# 第 2 章　编程基础


使用程序设计语言，必须熟练掌握其基本的语法规则，才能在后续内容的学习中运用自如，并减少编程时可能发生的错误。本章主要介绍 Visual Basic.NET 的数据类型、常量、变量、运算符、表达式、函数等方面的基本概念和基础知识。


通过本章的学习，应该重点掌握以下内容:

- 数据类型的概念、不同类型的数据表示方法
- 常量的定义与变量的声明
- 运算符的优先级与表达式的组成规则
- 函数的概念、常用函数的表示方法

## 2.1　数据类型

数据是程序的必要组成部分，也是程序处理的对象，程序中的数据是分属不同类型的，不同类型的数据，对其进行计算、处理的方法也不同。程序员在编程时根据需要选择适当的数据类型，可以提高程序运行的效率并节约存储空间。

Visual Basic.NET 提供的数据类型概括起来包括六大类：数值数据类型、字符数据类型、日期数据类型、布尔数据类型、对象数据类型、用户自定义数据类型。表 2-1 列出了 Visual Basic.NET 支持的数据类型。

表 2-1　Visual Basic.NET 的基本数据类型

| 数据类型 | 类型关键字 | 类型标识符 | 值类型字母 | 存储空间（字节） | 表示数的范围 |
|---|---|---|---|---|---|
| 字节型 | Byte | | | 1 | 0～255 |
| 短整型 | Short | | S | 2 | -32768～32767 |
| 整型 | Integer | % | I | 4 | -2147483648～2147483647 |
| 长整型 | Long | & | L | 8 | -9223372036854775808～9223372036854775807 |
| 单精度浮点型 | Single | ! | F | 4 | 负数：-3.402823E38～-1.401298E-45<br>正数：1.401298E-45～3.402823E38 |

续表

| 数据类型 | 类型关键字 | 类型标识符 | 值类型字母 | 存储空间（字节） | 表示数的范围 |
|---|---|---|---|---|---|
| 双精度浮点型 | Double | # | R | 8 | 负数：-1.79769313486232E308～-4.94065645841247E-324<br>正数：4.94065645841247E-324～1.79769313486232E308 |
| 货币型 | Decimal | @ | D | 16 | 小数位为 28 时：-7.922816251426433759354395033 5～7.9228162514264337593543950335 |
| 字符型 | Char | | C | 2 | 0～65535 |
| 字符串型 | String | $ | | 根据实际情况 | 字符串长度最多约 20 亿（$2^{31}$）个 Unicode 字符 |
| 布尔型 | Boolean | | | 2 | True 或 False |
| 日期型 | Date | | | 8 | 0001 年 1 月 1 日 0:00:00 到 9999 年 12 月 31 日 23:59:59 |
| 对象型 | Object | | | 4 | 可保存任何类型的数据 |
| 用户自定义类型 | | | | 根据实际情况 | 根据各个成员声明的数据类型而定 |

### 2.1.1　数值数据类型

数值数据类型包括字节型、短整型、整型、长整型、单精度浮点型、双精度浮点型和货币型。

1．字节型（Byte）

字节型是无符号整数类型，存储时占用 1 个字节（8 位），表示的是 0～255 范围的整数，不能表示负数。例如 15、126 等。

2．短整型（Short）

短整型是有符号整数类型，以 2 个字节存储，可表示的整数范围是-32768（$-2^{15}$）～32767（$2^{15}$-1）。例如-15、32126 等。

3．整型（Integer）

整型也是有符号整数类型，以 4 个字节存储，可表示的整数范围是-2147483648～2147483647。例如-1500216、2012126 等。

4．长整型（Long）

长整型也是有符号整数类型，以 8 个字节存储，可表示的整数范围是-9223372036854775808～9223372036854775807。

5．单精度浮点型（Single）

单精度浮点型是用来存储单精度浮点数的，以 4 个字节存储，其中符号占 1 位，指数占 8 位，其余 23 位表示尾数。浮点数的有效范围比 Decimal 类型的数要大得多，但可能会产生小的进位（四舍五入）误差。单精度浮点数可以精确到 7 位十进制，负数的表示范围为-3.402823E38～-1.401298E-45，正数的表示范围为 1.401298E-45～3.402823E38。例如-1.5E05、1.6235E12 等。

6. 双精度浮点型（Double）

双精度浮点型是用来存储双精度浮点数的，以 8 个字节存储，其中符号占 1 位，指数占 11 位，其余 52 位表示尾数。双精度浮点数可以精确到 15 位或 16 位十进制，负数的表示范围为 -1.79769313486232E308～-4.94065645841247E-324，正数的表示范围为 4.94065645841247E-324～1.79769313486232E308。例如 1.6123456E120。

7. 货币型（Decimal）

货币型用来存储小数，是精确小数的表示形式，以 16 个字节存储。当小数位为 0 时，可表示最大可能值为±7922816251426433759354395033 5。当小数位为 28 时，最大可表示为±7.922816251426433759354395033 5。最小非 0 数字为±0.0000000000000000000000000001。例如 0.12345。

Decimal 类型比较适合财会类的计算，可记录的数的位数很大，但又不允许出现进位（四舍五入）误差。

**说明**：对于整型数据，Visual Basic.NET 还允许使用八进制和十六进制的形式来表示。但在输出时，系统会自动把它们转换成十进制数据的形式。Visual Basic.NET 规定十六进制数必须加前缀“&H”或“&h”，八进制数必须加前缀“&O”或“&o”或“&”。例如，十进制数 17 可表示为&H11、&O21 或&21。

### 2.1.2 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。

1. 字符型（Char）

字符型用于存储单个字符，以 2 个字节存储，是单个双字节 Unicode 字符，以无符号数（0～65535）形式存储，显示时仍然是以文本符号的形式显示。例如"A"、"1"。

2. 字符串型（String）

字符串型（String）用来存放一个字符序列，一个字符串最多可以存储 20 亿（$2^{31}$）个 Unicode 字符。一个字符串的两侧要用双引号括起来，例如"Visual Basic.NET"、"计算机程序设计"等，其中长度为 0 的字符串（不包含任何字符）称为空字符串，表示形式为""。

### 2.1.3 布尔数据类型

布尔型（Boolean）用来表示 true/false、yes/no、on/off 等逻辑值信息，以 2 个字节存储，取值只有两种：True（真）或者 False（假）。

布尔型可与数值类型进行相互转换。将一个数值型数据转换成布尔型时，0 值转换成 False，非 0 值转换成 True；而将一个布尔型数据转换成数值型时，False 转换为 0，True 转换为-1。

### 2.1.4 日期数据类型

日期型（Date）数据以 8 个字节存储，可以同时或分别表示日期与时间。其日期范围为：公元 0001 年 1 月 1 日～公元 9999 年 12 月 31 日，时间范围为 0:00:00～23:59:59。Date 类型的数据要写在两个“#”之间，其中日期必须以 mm/dd/yyyy（月/日/年）的格式定义，时间必须以 hh:mm:ss（小时:分钟:秒）的格式定义，例如：

```
#2/15/2013#
#14:20:30#
```

```
#2/15/2013 14:20:30#
```

### 2.1.5　对象数据类型

对象数据类型（Object）为一个 32 位地址，用来引用应用程序或其他应用程序中的对象。可以指定一个被声明为 Object 的变量去引用应用程序所识别的任何实际对象。Object 变量也可以引用其他任何类型的数据，这个功能使 Object 类型取代了 VB6.0 中的 Variant 类型。

### 2.1.6　用户自定义数据类型

以上介绍的是 Visual Basic.Net 的基本（标准）数据类型，在程序中可以直接使用，有时用户需要根据程序的实际需要定义一些用户自定义数据类型，如数组、枚举、结构、集合等，这些数据类型将在第 5 章介绍。

## 2.2　常量和变量

在程序运行过程中，常量和变量都可以用来存储数据，它们都有自己的名字和数据类型。不同的是，在程序执行过程中，变量中存储的值是可以改变的，而常量的值则始终保持不变。

### 2.2.1　常量

常量即常数，是在程序运行过程中其值保持不变的量，它可以是任何数据类型。常量分直接常量和符号常量。

1．直接常量

直接常量是以数值、字符串或某种特定的形式直接表示的各种数据，如 2.1 节中介绍的各种类型的常数。根据数据类型的不同，直接常量可以分为数值型常量、字符串型常量、日期型常量和布尔型常量。

直接常量的类型和值由它本身的表示形式决定，不需要声明和定义，Visual Basic.NET 规定常数根据输入的形式决定保存它所使用的数据类型，默认情况下，把整数常量作为 Integer 类型处理，把小数常量作为 Double 类型处理，例如：

| | |
|---|---|
| 100 | 整型（Integer） |
| 3.14 | 双精度浮点型（Double） |
| "V" | 字符型 |
| "Visual" | 字符串型 |
| True | 布尔型 |
| #2/8/2013 10:30:00 # | 日期型 |

为了显式地指明直接常量的类型，可在其后面加上类型标识符或值类型字母（参考表 2-1），例如：

| | |
|---|---|
| 100% | 整型（Integer）常量 |
| 100& | 长整型（Long）常量 |
| 1.56! | 单精度浮点型（Single）常量 |
| 1.56# | 双精度浮点型（Double）常量 |
| 12345. 6789@ | 货币型（Decimal）常量 |

| | |
|---|---|
| 100S | 短整型（Short）常量 |
| 100I | 整型（Integer）常量 |
| 100L | 长整型（Long）常量 |
| 1.56F | 单精度浮点型（Single）常量 |
| 1.56R | 双精度浮点型（Double）常量 |
| 12345. 6789D | 货币型（Decimal）常量 |

2. 符号常量

符号常量是以标识符形式出现的常量，即用一个标识符代表一个具体的常量值，这样既方便书写又便于记忆。Visual Basic.NET 有以下两种符号常量：内部（系统定义）常量和用户自定义常量。

（1）内部常量。内部常量又称为系统常量，是由 Visual Basic.NET 提供的，这些常量可在代码中直接引用。内部常量常用“vb”作为前缀，例如：

vbCrLf：表示回车/换行字符组合。

vbCr：表示回车符。

vbTab：表示 Tab 字符。

（2）用户自定义符号常量。尽管 Visual Basic.NET 内部已经定义了大量的常量，但有时候还是需要创建自己的符号常量。用户自定义常量的语法格式如下：

```
Const 符号常量名 [As 数据类型 ] = 表达式
```

其中：

符号常量名：是有效的符号名，命名规则同变量名一样，可参考 2.2.2 节。

As 数据类型：用来说明常量的数据类型。如果省略了 As 子句，则由系统根据表达式的值确定最合适的数据类型。例如：

```
Const PI = 3.1415926          '定义了一个 Double 类型的符号常量
Const TODAY = #2/8/2013#      '表示定义了一个 Date 类型的符号常量
```

表达式：可以是各种类型的直接常量，或者是由常数和运算符组成的表达式，但不能使用函数和变量。

定义符号常量时，可以在常量名后加上类型标识符来表示符号常量的类型。例如：

```
Const NUMBER1% = 15    '相当于 Const NUMBER1 As Integer = 15
```

在程序中引用上述方法定义的符号常量时，类型标识符可以省略。即可以在程序中使用 NUMBER1 代替符号常量 NUMBER1%。

一个 Const 语句可以定义多个符号常量，中间用逗号分隔，例如：

```
Const C1 As Integer = 12, C2 As Integer = 24
```

**注意**：用 Const 定义的常量在程序运行过程中不能被重新赋值，否则将出现错误提示。

### 2.2.2 变量

在程序运行过程中，大量的数据是会随时发生变化的，这就需要使用变量来存储这些数据。例如，计算圆面积 $S=\pi R^2$，其中，圆周率 π 可以定义为一个符号常量，而圆面积 S 和半径 R 就应该使用变量来表示。

在程序运行过程中值可以随时变化的数据称为变量。一个变量相当于一个容器，对应着计算机内存中的一块存储单元，把一个数据存入变量中，就是将数据存放到变量对应的内存单元中。变量被重新赋值时，变量中的原有数据将会被新的数据覆盖。

每个变量均有属于自己的名字和数据类型。变量的名字称为变量名，程序通过变量名来引用变量的值。变量的数据类型决定该变量可以存储哪种类型的数据。

1. 变量的命名规则

在 Visual Basic.NET 中，变量的命名要遵循以下规则：

（1）变量名必须以字母、汉字或下划线“_”开头，并且只能由字母、汉字、数字和下划线“_”组合而成。建议变量名要简明、便于记忆，最好做到见名知义。

（2）不允许将 Visual Basic.NET 的关键字未作任何修改用于变量名。例如，变量名 if、Double#是不合法的，但如果改为 if123、Double1，则可以用作变量名。

（3）在同一范围内，变量名必须是唯一的。

（4）变量名最长不能超过 255 个字符。

**注意**：对于变量名，Visual Basic.NET 不区分其中所含字母的大小写，即变量 A 等价于变量 a。

在 Visual Basic.NET 中变量名、数组名、结构类型名、过程名和符号常量名都必须遵循上述规则。

2. 变量的声明

使用变量前，一般需要先声明这个变量。声明变量就是事先将变量的有关信息通知程序，包括变量名以及数据类型。声明变量通常使用 Dim 声明关键字，使用 Dim 语句声明变量的语法格式为：

```
Dim 变量名 [As 类型]
```

其中：

变量名：用户定义的标识符，应遵循变量名的命名规则。

As 类型：用来定义被声明变量的数据类型，省略时默认为 Object 类型。例如：

```
Dim number1 As Integer
Dim mystring As String
```

表示把 number1 定义为整型变量，把 mystring 定义为字符串型变量。

说明：

（1）使用 Dim 语句，Visual Basic.NET 自动将变量初始化：数值型变量赋初值为 0，字符串型变量赋初值为空串，布尔型变量赋初值为 False，日期型变量赋初值为#0:00:00#。

（2）声明变量的同时也可以直接给变量赋初值。例如：

```
Dim x As Integer=15
Dim flag As Boolean=True
```

表示把 x 定义为整型变量，并赋初值为 15；把 flag 定义为布尔型变量，并赋初值为 True。

（3）一个 Dim 语句可以同时定义多个不同类型的变量，各变量之间以逗号进行分隔。例如：

```
Dim number1 As Integer, mystring As String
```

（4）一个 Dim 语句也可以使用一个 As 子句同时定义多个同类型的变量。例如：

```
Dim a,b,c As Integer,x,y As Double, mystring As String
```

表示把 a、b、c 定义为整型变量，把 x、y 定义为双精度浮点型变量，把 mystring 定义为字符串型变量。

（5）在声明多个同类型的变量的同时不能初始化它们的内容。例如：

```
Dim a=10,b=20,c=30 As Integer
```

以上语句是错误的。

（6）声明变量时，使用的声明关键字不同以及声明变量的语句在程序中的位置不同，所定义的变量的种类和作用范围会有很大的差别。这一点将在第 6 章介绍。

3. 变量的隐式声明与显式声明

变量的隐式声明是使用变量之前不声明，在程序中使用变量时在其后跟一个类型标识符来说明该变量的数据类型。如 x%、mystring$分别声明了整型变量 x 和字符串型变量 mystring。

在默认情况下，Visual Basic.NET 编译器要求强制使用显式声明，也就是说，每个变量在使用前必须先用声明语句进行声明。程序员可以用编译器选项去掉这个限制，允许隐式声明。

方式 1：在解决方案资源管理器窗口中右击项目名称，在弹出的菜单中选择“属性”命令，出现项目属性对话框，单击“编译”选项，如图 2-1 所示，在 Option explicit 下拉列表中选择 Off（On 表示显式声明）。

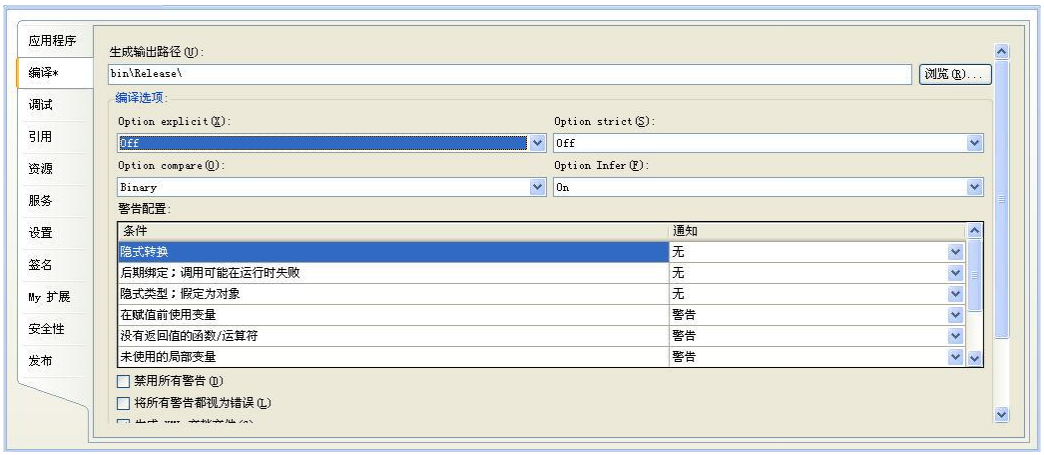


图 2-1　项目属性对话框中设置 Option Explicit 开关

方式 2：通过在代码开始处使用 Option Explicit 语句控制编译器的行为方式，该语句的语法格式为：

```
Option Explicit [on|off]
```

将 Option Explicit 语句设置为 off，表示隐式声明；设置为 on 则要求显式声明；如果省略，默认为显式声明。例如下面的代码允许使用隐式声明：

```
Option Explicit Off
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)Handles Button1.Click
        str1$ = "Visual Basic.NET"
        Label1.Text = str1
    End Sub
End Class
```

虽然通过设置可以隐式声明变量，但是建议程序员在使用变量时采用显式声明，因为这样可以减少命名冲突和拼写错误造成程序出错，从而提高程序的运行效率。

## 2.3　运算符和表达式

在计算机中，基本的运算关系可以通过一些简洁的符号来描述，这些符号称为“运算符”，

而参加运算的数据则被称为“操作数”。由运算符和操作数可构成表示各种运算关系的式子，称为表达式。

Visual Basic.NET 提供了丰富的运算符，概括起来有 6 种类型：算术运算符、赋值运算符、连接运算符、关系运算符、逻辑运算符和复合赋值运算符。

### 2.3.1　算术运算符

算术运算符是在程序中进行算术运算的符号，算术运算包括最常见的加、减、乘、除运算，以及指数运算、取模运算等，用于数值型数据的简单计算。Visual Basic.NET 提供了 8 种基本的算术运算符，如表 2-2 所示。

表 2-2　算术运算符

| 运算符 | 运算关系 | 表达式实例 | 运算结果 |
|---|---|---|---|
| + | 加法 | 4+3 | 7 |
| - | 减法 | 4-3 | 1 |
| * | 乘法 | 2*50 | 100 |
| / | 除法 | 11/2 | 5.5 |
| \ | 整除 | 11\2 | 5 |
| ^ | 指数运算 | 5^2 | 25 |
| Mod | 取模运算（求余） | 18 Mod 5 | 3 |
| - | 取负 | -a（设 a=-8） | 8 |

在表 2-2 所列的 8 个运算符当中，取负运算“-”只需要一个操作数，称为单目运算符，其他的运算符都需要两个操作数，称为双目运算符。各运算符含义与数学中基本相同。

需要重点说明的问题如下：

（1）注意“/”与“\”的区别，“/”为浮点数除法运算符，执行标准的除法运算，运算结果为浮点数。“\”为整除运算符，执行整除运算，运算结果为整数。整除时的操作数一般为整型数，当遇到非整数时，首先要对小数部分进行四舍五入取整，然后再进行整除运算。例如：

```
11/2            '结果为 5.5
11\2            '结果为 5
25\4.6          '先将 4.6 四舍五入为 5，再进行计算，结果为 5
23.9\6.3        '四舍五入后计算 24\6，结果为 4
```

（2）Mod 是取模运算，用于求两数相除的余数。运算结果的符号取决于左操作数的符号。例如：

```
16 Mod 3        '结果为 1
14 Mod 3        '结果为 2
-14 Mod 3       '结果为-2
```

优先级是指当一个表达式中存在多个运算符时，各运算符的执行顺序。算术运算符的优先顺序为：

指数运算→取负→乘法和除法→整除→取模运算→加法和减法

优先级相同时，按照从左到右的顺序执行运算。可以用括号改变运算时的优先顺序，括号内的运算总是优先于括号外的运算。

### 2.3.2 赋值运算符

赋值运算符用于赋值语句，用“=”表示，赋值运算可以将指定的值赋给运算符左侧的变量或属性。语法格式如下：

变量或属性名=表达式

赋值语句先计算表达式的值，然后再把这个值赋给左侧的变量或属性。例如：

```
Dim s As Single
s = 3.14 * 5 * 5       '先计算 3.14 * 5 * 5 得到 78.5，再把 78.5 赋给变量 s
TextBox1.Text ="Hello World!"  '把字符串"Hello World!"赋给文本框的 Text 属性
```

### 2.3.3 连接运算符

连接运算符用于将两个字符串进行连接，形成一个新的字符串。用于字符串连接运算的运算符有两个：“&”和“+”。

“&”运算符用来强制两个表达式作字符串连接。对于非字符串类型的数据，先将其转换为字符串型，再进行连接运算。

“+”运算符具备加法和连接两种运算功能。当两个表达式均为字符串时，进行连接运算。如果一个是字符串（必须能够转换为数值）而另一个是数值型数据，则先将字符串转换为数值，再进行加法运算。如果该字符串无法转换为数值，则出现错误。例如：

```
"中国" & "北京"       '结果为 "中国北京"
"中国" + "北京"       '结果为 "中国北京"
"123" & "456"         '结果为 "123456"
"123" + "456"         '结果为 "123456"
"123" & 456           '结果为 "123456"
"123" + 456           '结果为 579
123 & 456             '结果为 "123456"
"abc" +123            '出现错误
```

### 2.3.4 关系运算符

关系运算符又称比较运算符，用于对两个类型相同的数据进行比较运算。其比较的结果是一个逻辑值 True 或者 False。Visual Basic.NET 提供了多种关系运算符，如表 2-3 所示。

表 2-3 关系运算符

| 运算符 | 名称 | 例子 |
|---|---|---|
| = | 等于 | a = b |
| <> | 不等于 | a <>b |
| > | 大于 | a> b+c |
| < | 小于 | a < 5 |
| >= | 大于或等于 | x+y >=15 |
| <= | 小于或等于 | x+y <=z |
| Like | 比较模式 | "ABC" Like "A*C" |
| Is | 比较对象 | X Is Y |
| IsNot | 比较对象 | X IsNot Y |

1. 数值比较

对两个数值或算术表达式进行比较用表 2-3 中的前 6 种运算符。例如：

```
25>10
a+b>=c+d
x<>y
```

在上面的表达式中，如果关系运算符的两边的值满足关系（比较）要求结果为 True，否则结果为 False。

应避免对两个浮点数直接作“相等”或“不相等”的比较，可能会得出错误结果，这主要是因为浮点数在计算机中存储的是一个近似值（有误差）。

2. 日期比较

对日期型数据比较时，较早的日期小于较晚的日期。例如：

```
#5/12/2013#>#4/12/2013#     '结果为 True
```

3. 字符串比较

对字符型数据进行比较用表 2-3 中的前 7 种运算符。

如果比较的是单个字符，则比较两个字符的 ASCII 码值。例如：

```
"a"<"b"         '结果为 True
"A"<"a"         '结果为 True
```

默认情况下，Option Compare Binary|Text 语句的设置为 Binary，按上述 ASCII 码方式比较，如果设置为 Text，则不区分字符大小写，即"A"="a"、"B"="b"，此时按文本顺序进行比较。例如：

```
Option Compare  Text
"A" < "B"       '结果为 True
"B" < "b"       '结果为 False
```

两个字符串的比较，是按字符的 ASCII 码值将两个字符串从左到右逐个比较。即首先比较两者的第一个字符，其中 ASCII 码值较大的字符所在的字符串大，并结束比较。如果第一个字符相同，则需要比较第二个，依此类推，直到某一位置上的字符不同，或全部位置上的字符比较完毕。

Like 运算符是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，在 Like 表达式中可以使用的通配符如表 2-4 所示。

**表 2-4　匹配模式表**

| 通配符 | 含义 | 实例 | 可匹配字符串 |
|---|---|---|---|
| * | 可匹配任意多个字符 | M* | Max，Money |
| ? | 可匹配任何单个字符 | M? | Me，My |
| # | 可匹配单个数字字符 | 123# | 1234，1236 |
| [charlist] | 可匹配列表中的任何单个字符 | [b-f] | b,c,d,e,f |
| [!charlist] | 不允许匹配列表中的任何单个字符 | [!b-f] | a,g,k, … |

例如：

```
"X" Like "X"            '结果为 True
"X" Like "x"            '结果为 False
"X" Like "XY"           '结果为 False
"XY" Like "X?"          '结果为 True
```

```
"width" Like "w*h"          '结果为 True
"X" Like "[U-Z]"            '结果为 True
"X" Like "[!U-Z]"           '结果为 False
"X6Y" Like "X#Y"            '结果为 True
"X6YaW" Like "X#Y?[U-Z]"    '结果为 True
```

4. 对象比较

对两个对象进行比较时用 Is 与 IsNot 运算符。Is 运算符用来判断两个对象是否引用了同一个对象，不进行值的比较，如果两个对象变量都引用了同一个对象，结果为 True，否则为 False。IsNot 与 Is 功能相反。

### 2.3.5 逻辑运算符

逻辑运算符又称布尔运算符，对布尔型数据进行运算。逻辑运算通常用于表示复杂的关系。Visual Basic.NET 提供了多种逻辑运算符，如表 2-5 所示。

表 2-5 逻辑运算符

| 运算符 | 名称 | 例子 | 说明 |
|---|---|---|---|
| Not | 非 | Not (1 > 0) | 值为 False，由真变假或由假变真，即进行取“反”操作 |
| And | 与 | (4 > 5) And (3 < 4) | 值为 False，两个表达式的值均为 True，结果才为 True，否则为 False |
| Or | 或 | (4 > 5) Or (3 < 4) | 值为 True，两个表达式中只要有一个值为 True，结果就为 True，只有两个表达式的值均为 False，结果才为 False |
| Xor | 异或 | (4 > 5) Xor (3 < 4) | 值为 True，两个表达式的值不同时，结果为 True，否则为 False |
| AndAlso | 与 | (4 > 5) AndAlso (3 < 4) | 值为 False，两个表达式的值均为 True 时，结果为 True，否则为 False；但当第一个表达式的值为 False 时，则不再计算第二个表达式的值 |
| OrElse | 或 | (3 < 4) OrElse (4 > 5) | 值为 True，两个表达式中只要有一个值为 True，结果为 True，只有两个表达式的值均为 False，结果才为 False；但当第一个表达式的值为 True 时，则不再计算第二个表达式的值 |

用逻辑运算符将布尔变量、布尔值和其他各种表达式连接起来的式子称为逻辑表达式，逻辑表达式的值为逻辑值。逻辑运算的规则可以用真值表来表示，如表 2-6 所示。

表 2-6 逻辑运算真值表

| a | b | Not a | a And b | a Or b | a Xor b | a AndAlso b | a OrElse b |
|---|---|---|---|---|---|---|---|
| True | True | False | True | True | False | True | True |
| True | False | False | False | True | True | False | True |
| False | True | True | False | True | True | False | True |
| False | False | True | False | False | False | False | False |

当表达式中出现多个逻辑运算符时，它们的运算顺序是不同的，逻辑运算符的优先级由高到低依次是 Not、And 和 AndAlso、Or 和 OrElse、Xor。

**注意**：AndAlso、OrElse 是 Visual Basic.NET 新增的逻辑运算符，VB6.0 中 6 种逻辑运算符是 Not、And、Or、Xor、Eqv（等价）和 Imp（蕴含）。

### 2.3.6　复合赋值运算符

部分算术运算符可以和赋值运算符结合使用构成复合赋值运算符，这些运算符的功能和用法如表 2-7 所示。

表 2-7　复合赋值运算符

| 运算符 | 功能 | 示例 |
|---|---|---|
| += | 先相加再赋值 | A+=B，等价于 A=A+B |
| -= | 先相减再赋值 | A-=B，等价于 A=A-B |
| *= | 先相乘再赋值 | A*=B，等价于 A=A*B |
| /= | 先相除再赋值 | A/=B，等价于 A=A/B |
| \= | 先整除再赋值 | A\=B，等价于 A=A\B |
| ^= | 先乘方再赋值 | A^=B，等价于 A=A^B |
| &= | 先字符串连接再赋值 | S1&=S2，等价于 S1=S1 & S2 |

### 2.3.7　表达式与运算符优先顺序

表达式是程序设计语言的基本语法单位，由运算符和操作数组成。表达式中的操作数可以是常量、变量或者函数。从广义上讲，一个单独的常量、变量或者函数也可以称之为表达式。表达式本身也是有类型的，它表示了运算结果的类型。

本节前面介绍各种运算符时，已经给出了不少表达式的例子。在书写表达式时，应注意以下几点：

（1）表达式要在同一行书写。例如，分式必须采用除法运算符。即 $\frac{a+b}{c+d}$ 应写成：(a + b) / (c + d)。

（2）不要采用上下标的形式书写表达式。例如，$a_1+a_2$ 应写成：a1+a2；$2^3$ 应写成 2^3。

（3）乘号（*）不能省略。例如，5x 应写成：5 * x；a(x+y)应写成：a*(x+y)。

（4）可以使用括号改变运算顺序，但只能用圆括号，不允许使用方括号和花括号。例如，(a+b)/((x+y)*(x-y))不能写成：(a+b)/[(x+y)*(x-y)]。

当一个表达式含有多种运算符时，系统会按照 Visual Basic.NET 规定的顺序进行计算，这个顺序就是运算符的优先级。优先级高的运算符先运算，优先级低的运算符则后运算。各种运算符的优先级以及同一级运算符的运算次序如表 2-8 所示。

表 2-8　运算符的优先级

| 运算符 | 优先级 | 运算次序 |
|---|---|---|
| () | 1 | 由内向外 |
| ^ | 2 | 由内向外 |
| -（取负） | 3 | 由内向外 |
| *　/ | 4 | 从左至右 |

续表

| 运算符 | 优先级 | 运算次序 |
|---|---|---|
| \ | 5 | 从左至右 |
| Mod | 6 | 从左至右 |
| + - | 7 | 从左至右 |
| & | 8 | 从左至右 |
| = > < >= <= <> Like Is | 9 | 从左至右 |
| Not | 10 | 从左至右 |
| And AndAlso | 11 | 从左至右 |
| Or OrElse | 12 | 从左至右 |
| Xor | 13 | 从右至左 |
| = += -= *= /= \= ^= &= | 14 | 从右至左 |

**提示：**用户可以使用 Visual Basic.NET 集成开发环境中的“命令窗口”测试表达式的值。在“命令窗口”中，输入“？”后跟一个表达式，输入完后按回车键，Visual Basic.NET 会给出表达式的结果或错误提示信息，如图 2-2 所示。

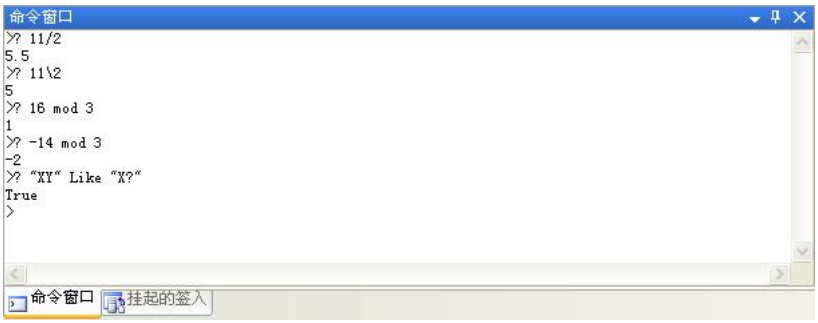


图 2-2 命令窗口

**提示：**打开“命令窗口”的方法是，选择“视图”→“其他窗口”→“命令窗口”。

## 2.4 常用内部函数

在编程时用户可以直接使用 Visual Basic.NET 的函数库或.NET 框架基础类库中各个类中的函数来完成相应的计算，这些由系统本身提供的、用户可直接使用的函数称为内部函数（或库函数、标准函数）。函数是一种特定的运算，在程序中要使用一个函数时，只要给出函数名和相应的参数，就能得到它的函数值。函数的使用格式为：

函数名[(参数表)]

**说明：**

（1）参数表中的参数可以有一个或者多个（函数本身的要求），多个参数之间以逗号进行分隔。

（2）方括号表示可选部分。对于没有参数的函数，只需书写函数名，括号可以省略。

（3）函数调用时，参数可以是常量、变量、表达式，也可以是函数。

Visual Basic.NET 包括两类函数：内部函数和用户定义函数。

内部函数也称标准函数，可大体分为数学函数、字符串函数、转换函数、判断函数、日期函数等几大类。用户自定义函数是程序员根据需要自行开发和定义的函数，将在本书 6.2 节中详细介绍。

利用函数可以大大增强程序的表达能力和开发能力。

### 2.4.1　算术函数

Visual Basic.NET 中的数学函数在 System 命名空间下的 Math 类中，常用的函数和功能如表 2-9 所示。函数的调用形式如下：

```
System.Math.函数名(参数表)
```

如果在程序的开头加上了 Imports 语句：Imports System.Math，在程序中使用数学函数时也可以直接写函数名。

表 2-9　常用数学函数

| 函数名 | 功能 | 示例 |
|---|---|---|
| Sin(x) | 求 x 的正弦值，x 为弧度 | Sin(30*3.141593/180) 结果为 0.50000005000000569 |
| Cos(x) | 求 x 的余弦值，x 为弧度 | Cos(3.141593/4) 结果为 0.70710671994929331 |
| Tan(x) | 求 x 的正切值，x 为弧度 | Tan(0) 结果为 0 |
| Atn(x) | 求 x 的反正切值，x 为弧度 | Atn(0) 结果为 0 |
| Sqrt(x) | 求 x 的平方根 | Sqrt (25) 结果为 5 |
| Abs(x) | 求 x 的绝对值 | Abs(-3.3) 结果为 3.3 |
| Sign(x) | 判断 x 的符号，若 x>0，返回值为 1；若 x<0，返回值为-1；若 x=0，返回值为 0 | Sgn(6) 结果为 1 |
| Exp(x) | 求以 e 为底的指数（$e^x$） | Exp(1) 结果为 2.7182818284590451 |
| Log(x) | 求 x 的自然对数（ln x） | Log(1) 结果为 0 |
| Log10(x) | 求以 10 为底的 x 的常用对数 | Log10(10)结果为 1 |
| Pow(x,y) | 求 $x^y$ | Pow(5,2)结果为 25 |
| Round(x) | 对 x 进行四舍五入取整 | Round(6.5) 结果为 6 |
| Floor(x) | 取整，返回小于或等于 x 的最大整数 | Floor(-5.2) 结果为-6，Floor(5.5) 结果为 5 |
| Ceiling(x) | 取整，返回大于或等于 x 的最小整数 | Ceiling(-5.2) 结果为-5，Ceiling(5.5) 结果为 6 |
| Max(x,y) | 返回 x 和 y 中的较大者 | Max(5,8) 结果为 8 |
| Min(x,y) | 返回 x 和 y 中的较小者 | Min(5,8) 结果为 5 |

命名空间（NameSpace）是.NET 中的各种语言使用的一种代码组织形式。命名空间是用来组织和重用代码的编译单元。Visual Basic.NET 是完全面向对象的语言，应用程序的所有代码均封装在各个类中。不同的人编写的程序不可能所有的类都没有重名现象，并且对于类库来说，这个问题尤其严重，为了解决这个问题，引入了命名空间这个概念。通过命名空间把类库划分为不同的组，将功能相近的类划分到相同的命名空间。通过使用命名空间，你所使用的库函数或类就是在该命名空间中定义的，这样一来就不会引起不必要的冲突了。实际上一个应用程序的所有代码都被包含在某些命名空间中。

在对命名空间中类的方法（函数）使用时可以采用完全限定名形式：

命名空间.类.方法（函数）名

例如 System.Math. Sqrt (25)求 25 的平方根。

由于采用完全限定名形式过于繁琐，可以在程序开始使用 Imports 语句导入程序需要的命名空间甚至命名空间下的类，这样程序中就可以采用不完全限定名形式，直接写方法（函数）名。

### 2.4.2 字符串函数

Visual Basic.NET 中的字符串处理函数在 Microsoft.VisualBasic 命名空间下的 Strings 类中，常用的函数和功能如表 2-10 所示。

表 2-10 常用字符串函数

| 函数名 | 功能 | 示例 |
|---|---|---|
| LTrim(s) | 删除字符串 s 左端的空格 | LTrim("　ABC")结果为"ABC" |
| RTrim(s) | 删除字符串 s 右端的空格 | RTrim("ABC　")结果为"ABC" |
| Trim(s) | 删除字符串 s 两端的空格 | Trim("　123　")结果为"123" |
| Left(s, n) | 截取字符串 s 左端的 n 个字符，生成子串 | Left("ABC123",4) 结果为"ABC1" |
| Right(s, n) | 截取字符串 s 右端的 n 个字符，生成子串 | Right("ABC123",4) 结果为"C123" |
| Mid(s, m, n) | 从字符串 s 的第 m 个字符位置开始，取出 n 个字符 | Mid("ABC123",2,3) 结果为"BC1" |
| InStr([start],s1,s2 [, 比较方式]) | 从 s1 的 start 位置开始查找 s2,若找到则返回 s2 在 s1 中出现的位置，否则返回 0 值。 | InStr(2,"asdfasdf","asdf")结果为 5 |
| Len(s) | 求字符串 s 的长度（字符数） | Len("人数 1234")结果为 6 |
| Space(n) | 返回由 n 个空格组成的字符串 | "A" + Space(3) + "B"结果为"A　B" |
| StrDup(n, 字符串) | 返回由 n 个字符串首字母组成的字符串 | StrDup(3, "About")结果为"AAA" |
| UCase(s) | 将 s 中的小写字母转换为大写，其余不变 | UCase("About")结果为"ABOUT" |
| LCase(s) | 将 s 中的大写字母转换为小写，其余不变 | LCase("About")结果为"about" |
| Chr(n) | 将 ASCII 码值 n 转换成字符 | Chr(65)结果为"A" |
| Asc(s) | 将字符串 s 中的第一个字符转换为 ASCII 码值 | Asc("BCD")结果为 66 |
| StrComp(字符串 1，字符串 2[,比较方式]) | 比较字符串 1 和字符串 2 | StrComp("ABCD","BDC")结果为-1 |
| StrReverse(字符串) | 将字符串反序 | StrReverse("ABCD")结果为"DCBA" |
| Replace(字符串 1，字符串 2，字符串 3) | 将字符串 1 中与字符串 2 相同的部分替换为字符串 3 | Replace("ABCD","BC","XXX") 结 果为"AXXXD" |

说明：

（1）InStr 函数用来确定 s2 在 s1 中第一次出现的位置，找不到时返回值为 0。Start 用来指定起始位置，省略时默认为 1。“比较方式”用来指定字符串的比较方式，即是否区分字母的大小写。0 表示区分大小写，1 表示不区分大小写，省略时按 Option Compare 语句指定的方式进行比较。

（2）StrComp 函数用来求两个字符串的比较结果。若字符串 1 大于字符串 2，结果为 1；

若字符串 1 小于字符串 2，结果为-1；两个字符串相等，则结果为 0。“比较方式”可以取 0 或 1，0 表示区分大小写，1 表示不区分大小写。

### 2.4.3　日期与时间函数

Visual Basic.NET 中的日期与时间函数在 Microsoft.VisualBasic 命名空间下的 DateAndTime 类中，常用的函数和功能如表 2-11 所示。

表 2-11　常用日期时间函数

| 函数名 | 功能 | 示例 |
|---|---|---|
| Now 或 Now() | 返回系统当前的日期和时间 | Now 结果为#6/15/2012 15:18:16# |
| Hour(D) | 返回时间中的钟点数 | Hour(Now())结果为 15 |
| Minute(D) | 返回时间中的分钟数 | Minute (Now())结果为 18 |
| Second(D) | 返回时间中的秒数 | Second (Now())结果为 16 |
| Today 或 Today() | 返回系统当前的日期 | Today()结果为#6/15/2012# |
| Year(D) | 返回日期中的年份数 | Year(Today())结果为 2012 |
| Month(D) | 返回日期中的月份数 | Month(Today())结果为 6 |
| Day(D) | 返回日期中的日期数 | Day(Today())结果为 15 |
| WeekDay(D) | 返回指定日期对应的星期数，1 代表星期日，2 代表星期一，依次类推 | WeekDay(#6/15/2012#)结果为 6 |
| TimeOfDay() | 返回系统当前的时间 | TimeOfDay()结果为#11:50:31 AM# |
| Timer 或 Timer() | 从零时起到现在经历过的秒数 | Timer()结果为 42669.765625 |

### 2.4.4　转换函数

Visual Basic.NET 中的转换函数在 Microsoft.VisualBasic 命名空间下的 Conversion 类中，常用的函数和功能如表 2-12 所示。

表 2-12　常用转换函数

| 函数名 | 功能 | 示例 |
|---|---|---|
| Val(x) | 将字符串 x 转换成对应的数值 | Val("-123.45")结果为-123.45 |
| Str(x) | 将数值转换成对应的字符串 | Str(123.45)结果为"123.45" |
| Fix(x) | 对数值 x 取整，截去小数部分 | Fix(5.6)结果为 5，Fix(-5.6)结果为-5 |
| Int(x) | 返回不大于 x 的最大整数 | Int(5.6)结果为 5，Int(-5.6)结果为-6 |
| Hex(x) | 将十进制数 x 转换为对应的十六进制数的字符串形式 | Hex(126)结果为"7E" |
| Oct(x) | 将十进制数 x 转换为对应的八进制数的字符串形式 | Oct(126)结果为"176" |

说明：

（1）Val 函数可将数字字符串转换为数值，当遇到非数字字符时，结束转换。例如，Val("a1")返回 0，Val("1a1")返回 1。但有以下两种特殊情况：

- 转换时忽略数字之间的空格。例如，Val("12　34")返回数值 1234。

- 能识别指数形式的数字字符串，例如 Val("1.234e2")或者 Val("1.234d2")都可得到数值 123.4。其中的字母也可以是大写的 E 或者 D。

（2）Str 函数将数值转换成对应的字符串，数值为负数时，结果为直接在数值两端加上双引号，如：Str (-123.45)结果为"-123.45"；数值为正数时，结果为在数值前面空一格（正号的符号位）两端再加上双引号，如 Str (123.45)结果为" 123.45"。

### 2.4.5 数据类型转换函数

在一个表达式中，如果参加运算的数据类型不同，要先将不同的数据类型转换成同一类型，然后进行运算。类型转换可隐式或显式地进行，隐式转换不需要在源程序代码中使用任何特殊语法。例如：

```
Dim x As Byte
Dim y As Integer
y=15
x=y
```

此时系统自动将 Integer 类型的变量 y 的值隐式转换为 Byte 类型，然后赋值给变量 x。

显示转换需要用 Visual Basic.NET 提供的数据类型转换函数来实现，但需要注意的是被转换的数值的大小必须在转换以后的数据类型的表示范围以内。常用的数据类型转换函数和功能如表 2-13 所示。

表 2-13 常用数据类型转换函数

| 函数名 | 功能 | 示例 |
|---|---|---|
| CBool(x) | 将 x 的值强制转换为 Boolean 类型 | CBool(12) 结果为 True<br>CBool(0) 结果为 False |
| CByte(x) | 将 x 的值强制转换为 Byte 类型，当个位数为奇数时，小数部分的第 1 位进行四舍五入 | CByte(2.5) 结果为 2<br>CByte(3.5) 结果为 4 |
| CShort(x) | 将 x 的值强制转换为 Short 类型，当个位数为奇数时，小数部分的第 1 位进行四舍五入 | CShort(2.5) 结果为 2<br>CShort(3.5) 结果为 4 |
| CInt(x) | 将 x 的值强制转换为 Integer 类型，当个位数为奇数时，小数部分的第 1 位进行四舍五入 | CInt(2.5) 结果为 2<br>CInt(3.5) 结果为 4 |
| CLng(x) | 将 x 的值强制转换为 Long 类型，当个位数为奇数时，小数部分的第 1 位进行四舍五入 | CLng(2.5) 结果为 2<br>CLng(3.5) 结果为 4 |
| CSng(x) | 将 x 的值强制转换为 Single 类型 | CSng(3) 结果为 3.0 |
| CDbl(x) | 将 x 的值强制转换为 Double 类型 | CDbl(3) 结果为 3.0 |
| CDec(x) | 将 x 的值强制转换为 Decimal 类型 | CDec(3.65) 结果为 3.65D |
| CStr(x) | 将 x 的值强制转换为 String 类型 | CStr(12.345) 结果为"12.345" |
| CDate(x) | 将 x 的值强制转换为 Date 类型 | CDate("6-18-2012") 结果为#6/18/2012# |
| CType(变量名，数据类型) | 对变量作数据类型转换 | CType(x，Double)将变量 x 转换为 Double 类型 |

Visual Basic.NET 能用这些函数强制将表达式转换为目标数据类型。例如：

```
Dim a As Double
a=12.3
```

```
b=Cint(a)
c=Ctype(a,Integer)
```

以上两种方式都可以将变量 a 由 Double 类型转换为 Integer 类型，再赋值给 b 和 c。

### 2.4.6　随机函数

Visual Basic.NET 中的随机函数在 Microsoft.VisualBasic 命名空间下的 VBMath 类中，常用的函数有 Rnd 和 Randomize。

1. Rnd 函数

随机函数 Rnd 用来产生一个 0～1 之间的随机数（不包括 0 和 1），格式如下：

```
Rnd[(x)]
```

其中，x 是可选参数，x 的值将直接影响随机数的产生过程。当 x<0 时，每次产生相同的随机数。当 x>0（系统默认值）时，产生与上次不同的新随机数。当 x=0 时，本次产生的随机数与上次产生的随机数相同。

例如：

```
Debug.Print(Rnd(-1) & vbTab  & Rnd(-1))      'x < 0，每次产生相同的随机数
```

结果可能为：0.224007　　0.224007

```
Debug.Print(Rnd(1) & vbTab  & Rnd(2))        'x > 0，每次产生不同的随机数
```

结果可能为：0.03584582　0.08635235

```
Debug.Print(Rnd(1) & vbTab  & Rnd(0))        'x = 0，第 2 个随机数与第 1 个相同
```

结果可能为：0.1642639　　0.1642639

Debug 是 Visual Basic.NET 系统的内部的一个类，它的 Print 方法表示在 Visual Basic.NET 集成开发环境中的输出窗口中输出信息。

**提示**：单击“视图”菜单中“输出（O）”可打开输出窗口，也可单击“调试”菜单中“窗口”再选择“输出（O）”可打开输出窗口。

Rnd 函数产生的随机数为单精度数，若要产生随机整数，可利用取整函数来实现。例如：要产生 100～200 之间的随机整数，可用如下表达式来实现。

```
System.Math.Floor(Microsoft.VisualBasic.VBMath.Rnd() * 101 + 100)
```

用 Rnd 函数得到的“随机数”是通过一个“随机化公式”计算出来的。此公式中有一个参数 r 称为“随机化种子”，将 r 和公式中其他的常数进行四则运算，得到一个数，这就是第一个随机数，以上过程可以表示为：r → f(r)。

f(r)是一个以 r 为自变量的函数，r 即为“随机化种子”。给定一个初始的 r 就能计算出一个随机数 f(r)，然后将 f(r)赋给 r，再用这个新的 r 作为随机化种子，代入 f(r)，通过随机化公式计算出一个新的随机数，再将它赋给 r，再计算 f(r)，……，如此不断迭代，可得到一个随机数序列。

从上面的叙述可以看到，只要“随机化种子”r 的初值相同，则每次产生的随机数序列总是相同的，并不能真正做到随机化。为了解决这个问题，就需要在每次运行程序时指定不同的“随机化种子”。

2. Randomize 函数

Visual Basic.NET 提供了一个随机化语句，它的作用是产生新的随机化种子，格式为：

```
Randomize [(表达式)]
```

其中，“表达式”为可选参数。若指定参数，Visual Basic.NET 将产生一个与该表达式对应

的随机数序列，如：Randomize 5，表示指定随机化种子 r 的值为 5。若省略参数，Visual Basic.NET 取内部计时器的值作为新随机数的“种子数”。由于计算机内部的时钟是不断变化的，故每次的种子数不同，从而可产生出不同的随机数序列。

## 2.5 Visual Basic.NET 基本语句格式

Visual Basic.NET 程序中的一行代码称为一条语句。语句可以由 Visual Basic.NET 的关键字、属性、函数、运算符，以及能够生成 Visual Basic.NET 编辑器可识别指令的符号任意组合而成。一个完整的语句可以只有一个关键字，也可以是各元素的组合。在编写程序语句时要遵循一定的规则，也就是语法。在输入语句的过程中，Visual Basic.NET 自动对输入内容进行语法检查，如果发现错误，将作出标示，把鼠标移至标示处，会提示出错原因。Visual Basic.NET 还会对语句进行简单的格式化处理，例如将关键字、函数的第一个字母转换为大写。在编写程序语句时要遵循的规则有：

- 一个语句行以回车键结束，其长度最多不能超过 1023 个字符。
- 语句中的命令字母不分大小写。
- 一行上可以书写一至多条语句，当在一行上书写多条语句时，语句间要用半角英文冒号“:”隔开，例如：

```
t=a: a=b: b=t          '表示本行中书写了三条语句
```

为了提高程序的可读性，常在程序的适当位置上对程序代码进行必要的说明，这就是注释。

格式 1：Rem 注释内容

例如：

```
Rem 求圆面积
c = 3.14 * r* r
```

格式 2：'注释内容

```
c = 3.14 * r* r            '求圆面积
```

- 当一条语句很长时，也可以断成若干行来写，但这时要在每行的断开处末尾加上空格及下划线“_”作为续行标志，表示下一行与本行是同一条语句。例如：

```
c = 2 * 3.14 _
* r
```

- 为方便阅读程序，建议一行上只写一条语句。

**关键字**：关键字是 Visual Basic.NET 事先定义的，有特殊意义的标识符，有时又叫保留字。例如：Const、Dim、As、Mod、And、Or、Not、If、Then、Else 等。

## 实验二 Visual Basic.NET 语言基础练习

### 一、实验目的

通过本次实验，能够了解并掌握常量和变量的声明方法及命名规则、命令窗口的使用方法、运算符的作用及表达式的组成，了解 Visual Basic.NET 中函数的概念，掌握函数的一般使用方法。

## 二、实验内容与步骤

1. 常量和变量的声明

设计步骤如下：

（1）建立应用程序用户界面。

新建一个项目，在窗体中增加一个命令按钮 Button1，将其 Text 属性设为“显示”，再增加 4 个标签 Label1～Label4。

（2）设计代码。

编写命令按钮 Button1 的 Click 事件代码：

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles Button1.Click
        Dim string1 As String                    '声明字符串型变量
        Dim a%, b!                               '声明数值型变量
        Dim day1 As Date                         '声明日期型变量
        Const PI As Double = 3.1415              '声明常量
        string1 = "学习 VB，要多读程序，多上机练习"
        Label1.Text = string1
        a = 2
        b = 3.45
        Label2.Text = "a+b=" & (a + b)
        day1 = Now()
        Label3.Text = day1
        Label4.Text = PI
    End Sub
End Class
```

（3）运行程序，单击命令按钮，查看运行结果。

2. 命令窗口的使用

选择“视图”菜单中的“其他窗口”再选择“命令窗口”，打开“命令窗口”。在“命令窗口”中，输入“？”后跟一个表达式，输入完后按回车键 Visual Basic.NET 会给出表达式的结果或错误提示信息。如图 2-2 所示。

请仿照图 2-2，在命令窗口中输入若干表达式，并查看输出结果。

3. 表达式的运算规则

打开命令窗口，键入如下代码：

```
? 8 Mod 3        '注意：运算符 Mod 两端要留空格
```

按回车键，将在命令窗口中看到 8 Mod 3 的结果为 2。用同样的方法分别输入如下代码，先估算一个结果，再与命令窗口中的实际输出进行比较。

? 6 / 8　　结果为：_____

? 25 \ 7.8　　结果为：_____

? "欢迎使用" + "Visual Basic"　　结果为：_____

? "11" + "22"　　结果为：_____

? 11 + "22"　　结果为：_____

? "11" & "22"　　结果为：_____

? "True" & "False"　　结果为：_____

4. 常用函数的使用

打开“命令窗口”，键入如下代码：

? System.Math.Sin(31 / 180 * 3.14)　　结果为：_____

? System.Math.Sqrt(12)　　结果为：_____

? System.Math.Abs(-3.5)　　结果为：_____

? Microsoft.VisualBasic.VBMath.Rnd(1)　　结果为：_____

? Microsoft.VisualBasic.Conversion.Int(-3.6)　　结果为：_____

? Microsoft.VisualBasic.Conversion.Fix(-3.6)　　结果为：_____

? System.Math.Sign(-3.6)　　结果为：_____

? Microsoft.VisualBasic.Strings. Asc("A")　　结果为：_____

? Microsoft.VisualBasic.Strings. Len(Microsoft.VisualBasic.Strings. Chr(70))

结果为：_____


假设以下各题中用到的函数已引入了相应的命名空间。

**一、选择题**

1．设 x="Visual Basic.net"，下面使 y="Basic"的语句是（　　）。

A．y=Left(x,8,5)　　B．y=Mid(x,8,5)

C．y=Right(x,5,5)　　D．y=Left(a,8,5)

2．设有声明语句 Dim x As Integer，如果 Sign(x)的值为 1，则 x 的值是（　　）。

A．正整数　　B．负整数　　C．0　　D．任意整数

3．以下关系表达式中，其值为 False 的是（　　）。

A．"ABC">"AbC"　　B．"the"<>"they"

C．"VISUAL"=UCase("Visual")　　D．"Integer">"Int"

4．产生[1,100]区间的数的随机函数表达式正确的为（　　）。

A．Rnd() * 99 + 1　　B．Int(Rnd() * 99+ 1)

C．(Rnd() * 90) + 10　　D．Int(Rnd() * 100) + 1

5．下列选项中，合法的变量名是（　　）。

A．Byval　　B．123book　　C．String　　D．Sum_2

6．下列声明语句中错误的是（　　）。

A．Const var1=123　　B．dim a:b as integer

C．dim a,b as string　　D．dim var3 as integer

7．设 x=4，y=8，z=7，则表达式 x<y And (Not y>z) Or z<x 的值是（　　）。

A．True　　B．False　　C．1　　D．-1

8．以下表达式的结果（　　）不是字符串类型。

A．"45" + "123"　　B．"45" & "123"

C．45 + "123"　　D．全部

## 二、填空题

1．算术表达式 3-5*7 MOD 2^3 的运算结果是_____。

2．算术表达式 4+2*3^2\2*4 的运算结果是_____。

3．假设变量 a=1,b=2,c=3，则逻辑表达式 a+b>c And b=c 的值是_____。

4．表达式"a2a"　Like　"a#a" 的值为_____，表达式 16\5 的值为_____。

5．数学表达式 a<=x<=b 在 VB.NET 中应写成_____。

6．在 Visual Basic.NET 中，字符型常量应使用_____将其括起来，日期型常量应使用_____符号将其括起来。

7．可获得系统当前日期和时间的函数分别为_____和_____。

8．定义符号常量应使用_____关键字。

9．VB.NET 变量的命名规则要求，变量名必须以_____开头。

10．表达式 Fix(-100.45)+Int(-100.45)-Sign(-100.45)+Val("-100")的值是_____。

11．要引入 System 命名空间下的 Math 类，则应在程序代码中书写的语句是_____。

12．如果 Option Compare 选项设置为 Binary，则 A，a，W 3 个字母从小至大的排序顺序为_____。

## 三、把下列数学表达式转换成等价的 Visual Basic.NET 表达式：

（1）$\frac{-b+\sqrt{b^2-4ac}}{2a}$

（2）$\cos^2(31^0)+5e^3$

（3）$\frac{x+y+z}{\sqrt{x^3+y^3+z^3}}$

（4）$8e^x \ln 10$

（5）$2\sin(\frac{x+y}{2})\cos(\frac{x-y}{2})$

（6）$\frac{x^2+y^2}{2a^2}$

（7）$\sqrt[3]{x}\sqrt[4]{y}$

（8）$\frac{(3+a)^2}{2c+4d}$

## 四、计算下列表达式的值（可在上机时验证）

（1）3 * 4 * 5 \ 2

（2）7 / 6 * 3 * (-4.3 + Abs(4.3)) \ 2.6

（3）14 / 4 * 2 ^ 3 \ 1.6

（4）58 \ 3 Mod 2 * Int(3.7)+ 25 \ 7.7 + 27.9 \ 5.4

（5）Exp(0) + Len("我爱祖国")

（6）Cos(0) + Int(-3.6) + Abs(Fix(-3.6)) + Sign(Rnd(-3.6))

（7）Asc("A") & UCase(Mid("voice", 1, 1)) & Chr(66)