBox("是否退出系统?",vbOKCancel＋vbQuestion,"退出")”语句对应的消息框如图 2-10 所示。

图 2-10　MsgBox 消息框示例

【例 2-7】 例【2-6】的界面中包括一个“退出”按钮(btnEnd)，单击“退出”按钮后显示如图 2-10 所示的消息框。单击消息框的“确定”按钮退出系统，单击“取消”按钮取消操作。

程序代码如下：

```
Private Sub btnEnd_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles btnEnd.Click
    Dim Response
    labFileName.Text＝"D:\Program Files\DEV-CPP\DevCPP.exe"
    Response＝MsgBox("是否退出系统?",vbOKCancel＋vbQuestion,"退出")
    If Response＝1 Then
        End
    End If
End Sub
```

程序分析：

用变量 Response 存放 MsgBox 函数的返回值，当在图 2-10 中单击“确定”按钮时 Response＝1，单击“取消”按钮时 Response＝2。

2.5　程序的语句

在 Visual Basic. NET 中，语句是完整的指令，它可以包含关键字、运算符、变量、常量、常数和表达式等。语句有以下几种：Option 语句、Imports 语句、声明语句、赋值语句和调试

语句。

1. Option 语句

Option 语句为后续的代码建立基本规则，以防止语法和逻辑错误，包括 Option Explicit、Option Strict 及 Option Compare 语句。

(1)Option Explicit 语句。这种语句用于文件级，说明是否强制对该文件中的所有变量进行显式声明，可缩短以后用于调试的时间。

格式：

```
Option Explicit [On|Off]
```

参数说明如下：

- On：可选项。启用 Option Explicit 检查。如果在 Option Explicit 语句后未指定 On 或 Off，则默认为 On。
- Off：可选项。禁用 Option Explicit 检查。

如果使用，则 Option Explicit 语句必须出现在文件中所有其他源语句之前。当 Option Explicit 出现在文件中时，必须使用 Dim、Private、Public 或 ReDim 语句显式声明所有变量。试图使用未声明的变量名将发生编译错误。

如果没有使用 Option Explicit 语句，则所有未声明的变量都是 Object 类型。使用 Option Explicit 可避免拼错现有变量的名称，或避免在变量范围不清楚的代码中产生混淆。2.2.2 中有该语句的举例说明。

(2)Option Strict 语句。Option Strict 语句必须出现在文件中的任何其他源代码之前。Visual Basic. NET 允许将某些数据类型转换为其他数据类型。在将一种数据类型的值转换为另一种精度较低或表示范围较小的数据类型(收缩转换)时，可能发生数据丢失。如果此类收缩转换失败，将会发生运行错误。Option Strict 确保可为这些收缩转换提供编译时通知，从而避免这种错误。

格式：

```
Option Strict [On|Off]
```

参数说明如下：

- On：可选项。启用 Option Strict 检查。如果在 Option Strict 语句之后未指定 On 或 Off，则默认为 Off。
- Off：可选项。禁用 Option Strict 检查。

(3)Option Compare 语句。用于声明比较字符串数据时所使用的默认比较方法，即按它们的 Binary 排列或 Text 排列。

格式：

```
Option Compare [Binary|Text]
```

参数说明如下：

- Binary：可选项。按字符的内部二进制表示形式导出的排列顺序进行字符串比较。
- Text：可选项。按系统的区域设置确定的不区分大小写的文本排列顺序进行字符串比较。如果程序未包含 Option Compare 语句，则默认的文本比较方法是 Binary。

2. Imports 语句

Imports 语句允许对类和其他定义在导入命名空间的类型进行命名，而无须对它们进行

限制。

格式：

```
Imports [<别名>=] <命名空间>
```

或

```
Imports [<别名>=] <命名空间>.<元素>
```

参数说明如下：

- <别名>：可选项。一个名称，作为<命名空间>的别名使用。当 Imports 语句不包括<别名>时，即可在文件中无条件访问指定的<命名空间>中所定义的元素。如果指定<别名>，则必须将<别名>用作命名空间所包含名称的限定符。当需要使用在一个或多个命名空间中声明的同名项时，别名是有用的。
- <命名空间>：必选项。所导入的命名空间的名称。命名空间可以有任何数量的嵌套级深度。
- <元素>：可选项。命名空间中所声明的元素名称。可以是枚举、结构、类或模块。

每个文件可以包含任意数量的 Imports 语句。Imports 语句必须位于任何声明（包括 Module 或 Class 语句）之前，并位于任何标识符引用之前。

Imports 语句的可用元素的范围取决于使用 Imports 语句时的具体程度。例如，如果只指定了命名空间，那么该命名空间的所有唯一命名的成员以及该命名空间内模块的成员都是无条件可用的。如果同时指定了命名空间和该命名空间的一个元素名称，则只有该元素的成员无条件可用。

不允许在模块级定义与导入具有相同别名的成员。

例如，导入 Microsoft.VisualBasic.Strings 并给其分配一个别名 StrN，可用来访问 Right 方法。

```
Imports StrN=Microsoft.VisualBasic.Strings
    Sub ShowNet()
        MsgBox(StrN.Right("Visual Basic.NET",4))     '显示.NET
    End Sub
```

3.声明语句

使用声明语句可命名和定义过程、变量、数组和常数。声明的同时也定义了它们的范围，具体取决于声明的位置和用来声明它们的关键字。

例如：

```
Public Sub ApplyFormat()
    Const limit As Integer=33
    Dim myWidget As Widget
    …
End Sub
```

该示例包含 3 个声明：

(1)Public Sub 语句（带有匹配的 End Sub 语句）声明为 ApplyFormat 的过程。每当调用或运行 ApplyFormat 过程时，执行包含在 Public Sub 和 End Sub 语句中的所有语句。

(2)Const 语句声明常数 limit，指定 Integer 数据类型和初值 33。

(3)Dim 语句声明变量 myWidget。在此示例中数据类型是对象，即 Widget 对象。可以

将变量声明为在使用应用程序中公开的任何对象。Dim 语句是用于声明变量的一种语句类型。声明中使用的其他关键字有 ReDim、Static、Public、Private、Protected 和 Friend。

声明语句保留创建变量所需的内存,但不显式创建它。

如果变量是对象变量,则声明它时可以使用 New 关键字显式创建其类的实例,例如:

```
Dim x As New System.Windows.Forms.Form()
```

4.赋值语句

赋值语句执行赋值运算。简单的赋值运算包括将运算符右侧表达式的值赋给左侧的变量或对象的属性。运算符右侧可以是任何表达式(包括常量、变量、函数等)。

格式:

＜名称＞endow＜表达式＞

参数说明如下:

- ＜名称＞:变量或属性的名称。
- endow:赋值运算符,如=、&=、+=等。
- ＜表达式＞:可以是算术、字符串、日期、关系或逻辑表达式。

赋值语句计算时,先计算右边表达式的值,然后将其赋给左边的变量或对象的属性,例如:

```
Dim x As String
x="Con" & "cat" & "enation"     'x="Concatenation"
```

使用 Boolean 文本或 Boolean 表达式作为右侧参数,还可以赋予 Boolean 变量的值:

```
Dim x As Boolean=True
x=12>234                        'x=False
x=45>678 Or 45>12               'x=True
```

运算符左侧的参数还可以是属性。如设置文本框的文本属性的值:

```
MyTextBox.Text="This" & "is a" & "String"   MyTextBox.Text="This is a String"
```

5.调试语句

1)注释语句(Rem)

格式:

```
Rem＜注释内容＞
```

或

```
'＜注释内容＞
```

可以将 Rem 语句单独放在一行,也可以将其放在另一语句后。Rem 语句必须是该行上最后的语句。如果它跟在另一语句后面,则 Rem 与该语句间必须有一个空格。

一般都使用单引号(')代替 Rem。无论注释是跟在同一行上的另一语句后面,还是单独在一行,都可以这样做。

为了帮助说明其代码,大部分程序员大量使用嵌入式的注释。代码中的注释可以为以后阅读或使用某过程或特定指令的任何人员解释该过程或指令。但是在运行过程中将忽略注释。

2)暂停语句(Stop)

Stop 语句提供了一种以编程方式设置断点的替换方法。当调试器遇到 Stop 语句时,它将中断程序的执行,进入中断模式。

格式：

```
Stop
```

程序停止执行，不释放程序占用的内存，进入中断模式。此时，可以检查各变量的值。可以将 Stop 语句放在过程的任何地方以中止执行。使用 Stop 语句类似于在代码中设置断点。但它不关闭任何文件或清除任何变量，除非在已编译的可执行文件(.exe)中遇到。

注意：在应用程序的发布版本中应移除所有的 Stop 语句。

3)结束语句

结束语句(End)立即终止执行，使其他程序所持有的对象引用无效。

格式：

```
End
```

End 语句提供一种强迫程序停止的方法，可以放在过程的任何位置以结束代码执行，关闭 Open 语句打开的文件，并且清除变量，释放内存。End 语句调用 System 命名空间中的 Environment 类的 Exit 方法。System. Environment. Exit 要求使用者具有 SecurityPermissionFlag. UnmanagedCode 权限。如果使用者没有该权限，则会出现 SecurityException 错误。

End 语句清除模块级和类级的所有变量以及所有模块中的全部静态局部变量。

当与其他关键字一起使用时，End 指示过程或块的定义的结尾，如 End Class、End Enum、End Function、End IF、End Structure、End Sub、End Get、End Interface、End Module、End Namespace、End Property、End Select、End Set、End SyncLock、End Try、End While、End With。

6. 语句的书写规则

1)一行中有多条语句

在一行中可以有多条语句，语句之间用冒号(:)字符分隔。

例如：

```
Dim MyString As String="Hello World":MsgBox(MyString)
```

虽然这种形式的语法偶尔带来方便，但是它使代码难以阅读和维护。因此，建议读者保持一行一条语句。

2)跨多行继续一条语句

通常一行容纳一条语句，但当一行中容纳不下时，可以使用行继续符在下一行继续一条长语句，行继续符由一个空格及一个下划线字符“_”组成。在下面示例中 MsgBox 可执行的语句连续跨两行：

```
Public Sub DemoBox()
    Dim my Var As String
    MsgBox ("Hello" & myVar & _
           ".How are you?")
End Sub
```

3)检查编译错误

输入一行代码后，如果该行显示有蓝色波浪下划线(也可能显示错误信息)，则该语句中有语法错误。必须找出语句中有什么错误(如通过查找任务列表或通过悬停在错误上并阅读帮助文本)，然后改正它。在修正代码中的所有语法错误之前，程序无法正确地进

行编译。

2.6 程序控制结构

各种复杂的程序都是由若干基本结构组成的，当用 Visual Basic. NET 语言编程时，可以用 3 种控制结构来控制代码行的执行顺序，这 3 种基本控制结构是顺序结构、分支结构和循环结构，它们具有单入口、单出口的特点。

2.6.1 顺序结构

顺序结构是最简单的一种程序结构，整个程序按照语句的书写顺序依次执行。如图 2-11 所示，先执行 A，再执行 B，即自上而下依次运行。

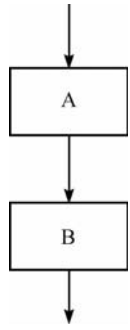

图 2-11 顺序结构

2.6.2 分支结构

分支结构如图 2-12 所示，首先判断条件 E 是否成立，然后根据判断的结果（真、假）决定执行哪些语句，分支结构有 3 种形式。

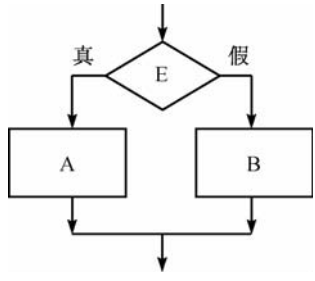

图 2-12 分支结构

1. If…Then 结构

If…Then 结构表示“如果……就”，是条件转移语句，根据条件测试后的结果，决定程序的下一步。其语法格式如下：

If 条件 Then 语句

或

```
If 条件 Then
  语句块
End If
```

其中，条件(表达式)的值为 Boolean 型。若条件为 True，则执行 Then 关键字后面的语句或语句块；否则，直接执行下一条语句或 End If 的下一条语句。若条件的值为数值，则当值为 0 时是 False，而任何非零数值都看做是 True。

例如，当满足条件 x＜y 时，执行 Sum＝Sum＋1：

```
If x＜y Then Sum＝Sum＋1
```

如果条件为 True 时要执行多行代码，则必须使用 If…Then…End If。例如：

```
If x＜y Then
    Sum＝Sum＋1
    z＝x
End If
```

2. If…Then…Else 结构

If…Then…Else 结构表示“如果……就……否则”，比前面的 If…Then 结构的条件选择和范围更广。其语法格式如下：

```
If 条件 1 Then
    语句块 1
[ElseIf 条件 2 Then
    语句块 2]
    …
    [Else
        语句块 n]
End If
```

运行时，首先测试条件 1，如果它为 False，就测试条件 2，以此类推，直到找到一个为 True 的条件就执行相应的语句块。如果条件都不是 True，则执行 Else 语句块。

【例 2-8】 设计一个查询是否中奖的程序。通过该程序查询是否中奖及中奖的等级。

界面设计：

界面由 4 个控件组成，分别是标签(Label1)、文本框(txtInput)、命令按钮“查询”(btnCheck)和标签(labResult)。程序运行界面如图 2-13 所示。

图 2-13　【例 2-8】运行界面

功能要求：

在文本框 txtInput 中输入奖券号码，单击“查询”按钮（btnCheck），在 btnCheck 的 Click 事件中查询是否中奖及中奖的等级。中奖号码为“123”，与中奖号码相同的为一等奖；前两位相同为二等奖；前一位相同为三等奖。中奖信息在标签 labResult 中显示。

程序代码如下：

```
Option Explicit On
Private Sub btnCheck_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles btnCheck.Click
    Dim strInput As String
    strInput＝txtInput.Text
    If strInput＝"123" Then
        labResult.Text＝"恭喜你，中了一等奖!"
    ElseIf strInput Like "12?" Then
        labResult.Text＝"恭喜你，中了二等奖!"
    ElseIf strInput Like "1??" Then
        labResult.Text＝"恭喜你，中了三等奖!"
    Else
        labResult.Text＝"谢谢你的参与!"
    End If
End Sub
```

程序分析：

(1)通配符“?”表示匹配单个字符。“12?”表示前两位为“12”的三位数。

(2)If 条件的顺序是：将“123”除外→将“12”除外→将“1”除外→剩余为不中奖的。

3. Select Case 结构

Select Case 结构与 If…Then…Else 结构类似，但对多重选择的情况，用 Select Case 语句代码效率更高，更易读。其语法格式如下：

```
Select Case 变量|表达式
    Case 值 1
        语句块 1
    Case 值 2
        语句块 2
    …
    Case 值 n
        语句块 n
    Case Else
        语句块 m
End Select
```

其中，值 1、值 2……、值 n 可以取以下几种形式：

(1)具体常数，如 1、2、"A"等。

(2)连续的数据范围，如 1 To 100、A To Z 等。

(3)满足某个条件的表达式,如 1>0 等。

(4)也可以同时设置多个不同的范围,用逗号将它们隔开,如-10,1 To 100。

Select Case 程序结构只计算一次表达式值,然后将表达式的值与结构中的每个 Case 的值进行比较。如果相等,就执行该 Case 值对应的语句块。如果不相等,则执行 Case Else 子句中的语句。

【例 2-9】　采用 Select Case 结构来设计查询是否中奖的程序。

本例界面设计、功能要求与【例 2-8】相同。程序运行界面见图 2-13。

程序代码如下:

```
Private Sub btnCheck_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles btnCheck.Click
    Dim strInput As String
    strInput=txtInput.Text
    Select Case strInput
        Case 123
            labResult.Text="恭喜你,中了一等奖!"
        Case 120 To 129
            labResult.Text="恭喜你,中了二等奖!"
        Case 100 To 199
            labResult.Text="恭喜你,中了三等奖!"
        Case Else
            labResult.Text="谢谢你的参与!"
    End Select
End Sub
```

程序分析:

(1)Select Case 结构在开始处计算表达式的值,而 If…Then…Else 结构在每个 ElseIf 语句都计算表达式的值。

(2)只有当 If 语句和每一个 ElseIf 语句的变量或表达式相同时,才能用 Select Case 结构替换 If…Then…Else 结构。因此 If 结构应用范围更广。

(3)如果不止一个 Case 与测试表达式相匹配,则只执行第一个匹配的 Case 语句块。

(4)Case Else 应放在 Select Case 结构的最后。

2.6.3　循环结构

循环结构用于处理重复执行的结构,可重复执行若干条语句。

1.Do 循环结构

可使用 Do 循环语句执行不确定次数的循环。Do 循环有两种形式,即“Do While 循环”和“Do 循环”。

“Do While 循环”的语法格式如下:

```
Do While|Until 条件
    语句块
```

```
    [Exit Do]
    语句块
Loop
```

“Do 循环”的语法格式如下：

```
Do
    语句块
    [Exit Do]
    语句块
Loop While|Until 条件
```

Do While 循环结构程序流程图如图 2-14 所示。

Do While 循环的步骤：执行 Do While 循环时首先测试条件，如果条件为 True，则执行语句块 A，然后继续执行 Do While 语句测试条件；如果条件为 False，则跳过所有语句到循环体外。

Do 循环结构程序流程图如图 2-15 所示。

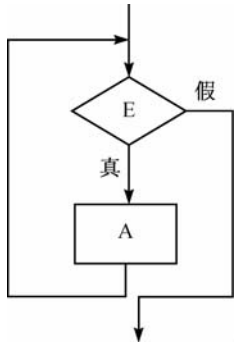

图 2-14 Do While 循环结构流程图

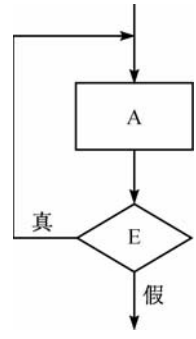

图 2-15 Do 循环结构流程图

Do 循环与 Do While 循环所不同的是先执行语句块 A，然后测试条件，如果条件为 True 则再次执行语句块 A，如此循环；如果条件为 False，则跳过循环体，这种 Do 循环保证语句块至少被执行一次。

Until 和 While 不同，判断条件正好相反。Until 循环结构是只要条件为 False（而不是 True），它们就执行循环，直到条件为真跳出循环体。Until 循环结构程序流程图如图 2-16 和图 2-17 所示。

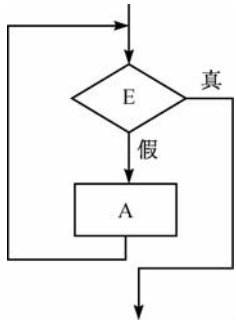

图 2-16 Do Until 循环结构流程图

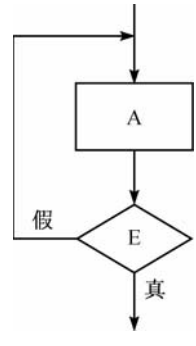

图 2-17 Until 循环结构流程图

【例 2-10】　用 Do While 循环计算 1～100 的和。

```
Dim i As Integer,Sum As Integer
Sum＝0
i＝1
Do While i＜＝100
    Sum＝Sum＋i
    i＝i＋1
Loop
```

Sum 的结果是 5050。

【例 2-11】　用 Do 循环计算 1～100 的和。

```
Dim i As Integer,Sum As Integer
Sum＝0
i＝1
Do
    Sum＝Sum＋i
    i＝i＋1
Loop While i＜＝100
```

Sum 的结果是 5050。

如果将循环体外的赋初值语句由“i＝1”改为“i＝101”，则两种不同的 Do…Loop 结构结果就不同了：在【例 2-10】中判断条件后直接跳出循环，Sum 的结果是 0；在【例 2-11】中进入循环体一次后，再判断条件跳出循环，Sum 的结果是 101。

2. While 循环结构

可使用 While 循环语句执行不确定次数的循环，While 循环也称为“当型”循环。“当型”循环结构程序流程图如图 2-18 所示。

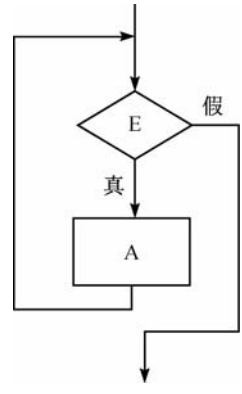

图 2-18　While 循环结构流程图

“当型”循环结构的语法格式如下：

```
While 条件
        语句块
        [Exit While]
```

```
    语句块
End While
```

“当型”循环的步骤：执行 While 循环时首先测试条件，如果条件为 True 就执行语句块，然后循环执行 While 语句测试条件；如果条件为 False，则跳过所有语句到循环体外。

【例 2-12】 用 While 循环计算 1～100 的和。

```
Dim i As Integer,Sum As Integer
Sum＝0
i＝1
While i＜＝100
    Sum＝Sum＋i
    i＝i＋1
End While
```

Sum 的结果是 5050。

可以看出，“当型”循环与“Do While”循环的程序执行流程基本相同。

3.For 循环结构

For 循环结构使用最为灵活。它使用一个计数器，每循环一次，计数器变量的值就会增加或减少。其语法格式如下：

```
For 计数器＝初始值 To 终止值［Step 步长］
    语句块
    ［Exit For］
Next［计数器］
```

For 循环结构流程图如图 2-19 所示。

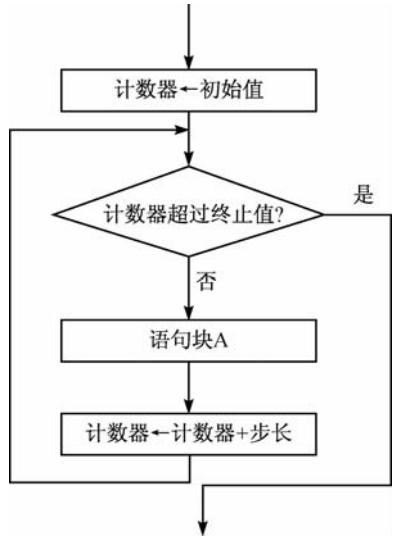

图 2-19 For 循环结构流程图

执行 For 循环的步骤如下：

(1)设置计数器等于初始值。

(2)如果步长为正,测试计数器是否大于终止值。若步长为负,则测试计数器是否小于终止值。如果是,则退出循环。

(3)否则,执行语句块 A。

(4)计数器＝计数器＋步长。

(5)转到步骤(2)。

注意:步长可正可负。如果为正,则初始值必须小于等于终止值,否则不能执行循环体内的语句。如果没有设置 Step 项,则步长默认值为 1。

【例 2-13】 用 For 循环结构计算 1～100 的和,步长为 1。

```
Dim i As Integer,Sum As Integer
Sum＝0
For i＝1 To 100
    Sum＝Sum＋i
Next i
```

注意:在不知道需要执行多少次循环时,适宜用 Do 循环,否则最好使用 For 循环。

【例 2-14】 在查询中奖号码的例子中添加摇奖程序。

功能要求:

摇奖是用 For 循环由随机数生成器循环产生 3 位数的中奖号码,为了获得摇奖的效果,出现每个号码之间有一段时间间隔,用一个 For 循环生成延时程序。

界面中增设一个文本框(txtPrize)、一个“摇奖”按钮(btnStart)和一个标签。单击“摇奖”(btnStart)按钮后的界面如图 2-20 所示。

图 2-20　单击“摇奖”按钮后的界面

程序代码如下:

```
Private Sub btnStart_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles btnStart.Click
    Dim i As Integer,j As Integer
    Dim a As Single
    Dim StrPrize As String
    StrPrize＝""
    For i＝1 To 3
        j＝Int(10 * Rnd())
        StrPrize＝StrPrize & j
        For a＝1 To 10000 Step 0.001
        Next a
```

```
            txtPrize.Text＝StrPrize
            txtPrize.Refresh()
        Next i
    End Sub
```

说明:以上代码仅为单击“摇奖”按钮要执行的代码,单击“查询”按钮的时候,还是跟中奖号码“123”进行的比较。

程序分析:

(1)字符串连接用“&”符号。

(2)使用空 For 循环来延时,每次循环增加步长 0.001。

(3)txtPrize.Refresh 语句用来在循环中刷新文本框,使每摇一次奖,在文本框立即显示新内容。

(4)Rnd()为产生随机数的函数,见表 2-8。

4.退出循环结构

用 Exit 语句可以直接退出 For 循环、Do 循环和 While 循环。程序执行中遇到 Exit 语句时,就不再执行循环结构中的任何语句而立即退出,转到循环结构的下面继续执行。其语法格式如下:

```
Exit For
Exit Do
Exit While
```

执行 Exit 语句只能退出该语句所在的循环体。Exit 语句几乎总是出现在循环体内嵌套的 If 语句或 Select Case 语句中。

【例 2-15】 通过在计算 1～100 的和的代码中使用 Exit For 语句,说明 Exit For 语句的用法。

```
Dim i As Integer,Sum As Integer
Sum＝0
For i＝100 To 1 Step －1
    Sum＝Sum＋i
    If Sum＞＝5000 Then Exit For      '当 Sum 大于等于 5000 时跳出循环
Next i
```

程序分析:程序执行结束后 Sum 的值为 5 000。

注意:当运行程序进入死循环时,按 Ctrl＋break 组合键可以终止程序的运行。

2.7 枚举类型

当一个变量只有几种固定的取值时,可以定义枚举类型,即将该变量的取值一一列举出来,从而使该变量的取值只限于列举出来的值的范围。这种方法不仅能提高程序的可读性,而且可以减少错误。

1.枚举数据类型的定义

枚举数据类型采用 Enum 语句定义,格式如下:

```
[Public|Private] Enum 枚举名称 [As 数据类型]
    成员名 1 [=常数表达式]
    成员名 2 [=常数表达式]
    …
End Enum
```

说明如下：

(1)定义枚举类型时，可在 Enum 关键字前面加上 Public、Private，表示枚举类型可以使用的范围。如果省略，则默认为 Public，表示所定义的 Enum 类型在整个程序中都是可见的、有效的。如果使用 Private 关键字，则表示所定义的 Enum 类型只在所声明的模块中可见、有效。

(2)“数据类型”可以是 Byte、Integer、Long、SByte、Short、UInteger、ULong 或 UShort 等。如果没有指定“As 数据类型”，则默认为 Integer 类型。

(3)“枚举名称”由用户给出，其命名规则同变量的命名规则，“成员名”是枚举数据类型中所包含的各个成员的名称，这些成员是常数。

(4)“常数表达式”是可选的，如果省略，在默认情况下，枚举中的第一个成员(常数)被初始化为 0，其后的成员初始化为比其前面的成员大 1 的数值。例如：

```
Public Enum MyWeekDay As Integer
    Mon
    Tue
    Wed
    Thu
    Fri
    Sat
    Sun
End Enum
```

在此表示定义了一个枚举类型 MyWeekDay，它包含 7 个成员，都省略了“常数表达式”，所以成员 Mon 的值为 0，成员 Tue 的值为 1，成员 Wed 的值为 2……

(5)可以用赋值语句显式地给枚举中的成员赋值，所赋的值可以是任何整数型的数。例如：

```
Public Enum MyWeekDay As Integer
    Mon=1
    Tue
    Wed
    Thu
    Fri
    Sat
    Sun
End Enum
```

在此表示定义了一个枚举类型 MyWeekDay，它包含 7 个成员，成员 Mon 的值为 1，成员 Tue 的值为 2，成员 Wed 的值为 3……

(6)引用枚举类型中的成员的格式是：

枚举名称.成员名

例如，要引用 MyWeekDay 中的成员 Tue 时，应写为 MyWeekDay. Tue。

2. 定义枚举类型的变量

当程序中定义了枚举类型后，就可以像定义一个基本数据类型的变量那样定义一个枚举类型的变量。格式与 2.1.1 中所介绍的定义基本数据类型的变量的格式完全一样。例如：

```
Public Class Form1
  Private Sub Form1_Click(ByVal sender As Object,ByVal e As _
System.Eventargs) Handles Me.Click
    Dim w As MyWeekDay
    …
    End Sub
End Class
```

这样，在书写代码的过程中，当给变量 w 赋值时(即当输入“w＝”时)，系统将自动提示可以给变量 w 赋值的值列表。

2.8 数　　组

在处理实际问题时，常常遇到处理同一类型的成批数据，如一个班级 30 名学生的课程成绩，这时可以用一个 s 来代表这批数据，用 s_1、s_2、s_3…表示第一个、第二个、第三个……学生的成绩。数组中不同元素以下标来区别，下标代表数组元素在数组中的顺序。在内存中，一个数组按照下标顺序占据一片连续的存储单元，在 Visual Basic. NET 的数组中，规定将下标用括号括起来。

2.8.1 数组的声明

数组是由名称相同而下标不同的一组下标变量组成的集合，其中任何一个下标变量都有一个确定的下标来识别，称为数组中的一个数组元素，它们有以下基本特性：

(1)整个数组中的下标有上界和下界，数组元素在上下界内是连续的，下标的上下界决定了该数组中所包含的元素个数(数组的大小)。

(2)一个数组中的所有元素具有相同的数据类型。

(3)根据数组中下标的个数将数组分成一维数组、二维数组和多维数组。由具有一个下标的下标变量所组成的数组称为一维数组，而由具有两个或两个以上下标的下标变量所组成的数组称为二维数组或多维数组。

和变量一样，在程序中使用数组，必须先定义数组，即声明数组。声明数组时需要说明数组名、类型、维数和数组的大小。

声明格式如下：

声明关键字 数组名(下标 1 上界[,下标 2 上界]…)[As 数据类型]

说明如下：

(1)语句中的“声明关键字”可以是 Dim、Private、Static、Public 等,通常用 Dim,与声明变量的含义相同。

(2)格式中的“数组名”命名规则与简单变量命名规则相同,“数据类型”表示数组的数据类型。当省略 As 子句后面的数据类型时,表示数组为 Object 类型,这时能在一个数组中同时存放各种不同类型的数据,而其他类型的数组中只能存放同一种类型的数据。

(3)Visual Basic. NET 规定,数组每一维的下标下界从 0 开始计数,每一维的下标上界分别由“下标 1 上界”、“下标 2 上界”等来指明,由此就确定了数组每一维的大小,例如:

```
Dim s(5) As Single
```

该语句表示定义了一个名称为 s 的一维数组,该数组中的元素的下标下界从 0 开始,下标上界到 5 为止,即下标的变化范围为 0～5。该数组中共有 6 个数组元素,它们都是 Single 类型,分别为 s(0)、s(1)、s(2)、s(3)、s(4)、s(5)。它们分别相当于 6 个变量,在内存中所占的内存单元是连续排列的,如图 2-21 所示。

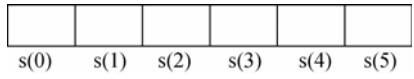

图 2-21　一维数组 s 在内存中的排列

(4)当定义多维数组时,各维之间要用逗号隔开,每一维指定下标上界即可。例如:

```
Dim a(1,3) As Double
```

表示定义了一个二维数组 a,其类型为 Double,第一维的下标上界为 1;第二维的下标上界为 3,共有 8 个元素,它们的名称分别为 a(0,0)、a(0,1)、a(0,2)、a(0,3)、a(1,0)、a(1,1)、a(1,2)、a(1,3),在逻辑上可以理解为这 8 个元素排列成 2 行 4 列的形式。

因此,可以将二维数组的第一下标看成行下标,行下标的大小决定了某个元素在第几行;第二下标可以看成列下标,列下标的大小决定了某个元素排在第几列。

在计算机内存中,二维数组各元素所占的内存单元是按行连续排列的,即先存放二维数组中第一行的元素,再存放第二行的元素……直到二维数组的最后一行。对于前面定义的二维数组 a,各元素所占的内存单元在内存中的排列顺序如图 2-22 所示。

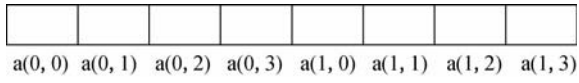

图 2-22　二维数组 a 在内存中的排列

(5)声明了数组后,数组中元素个数、类型等就确定了。程序中即可使用每一元素的名称来对数组的元素进行操作,数组中元素的名称由下标序号来确定,格式如下:

数组名(下标 1[,下标 2]…)

数组元素在程序中的使用方法与普通变量相同,可以赋值、参加运算等。所不同的是数组元素有下标,而普通变量不带下标。

例如,如果用语句 Dim x(2) As Integer 定义数组 x,则程序中语句 x(1)＝100 表示将元素 x(1)赋值成 100,此处的 x(1)就是指数组 x 中下标序号为 1 的元素。

(6)对于数值型数组,在定义后系统将各元素初始化为 0;对于字符型数组,在定义后系统将各元素初始化为空字符。

(7)可以声明没有大小的空数组,但这种数组必须使用 ReDim 语句重新定义大小后才

能使用。例如：

```
Dim a() As Integer,b(,) As Long
```

表示定义了一个一维的空数组 a 和一个二维的空数组 b。

【例 2-16】 求用语句 Dim a(2,3) As Integer 所定义的数组元素的个数。

本题中定义一个具有两个下标的二维数组 a，其中第一维下标的上界值为 2，第二维下标的上界值为 3。由于 Visual Basic. NET 规定数组的下标从 0 开始计数，所以数组 a 中包含的元素有 12 个，分别是 a(0,0)、a(0,1)、a(0,2)、a(0,3)、a(1,0)、a(1,1)、a(1,2)、a(1,3)、a(2,0)、a(2,1)、a(2,2)、a(2,3)。

2.8.2 数组的初始化及引用

1. 数组的初始化

数组的初始化就是在声明数组的同时给数组元素提供初值，其格式如下：

一维数组的初始化：

声明关键字 数组名() As 数据类型={初始值序列}

二维数组的初始化：

声明关键字 数组名(,) As 数据类型={{第一行数据},{第二行数据}}

说明如下：

(1)初始化时，“初始值序列”要用大括号括起来，且其中的各数据必须为常数，各数据间用逗号隔开。例如：

```
Dim a() As Integer={1,3,5,7,9}
```

表示声明了一维数组 a 并进行了初始化。数组 a 中共有 5 个元素，这 5 个元素的初值分别为 a(0)=1、a(1)=3、a(2)=5、a(3)=7、a(4)=9。

再如：

```
Dim b(,) As Integer={{1,2,3,4},{5,6,7,8}}
```

表示声明了二维数组 b 并进行了初始化。数组 b 中共有 8 个元素，这 8 个元素的初值分别是 b(0,0)=1、b(0,1)=2、b(0,2)=3、b(0,3)=4、b(1,0)=5、b(1,1)=6、b(1,2)=7、b(1,3)=8。

(2)当对数组进行初始化时，不能声明下标上界，数组的大小由系统根据初始值序列中数据的个数来确定。例如，语句 Dim c(5) As Integer={1,2,3,4,5,6}是错误的。

2. 数组的引用

数组的引用通常是指对数组中元素的引用，也就是使用数组中的元素，如在程序中可以给指定的数组元素赋值，用指定的数组元素参加有关的运算等。在程序中引用数组元素的方法是，在数组名后的括号中写明该元素的各个下标序号。

数组中每一个元素的使用方法如同程序中的一个简单变量一样，可以被赋值、参加各种运算等，只不过数组元素是带下标的变量，而简单变量不带下标。在引用数组中的元素时应注意以下几个问题：

(1)在引用数组元素时，下标可以用常数表示，也可以用变量表示。例如：

```
Dim s(i) As Single
```

当变量 i=1 时，s(i)就相当于数组元素 s(1)；当 i=3 时，s(i)就相当于数组元素 s(3)。

(2)引用数组元素时，其下标值应处在定义数组时所指定的下标下界值和上界值之间，否则会出现“下标越界”的错误。例如：

```
Dim a(10) As Integer
a(11)=100              '错误，此处下标越界，数组 a 中不存在下标序号为 11 的元素
```

2.8.3　动态数组

由于事先没法确定数组应该定义成多大才合适，所以用户希望程序在运行时还能改变数组的大小。在 Visual Basic. NET 中对已经声明的数组大小进行动态改变，也就是重新定义数组的大小，可以通过 ReDim 语句来实现，其格式是：

```
ReDim [Preserve] 数组名(下标 1 上界 [,下标 2 上界]…)
```

例如：

```
Dim x(3) As Integer     '声明数组 x，共 4 个元素
…
ReDim x(4)              '重新定义数组 x 的大小，使数组 x 中元素的个数增加到 5 个
```

再如：

```
Dim y(,) As Integer     '声明一个没有大小的二维空数组 y
…
ReDim y(,5)             '重新定义数组 y 的大小，使数组 y 中元素的个数为 6
```

说明如下：

(1)ReDim 语句的作用是按定义的上界重新给数组分配存储单元。

(2)ReDim 语句是一个可执行语句，它只能出现在过程中，而 Dim 语句是说明性语句，可以出现在程序的任何地方。

(3)可以用 ReDim 语句多次反复地定义同一个数组，并改变数组的大小，但不能用 ReDim语句改变数组的维数和数据类型。

(4)Ubound 函数与 Lbound 函数在 Microsoft. VisualBasic 命名空间下的 Information 类中，它们可以求得数组某一维的上界值、下界值。其格式为：

```
Ubound(数组名 [,维数])
Lbound(数组名 [,维数])
```

例如：

```
Dim MyArray(10,15,20)          '声明数组
Upper=Ubound(MyArray,1)        '返回数组 MyArray 第 1 维的上界值
Upper=Ubound(MyArray,2)        '返回数组 MyArray 第 2 维的上界值
```

(5)用 ReDim 语句重新定义一个数组的大小时，数组中各元素原有的值将丢失。有时用户希望能用 ReDim 语句改变数组大小又不丢失数组中原有数组元素的数据，这时使用具有 Preserve 关键字的 ReDim 语句就可做到这一点，但这时只能改变最后一维的大小，前面几维大小不能改变。例如：

```
ReDim Preserve DynArray(Ubound(DynArray)+1)
```

这样，在改变数组 DynArray 大小的同时，数组中原有元素的值继续保留。

2.9 结构类型

2.9.1 定义结构类型

普通变量中只能存放某种类型的一个数据，但有时在处理问题时，需要把几个不同类型的数据集合在一起当成一个整体来对待。例如，要表示一个学生的基本信息，包括学生学号、姓名、班级和年龄等，显然，用一个普通的变量是不能完成的，因为这其中有 4 个数据，而且它们的类型不同。另外，若用 4 个不同类型的变量分别表示学生学号、姓名、班级和年龄，这 4 个变量之间又缺乏必然的联系，不是一个整体。为了能同时表示这 4 个数据，并能使它们作为一个整体进行处理，就需要结构数据类型。

Visual Basic. NET 中定义结构数据类型的语法格式是：

```
[Public|Private] Structure 结构类型名
    Dim|Private|Public 成员名 1 As 数据类型
    Dim|Private|Public 成员名 2 As 数据类型
End Structure
```

说明如下：

(1)定义格式中的“结构类型名”是用户所要定义的类型的名称，需要由用户给出，其命名规则同变量的命名规则。“成员名”是指用户定义的结构类型中所包含的各个成员，其命名规则也同变量的命名规则。“数据类型”可以是前面所介绍的 Visual Basic. NET 中的各种数据类型。

(2)结构数据类型不能在过程内部定义，其定义语句必须放在模块或窗体的声明部分。

(3)声明结构类型时，关键字 Structure 前面可加 Public 或 Private 关键字，也可以省略，省略时默认为 Public，表示所定义的结构类型在整个程序中都是可见的，有效的。

例如，若要定义一个名称为 studentRecord 的结构类型来表示学生的基本信息，则定义语句为：

```
Public Structure studentRecord
    Dim Sid As Integer              '学号
    Dim Sname As String             '姓名
    Dim Sclass As String            '班级
    Dim Sage As Integer             '年龄
End Structure
```

该例中定义了一个名称为 studentRecord 的结构数据类型，它包含 4 个成员，即 Sid(为 Integer 类型)、Sname(为 String 类型)、Sclass(为 String 类型)、Sage(为 Integer 类型)。

2.9.2 声明和使用结构类型变量

1. 声明结构类型变量

如果程序中已经定义了结构数据类型，就可以在程序中定义某个变量属于这种类型，格

式与定义基本数据类型的变量格式完全一样。由于前面已经定义了 studentRecord 为结构数据类型，所以可以定义某个变量属于该种类型。

```
Dim stu As studentRecord
Private stu1 As studentRecord
```

2. 使用结构类型变量

如果已经定义了某个结构类型的变量，该变量就具有了该种结构类型的全部成员，要使用其中的某个成员时，格式为：

结构类型变量名.成员名

例如，要表示前面已定义的 stu 变量中的学号、姓名、班级和年龄成员，其表示形式分别是 stu. Sid、stu. Sname、stu. Sclass 和 stu. Sage。

结构类型变量中的每一个成员就是一个变量，可以像普通变量那样被赋值、参加各种运算等。

例如，若要对前面已定义的 stu 变量中的学号、姓名、班级和年龄成员赋值，可用以下 4 条赋值语句完成。

```
stu.Sid=" "
stu.Sname="ZhangSan"
stu.Sclass="计算机-1 班"
stu.Sage=18
```

当逐一表示结构类型变量中的成员时，书写太烦琐，这时可以用 with…End with 语句进行简化，语句格式如下：

```
with 结构类型变量名
  语句块
End with
```

例如，对 stu 变量中的学号、姓名、班级和年龄成员赋值的 4 条语句可简化为：

```
with stu
  .Sid=" "
  .Sname="ZhangSan"
  .Sclass="计算机-1 班"
  .Sage=18
End with
```

可见，在 with…End with 语句块中，引用结构类型变量中的某个成员时，可省略结构变量，直接写“.”和成员名即可。

2.10　集　　合

集合与数组不同，数组是用于存放同一类数据的，如果需要存放一组不同类型的数据，就需要使用集合来实现。引入集合主要是为了将一组相关的对象作为一个整体来操作。集合中的元素没有先后顺序的关系，相同的元素也不能出现在同一个集合中。

一个集合就是一组相互关联的对象，用在程序中处理控件和其他数据。在 Visual

Basic. NET 中，窗体上的对象都保存在同一个文件中，它们被看做是同一组的成员。窗体上的所有对象集合称为控件集合。每个窗体都有一个控件集合，该集合是在打开新窗体以及向窗体中添加对象时自动创建的。实际上，Visual Basic. NET 提供了多个标准的对象集合，以便程序设计时使用。

2.10.1 集合的建立

与数组不同的是，集合本身就是对象。创建对象集合的第一步是创建 Collection 对象，然后通过这个对象来添加、移除其中的成员或进行其他操作。

尽管集合通常用于存放对象，如用户界面上的控件对象等，但也可以在程序运行期间存放数值或字符串，因此，集合是数组的一种补充。有时用集合存取数据比用数组更加有效。例如，如果要处理的是一批个数较少的、动态的数据项，就可能需要使用集合。

【例 2-17】 自定义一个集合，为其添加成员并操纵其成员。

界面设计：

界面包括 3 个按钮，其 Name 属性值分别为 add、search 和 delete，3 个按钮的 Text 属性值分别为“添加成员”、“查找成员”和“删除成员”，如图 2-23 所示。

图 2-23 【例 2-17】界面设计

功能要求：

程序运行后，单击“添加成员”按钮，集合中将添加 4 个成员，分别是：

ycollect(1):"张三"

ycollect(2):"是"

ycollect(3):"北京大学的"

ycollect(4):"三好学生"

程序代码如下：

```
Public Class Form1
    Dim ycollect As New Collection  '定义一个集合及声明一个 Collection 变量
    Private Sub add_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles 添加成员.Click
        Dim a,b,c,d As String
        a="张三"
        b="是"
        c="北京大学的"
        d="三好学生"
```

```
            ycollect.Add(a,"这")
            ycollect.Add(b,"是")
            ycollect.Add(c,"一个")
            ycollect.Add(d,"键")
        End Sub
        Private Sub search_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles 查找成员.Click
            Dim ystr As String
            For Each ystr In ycollect
                If ystr="三好学生" Then
                    MsgBox(ystr)
                End If
            Next
        End Sub
        Private Sub delete_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles 删除成员.Click
            ycollect.Remove(1)
            ycollect.Remove("是")
        End Sub
    End Class
```

程序分析：

单击“查找成员”按钮，将找到值为“三好学生”的第四个成员，并显示其值，如图 2-24 所示；单击“删除成员”按钮，将删除第一个及其值为“是”的成员。

图 2-24　单击“查找成员”按钮

2.10.2　集合成员的引用

如果指定了一个对象在集合中的索引位置，就可以在程序中使用集合中的成员。Visual Basic. NET 按照程序创建对象的相反顺序在集合中存放对象，因此，可以利用某个对象的“创建顺序”单独引用该对象，或者使用循环结构依次处理几个对象。

例如，在设计应用程序时，先要设计窗体，即将一些控件添加到窗体上或窗体上已有的某个容器控件（如 Panel、GroupBox 等）中，也可以从窗体上或容器控件中移除控件。在设计阶段，这些控件是直接拖放到窗体上或容器控件中的，而在运行时，它们组成一个 Controls

集合,可以用程序代码来操纵这个集合,从而操纵其中包含的某个控件。例如,语句:

```
Controls(0).Text="最后添加的控件"
```

将窗体中最后添加的控件的 Text 属性设置为“最后添加的控件”。集合中倒数第二个创建的对象的索引值为 1,倒数第三个创建的对象的索引值为 2,以此类推。可见,如果向集合中添加一个对象,新对象将占据 0 索引位置,而其余对象的索引值都将递增 1。

集合中的对象可以单独使用,但一般来说,集合是作为一个整体来处理的。实际上,引入集合的目的就是高效地成批处理对象。例如,可以显示、移动或者重命名整个对象集合,对其进行排序或者调整其大小等。可使用 For Each…Next 语句逐个处理集合中所有对象。

【例 2-18】 使用 Controls 集合中的对象。

界面设计:

在窗体上添加 3 个按钮(Button 控件),Name 属性分别是 Button1、Button2、移动按钮,Text 属性分别是第 1 个控件、第 2 个控件、第 3 个控件,界面如图 2-25 所示。

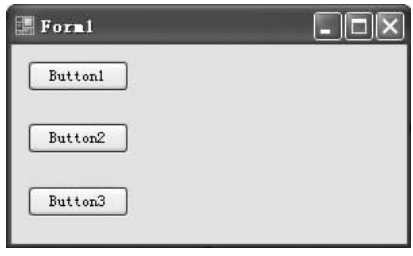

图 2-25 【例 2-18】设计界面

程序代码如下:

```
Public Class Form1
    Dim btn As Control
    Private Sub Button1_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles Button1.Click
        Dim i As Integer=3
        For Each btn In Controls
            btn.Text="第" & i & "个控件"
            i-=1
        Next
    End Sub
    Private Sub Button2_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles Button2.Click
        For Each btn In Controls
            btn.Left+=25
        Next
    End Sub
    Private Sub 移动按钮_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles 移动按钮.Click
    For Each btn In Controls
```

```
            If btn.Name <> "移动按钮" Then
                btn.Left += 25
            End If
        Next
        End Sub
    End Class
```

程序分析：

单击“第 1 个控件”按钮，3 个按钮的 Text 属性将会改变，如图 2-26 所示；单击“第 2 个控件”按钮，3 个按钮将会一起向右移动，如图 2-27 所示；单击“第 3 个控件”按钮，则除“第 3 个控件”按钮之外的其他两个按钮将会一起向右移动，如图 2-28 所示。

图 2-26　单击“第 1 个控件”按钮

图 2-27　单击“第 2 个控件”按钮

图 2-28　单击“第 3 个控件”按钮

实　　训

一、实训目的

计算机解决问题都是按照一定的算法进行的，算法是解决问题或处理事情的方法和步骤。对于求解同一个问题，常常能设计出多种不同的算法，它们的运行效率、内存使用可能有很大区别。通过本实训的练习，读者能够理解算法设计的重要性。

二、实训内容

在已经排序的数组中查找数据（数组元素值依次为 1、3、5、8、12、23、34、44、45、68），单击按钮 btnStart 后通过 InputBox 输入需要查找的数据，在 Label2 控件中输出被查找的数是否在数组中，如果不在，则在 Label2 控件中输出信息，提示被查找的数不在数组中，如果在，

则显示该数在数组中的位置。程序查找结果如图 2-29 所示。

图 2-29 查找的结果

程序代码如下：

```
Private Sub btnStart_Click(ByVal sender As System.Object,ByVal e As _
System.EventArgs) Handles btnStart.Click
    Const N=10
    Dim A(N),I,Num,Top,Bott,Min,loca As Integer
    Dim tmpstr,MyNum As String
    A(0)=1 : A(1)=3 : A(2)=5 : A(3)=8 : A(4)=12 : A(5)=23 : A(6)=34 _
: A(7)=44 : A(8)=45 : A(9)=68
    tmpstr="显示 A 元素"
    For I=0 To N-1
        tmpstr=tmpstr & A(I) & "  "
    Next I
    Label1.Text=tmpstr
    MyNum=InputBox("请输入查找数值","查找数据")
    Num=Val(MyNum)
    loca=-1
    Top=0 : Bott=N-1
    If Num<A(0)Or Num>A(N-1)Then loca=-2
    Do While loca=-1 And Top <=Bott
        Min=Int((Bott+Top)/2)
        If Num=A(Min)Then
            loca=Min
            Label2.Text=Num & "的位置在第" & loca+1 & "个。"
        ElseIf Num<A(Min)Then
            Bott=Min-1
        Else
            Top=Min+1
        End If
    Loop
    If loca=-2 Or loca=-1 Then Label2.Text="数组中无" & Num
```

```
End Sub
```

程序分析：

(1)N 为符号常量，是数组的元素个数。

(2)变量 loca 为标志，loca＝－2 表示不在数组范围内，loca＝－1 表示未找到；如果找到，则 loca 的值为查找的位置。

(3)图 2-29 显示的是输入 23 后查找的结果。

本 章 小 结

本章介绍了 Visual Basic. NET 中的数据类型，包括基本的字符型、整型、浮点型、布尔型、日期型，也有复杂的枚举型、结构型及数组和集合，还介绍了常量、变量、运算符、表达式以及常用函数，重点介绍了程序设计的 3 种基本结构，最后通过实训展示了在 Visual Basic. NET 中如何运用这些知识去解决实际问题。初学者通过学习本章内容，应该掌握 Visual Basic. NET 中的这些基本知识和 3 种基本结构的用法。

习　题　2

1. Visual Basic. NET 支持的数据类型有哪些？每种数据类型所占字节是多少？

2. Visual Basic. NET 的循环结构有几种？它们的区别是什么？

3. 已知程序段：

```
Dim a,b,c As Boolean
```

写出下面逻辑表达式的值：

```
blnResult＝a And Not b
blnResult＝a＋b＞c And b＝c
blnResult＝a or b＋c And b－c
blnResult＝Not(a＞b)And Not c or True
blnResult＝Not(a＋b)＋c－1 And b＋c/2
```

4. 写出判断 Year 是否是闰年的条件表达式。闰年的条件符合如下两者之一即可：

(1)能被 4 整除，但不能被 100 整除。

(2)能被 4 整除又能被 400 整除。

5. 试用 Visual Basic. NET 表达式表示下面各题：

(1)产生一个 11～99 的随机数。

(2)将一个两位数的个位数和十位数对换。

(3)取字符串变量 String1 的右边 4 个字符。

6. 编写一个程序，将一个一维数组中所有元素的值按逆序重新存放后输出。