第 2 章　Visual Basic.NET 程序设计基础

【本章导读】

本章首先介绍 Visual Basic.NET 基本数据类型、变量、常量、运算符和表达式；然后介绍程序基本控制结构和语句；最后介绍 Visual Basic.NET 程序代码编写规则，并通过两个简单的程序使读者对.NET 程序开发有一个初步的了解。

【本章要点】

- 数据类型、变量、常量、运算符、表达式
- 各种语句和程序控制结构
- 代码编码规则
- 程序基本结构

2.1　Visual Basic.NET 简介

BASIC 是 Beginner's All-purpose Symbolic Instruction Code（初学者通用符号指令代码）的缩写，是国际上广泛使用的一种计算机高级语言。BASIC 简单、易学，目前仍是计算机入门的主要学习语言之一。

BASIC 语言自其问世经历了以下五个阶段：

第一阶段（1964 年～70 年代初）：1964 年 BASIC 语言问世。

第二阶段（20 世纪 70 年代初～80 年代中）：微机上固化了 BASIC。

第三阶段（20 世纪 80 年代中～90 年代初）：推出了结构化 BASIC 语言。

第四阶段（1991 年～2002 年）：推出了 Visual Basic。

第五阶段（2002 年以后）：推出了 Visual Basic.NET。

1991 年，Microsoft 推出了 Visual Basic 1.0 版，在当时引起了很大的轰动。许多专家把 Visual Basic 的出现当作软件开发史上的一个具有划时代意义的事件。其实，以现在的目光来看，Visual Basic 1.0 的功能实在是太弱了，但它是第一个“可视”的编程软件，因此很多程序员都尝试在 Visual Basic 的平台上进行软件创作。此后 Microsoft 在四年内连续推出 Visual Basic 2.0、Visual Basic 3.0、Visual Basic 4.0 等 3 个版本。从 Visual Basic 3.0 开始，Microsoft 将 Access 的数据库驱动集成到了 Visual Basic 中，这使 Visual Basic 的数据库编程能力大大提高；从 Visual Basic 4.0 开始引入了面向对象的程序设计思想；Visual Basic 还引入了“控件”的概念，使大量已经编好的 Visual Basic 程序可以被直接重用，Visual Basic 5.0 版本引入 ActicveX 的概念，允许开发人员创建自己的 ActiveX 控件；Visual Basic 6.0 集成了 ActiveX Data Objects（ADO），提供了一种访问数据库的全新方法；2002 年 2 月，随着 Visual Basic.NET 的发布，许多原有的局限性被消除，变成了功能非常强大的开发工具。

2.2 基本数据类型

2.2.1 常用基本数据类型

数据是程序的主要组成部分，也是程序处理的对象。在高级语言中，广泛使用“数据类型”这一概念，数据类型体现了数据结构的特点。Visual Basic.NET 提供了系统定义的基本数据类型，并允许用户根据需要定义自己的数据类型。表 2-1 列出了 Visual Basic.NET 中定义的全部基本数据类型。

表 2-1 基本数据类型

数据类型	关键字	存储空间	取值范围
字节型	Byte	1 个字节	0～255（无符号）
		占用的内存空间最少	
字符型	Char	2 个字节	0～65535（无符号）
		不能直接转换为数值类型，必须通过 Asc()、AscW()完成转换	
布尔型	Boolean	2 个字节	True 或 False
		以数值形式存储，非零为 True，零为 False	
日期时间型	Date	8 个字节	0001 年 1 月 1 日凌晨 0:00:00 到 9999 年 12 月 31 日晚上 11:59:59
		以#为定界符，格式为 m-d-yyyy。例如#1-20-2003#	
短整型	Short	2 个字节	-32768～32767
		可以十、八、十六进制表达，使用后两种进制时应分别加前缀&O、&H	
整型	Integer	4 个字节	-2147483648～2147483647
		可以十、八、十六进制表达，使用后两种进制时应分别加前缀&O、&H	
长整型	Long	8 个字节	-9223372036854775808～9223372036854775807
		可以十、八、十六进制表达，使用后两种进制时应分别加前缀&O、&H	
单精度浮点型	Single	4 个字节	负值取值范围为-3402823E+38～-1401298E-45；正值取值范围为 1401298E-45～3402823E+38
		有效位数为 7 位	
双精度浮点型	Double	8 个字节	负值取值范围为-179769313486231E+308～-494065645841247E-324；正值取值范围为 494065645841247E-324～179769313486231E+308
		有效位数为 15 位	
数值型	Decimal	16 个字节	0 到+/-79228162514264337593543950335 之间不带小数点的数；0 到+/-79228162514264337593543950335 之间带 28 位小数的数；最小非零数为+/-0.0000000000000000000000000001(+/-1E-28)
		有效位数为 28 位	
字符串型	String	取决于实现平台	0 到大约 20 亿个 Unicode 字符
		一个半角字符占用 1 个字节，一个全角字符占用 2 个字节	
对象型	Object	4 个字节	可以存储任何类型的数据
用户自定义类型		取决于实现平台	结构中的每个成员都有由自身数据类型决定的取值范围，并与其他成员的取值范围无关
		用 Structure…End Structure 定义	

顺便指出，Byte、Short、Integer、Long、Single、Double、Decimal 均可归类为数值类型，当将布尔型数据转换为数值类型时，True 会被转换为-1，False 会被转换为 0；在将数值数据类型转换为 Boolean 值时，0 会转换为 False，而其他所有值都将转换为 True。

2.2.2　值类型和引用类型

在 Visual Basic.NET 中，数据类型分为值类型和引用类型两种，值类型的数据存储在内存中的堆栈上，每个变量都有自己的堆栈地址。当一个值类型变量的值赋值给另一个相同类型的变量时，数据在堆栈上进行了一次数据拷贝操作，数据占用不同的内存地址，所以一个值类型的变量改变后不会影响另一个变量。引用类型的数据由两部分组成：一部分是在堆中存储的数据，另一部分是在堆栈中存储的指向该数据存储位置的指针（内存地址）。当数据从一个引用类型的变量赋值给另一个相同类型的变量时，只是在堆栈上进行了一次指向堆中数据地址的拷贝，而在堆上没有进行任何数据拷贝的操作。

例如：赋值语句：Dim x As Integer = 5，变量 x 在内存中的存储情况如图 2-1 所示。

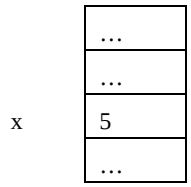

图 2-1　值类型 x 在内存存储的示意图

下面两句代码：

```
Dim x As Integer = 5
Dim y As Integer = x
```

变量 x、y 在内存中的存储情况如图 2-2 所示。

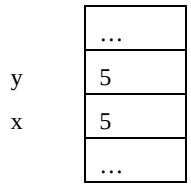

图 2-2　值类型 x、y 在内存存储的示意图

引用类型数据在内存中的存储方式和值类型不同，例如 Dim name1 As String = "张三"，name1 在内存的存储示意图如图 2-3 所示。

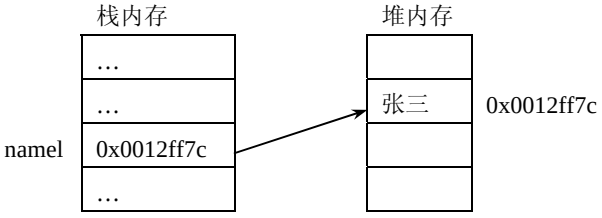

图 2-3　引用类型 name1 在内存中存储的示意图

下面两句代码：

```
Dim name1 As String = "张三"
Dim name2 As Integer = name1
```

变量 name1 和 name2 在内存中的存储的示意图如 2-4 所示。

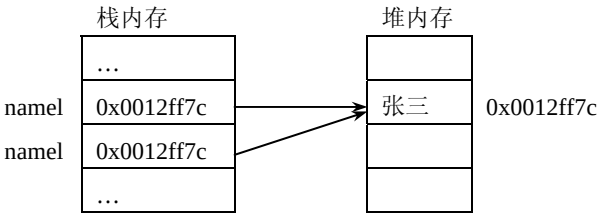

图 2-4　引用类型 name1 和 name2 在内存中存储的示意图

顺便指出：String 是特殊的引用类型，它的很多方法被重写过，所以 String 有很多值类型的特点。

如表 2-2 所示是值类型和引用类型之间的比较。

表 2-2　值类型和引用类型的比较

	值类型	引用类型
内存分配地点	分配在栈中	分配在堆中
效率	效率高，不需要地址转换	效率低，需要进行地址转换
内存回收	使用完后，立即回收	使用完后，等待系统自动回收
赋值操作	复制并创建同值新对象	对原有对象的引用
函数参数与返回值	是对象的复制	是原有对象的引用，并不产生新的对象
类型扩展	不易扩展	方便与类型扩展

Visual Basic.NET 中，值类型和引用类型分类表如表 2-3 所示。

表 2-3　值类型和引用类型分类表

类别		描述
值类型	简单类型	有符号整型：Sbyte、Short、Int、Long
		无符号整型：Byte、Ushort、Uint、Ulong
		Unicode 字符：Char
		IEEE 浮点型：Float、Double
		高精度小数：Decimal
		布尔型：Bool
	枚举类型	用户自定义类型：Enum
	结构类型	用户自定义类型：Struct
引用类型	类类型	所有其他类型的基类：Object
		Unicode 字符串：String
		用户自定义类型：Class
	接口类型	一维和多维数组
	数组类型	用户自定义类型：Interface
	委托类型	用户自定义类型：Delegate

2.2.3　基本类型转换

将值从一种数据类型更改为另一种类型的过程称为“转换”。类型转换主要有两种形式：收缩转换和扩大转换。收缩转换是将大值域数据类型向小值域的数据类型转换，容易导致数据精度损失和编译器异常；而扩大转换是将小值域数据类型向大值域的数据类型转换，这种转换不会造成数据丢失。

表 2-4 列出了可以进行安全转换的数据类型。

表 2-4　数据类型的安全转换

数据类型	扩大到数据类型
Byte	Byte、Short、Integer、Long、Decimal、Single、Double
Short	Short、Integer、Long、Decimal、Single、Double
Integer	Integer、Long、Decimal、Single、Double
Long	Long、Decimal、Single、Double
Decimal	Decimal、Single、Double
Single	Single、Double
Double	Double
任何枚举类型	其基础 Integer 类型以及它将扩大到的任何类型
Char	Char、String
任意类型	Object、它实现的任何接口

类型转换可以隐式或显式的方式进行。

- 隐式转换：从类型 A 到类型 B 的转换可以在所有的情况下进行，执行转换的规则很简单。隐式转换不需要编程者做任何工作（不需要编写代码）。
- 显式转换：从类型 A 到类型 B 的转换只能在某些情况下进行，转换的规则较复杂，需要额外编写代码。显式转换有三种方式：使用类型转换关键字、使用 Ctype 方法和使用 Convert 类。

1. 使用类型转换关键字进行显式转换

表 2-5 列出了可用的转换类型关键字。

表 2-5　转换类型关键字

类型转换关键字	目的类型	原类型
CBool	Boolean	任意数值类型（包括枚举类型）、String、Object
CByte	Byte	任意数值类型、枚举类型、Boolean、String、Object
CChar	Char	String、Object
CDate	Date	String、Object
CDbl	Double	任意数值类型（包括枚举类型）、Boolean、String、Object
Cdec	Decimal	任意数值类型（包括枚举类型）、Boolean、String、Object
CInt	Integer	任意数值类型（包括枚举类型）、Boolean、String、Object

续表

类型转换关键字	目的类型	原类型
CLng	Long	任意数值类型（包括枚举类型）、Boolean、String、Object
Cobj	Object	任意类型
Cshort	Short	任意数值类型（包括枚举类型）、Boolean、String、Object
Csng	Single	任意数值类型（包括枚举类型）、Boolean、String、Object
CStr	String	任意数值类型、Boolean、Char、Char 数组、Date、Object

CBool 示例：使用 CBool 函数将表达式转换为 Boolean 值。如果表达式的计算结果为非零值，CBool 将返回 True；否则返回 False。

```
Dim A, B, C As Integer
Dim Check As Boolean
A = 5
B = 5
Check = CBool(A = B)                    'Check 值为 True
C = 0
Check = Cbool©                          'Check 值为 False
```

CStr 示例：使用 CStr 函数将一个数值转换为字符串。

```
Dim MyDouble As Double
Dim MyString As String
MyDouble = 437.324
MyString = CStr(MyDouble)               'MyString 值为"437.324"
```

2. 使用 Ctype 方法进行类型转换

Ctype 方法声明方式为：Ctype(expression, typename)

其中，expression 代表任何有效的表达式，typename 代表任何数据类型。

代码示例：

```
Dim MyNumber As Long
Dim MyNewType As Single
MyNumber = 1000
MyNewType = Ctype(MyNumber,Single)      'MyNewType is set to 1000.0
```

3. 使用 Convert 类进行类型转换

Convert 类位于 System 名称空间下，含有许多公共方法，可将一个基本类型转换为另一个基本数据类型。Convert 类的公共方法如表 2-6 所示。

表 2-6 Convert 类的公共方法

公共方法	功能描述
IsDBNull	返回有关指定对象是否为DBNull类型的指示
ToBoolean	将指定的值转换为等效的布尔值
ToByte	将指定的值转换为 8 位无符号整数
ToChar	将指定的值转换为 Unicode 字符
ToDateTime	将指定的值转换为DateTime

续表

公共方法	功能描述
ToDecimal	将指定值转换为Decimal数字
ToDouble	将指定的值转换为双精度浮点数字
ToInt16	将指定的值转换为 16 位有符号整数
ToInt32	将指定的值转换为 32 位有符号整数
ToInt64	将指定的值转换为 64 位有符号整数
ToSByte	将指定的值转换为 8 位有符号整数
ToSingle	将指定的值转换为单精度浮点数字
ToString	将指定值转换为其等效的String表示形式
ToUInt16	将指定的值转换为 16 位无符号整数
ToUInt32	将指定的值转换为 32 位无符号整数
ToUInt64	将指定的值转换为 64 位无符号整数

代码示例：

```
Dim dNumber As Double
dNumber = 23.15
Dim iNumber As Integer
iNumber = System.Convert.ToInt32(dNumber)      ' Returns 23
Dim bNumber As Boolean
bNumber = System.Convert.ToBoolean(dNumber)    ' Returns Ture
Dim strNumber As String
strNumber = System.Convert.ToString(dNumber)   ' Returns "23.15"
```

2.2.4　值类型的装箱和拆箱

通常值类型比引用类型有更高的执行效率，但会过多地占用堆栈空间，因此很多情况下需要将值类型转换为引用类型使用，这种转换叫做装箱。

例如执行如下代码：

```
Dim i As Integer = 123      '声明一个值类型变量 i 并赋值
Dim obj As Object = i       '声明一个引用类型变量 obj 并赋值
```

以上两句代码发生了一次值类型的装箱操作，一次装箱操作需要顺序执行以下三个动作：

（1）在托管堆中创建一个对象实例 obj，给它分配内存。

（2）将 i 的值复制到对象实例 obj 中。

（3）将 obj 的地址压入堆栈中。

如图 2-5 所示。

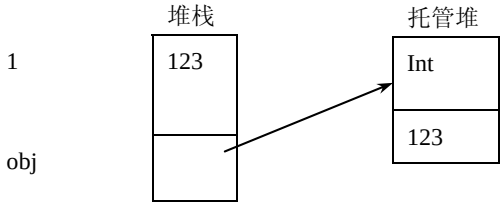

图 2-5　i 装箱后内存布局示意图

i 装箱后，变量 obj 和变量 i 相互独立，即改变 obj 或 i 的值都不会相互影响。顺便指出，装箱操作会从速度和内存两方面损伤应用程序性能，编写程序时应该尽量减少装箱的次数。

拆箱是获取指向对象中包含值类型部分（数据字段）的内存地址，并不涉及内存字节的拷贝，拆箱操作的代价要比装箱操作小的多。拆箱和装箱并不是严格意义上的互反操作，也就是说拆箱操作加上内存数据的拷贝操作才是装箱的互反操作。

2.3 变量与常量

2.3.1 变量的命名规则

变量是构成任何一种程序设计语言的基本要素。在程序运行过程中，变量担当临时容器的角色。它可暂时存放程序处理中产生的某些数据，以供后续程序引用。每个变量只能存储一个值，这个值可以被随时更改。

命名变量时，应遵循下列规则：

（1）变量名的首字符必须是英文字母、汉字或下划线，不能是数字，其余字符可以是英文字母、汉字、下划线或数字，变量名最长不得超过 16383 个字符。变量名若以下划线开头，则变量名中必须另外包括至少一个英文字母、汉字或数字。

（2）不能使用 Visual Basic 的保留标识（如对象、方法、事件关键字等）做变量名。

（3）为增强代码的可读性，命名变量时，可加前缀以表达变量的数据类型。不宜用单一字符命名变量。因为这样看似节省了些许时间，实则难读难记，付出的代价太大。

2.3.2 变量和常量的声明

1. 变量的声明

Visual Basic.NET 要求，变量必须在使用前加以说明。提出这样的要求有两方面的考虑：第一，在编译阶段若能确知变量的数据类型，可有效提高编译效率；第二，可有效减少输入错误。

变量声明语法如下：

```
[ReadOnly] Dim name [As [New] type] [=expression]
```

下面是几个例子：

```
Dim boolVar As Boolean                          '声明一个 Boolean 型变量 boolVar
Dim dtVar As Date                               '声明一个 Date 型变量 dtVar
Readonly Dim dtVar As Date                      '声明一个只读 Date 型变量 dtVar
Dim objVar As Object                            '声明一个 Object 型变量 objVar
Dim objVar                                      '声明一个 Object 型变量 objVar
Dim intX,IntY,IntZ As Integer                   '声明三个 Integer 型变量
Dim IntX As integer,dbY,dbZ As Double           '声明三个变量
Dim IntX As integer = 99                        '声明一个 Integer 型、初值为 99 的变量
```

在声明变量时，应注意以下几点：

（1）对某一变量而言，一旦声明，就不能再声明为其他类型（但可以使用类型转换函数将其转换为其他类型）。

（2）将某一数值存入一个有效位数不够的变量时，数值将被四舍五入。但需要注意的是，

如果舍入位为 5，则进行舍入操作时，Visual Basic.NET 会先检查舍入位的前一位，若该位数字为奇数，则进位；否则不进位。例如，下列语句执行后，intX 的值为 128，而 intY 的值为 130。

```
Dim intX,intY As Integer
intX=128.5
intY=129.5
```

2. 常量的声明

在程序设计中，对于一些会经常用到的常数（如圆周率），一般会定义为常量。与变量不同的是，常量一经定义，就不可更改。常量仅仅在编译过程中有意义，一旦编译完成，常量将被其所代表的值替代。常量的优点是便于阅读和修改程序。

在 Visual Basic.NET 中，常量有两种，分别是用户自定义常量和系统定义常量。

（1）用户自定义常量。程序设计者可用下列语句自定义常量：

```
Const name [As type] = expression
```

下面是几个例子：

```
Const PI=3.14159                        '省略 As 关键字，将使用表达式的数据类型
Const MyName As String ="SYR"           '声明一个 String 型常量，值为"SYR"
```

（2）系统定义常量。为了便于程序员使用，Visual Basic.NET 定义了许多常量，表 2-7 是一些例子。

表 2-7　系统定义常量举例

名称	值	说明
vbCrLf	Chr(13)+ Chr(10)	回车换行符（Carriage return–linefeed combination）
vbCr	Chr(13)	回车符（Carriage return）
vbLf	Chr(10)	换行（Line feed）
vbFormFeed	Chr(12)	对 Microsoft Windows 而言没有意义
vbNullChar	Chr(0)	空字符
vbNullString		长度为 0 的字符串，即空字符串
vbTab	Chr(9)	"Tab"
vbVerticalTab	Chr(11)	对 Microsoft Windows 而言没有意义

2.3.3　变量的作用域和生存期

变量的作用域即变量在程序代码中有效的范围，根据变量的作用域不同，可以把变量分为全局变量和局部变量。全局变量在整个程序中都有效，其作用域为整个程序；局部变量只在声明它的函数和过程中有效，其作用域只限于该函数或过程本身。

代码示例：

```
Module ConstAndVar
      Sub Write()
       Console.WriteLine("myString={0}", myString)
   End Sub
   Sub Main()
```

```
        Dim myString As String = "String defined in Main()"
        Write()
    End Sub
End Module
```

编译上述代码，任务列表中会出现报错信息：“名称'myString'未声明”。报错的原因是在 Write 方法中没有定义 myString 变量，但该函数在执行过程中需要使用 myString 变量；而 Main 方法中虽然定义了 myString 变量但它只能在 Main 方法中使用。实际上 Write 方法和 Main 方法是两个不同的子程序，在它们内部定义的变量，外部是看不见的，也不能被外部使用。若改正上述错误，可以在 Write 方法中声明一个变量 myString，代码如下所示：

```
Module ConstAndVar
    Sub Write()
Dim myString As String ="String defined in Write()"
        Console.WriteLine("myString={0}", myString)
    End Sub
    Sub Main()
        Dim myString As String = "String defined in Main()"
        Write()
Console.WriteLine("myString={0}", myString)
    End Sub
End Module
```

上述代码运行结果如图 2-6 所示。

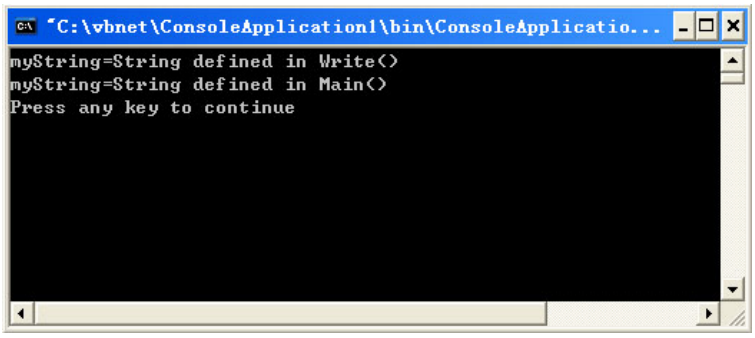

图 2-6　程序执行结果

这段代码按照如下顺序依次执行：

（1）Main()定义和初始化一个字符串变量 myString。

（2）Main()把控制权传给 Write()子程序。

（3）Write()子程序定义和初始化一个字符串变量 myString，它与 Main()中定义的 myString 变量完全不同。

（4）执行 Write()子程序中的 Console.WriteLine("myString={0}", myString)方法，Console.WriteLine 方法是将引号中的内容打印在屏幕上，其中“{0}”代表变量 myString 的值。此处 myString 的值是 Write()子程序中定义的值。

（5）Write()子程序执行完毕，将控制权交给 Main()，执行 Main()中的 Console.WriteLine

方法，该方法中 myString 的值应该是 Main()中定义的值。

由上述代码执行顺序可以得出如下结论：

（1）程序中代码的执行顺序一般是从上到下依次执行的。当执行的代码段是一个子程序或函数，程序会进入该子程序或函数内部继续执行，直到执行完毕，然后返回到该段代码，继续顺序执行。

（2）变量的作用范围与变量的声明位置密切相关，在不同位置声明变量，其作用域也不同。例如在 Sub…End Sub 中声明的变量和在 Module…End Module 中声明的变量的作用域不同。不同的 Sub…End Sub 声明的变量作用域也是不同的。

（3）变量的作用范围包括定义变量的代码块和直接嵌套在其中的代码块。上例中 Sub…End Sub 是嵌套在 Module…End Module 中的，定义在 Module…End Module 中的变量，Module…End Module 中的代码可以使用该变量，Sub…End Sub 中的代码也可以使用该变量，相反则不正确，也就是说，在 Sub…End Sub 中声明的变量是不能在 Module 中使用的，当然也不能被其他 Sub…End Sub 中的代码所使用。

（4）位于同一作用范围的变量不能同名，而位于不同作用范围的变量可以同名。例如，上例在 Write()中定义的 myString 和在 Main()定义的 myString 就属于不同范围内定义的变量，所以可以同名。

除作用域之外，变量还有生存期，生存期指的是变量可供使用的时间周期。在生存期内变量的值可以更改，但变量总是保留某些值。

（1）生存期的开始。当程序执行到某个过程或函数时，其内部声明的局部变量的生存期就开始了。也就是说，内存会为每个被声明的变量开辟一段空间用于存储变量初始化的值，然后变量被初始化为其数据类型的默认值。例如，数字变量被初始化为 0，Date 变量被初始化为公元 1 年的 1 月 1 日零时，Boolean 变量被初始化为 False，引用变量被初始化为 Nothing。

（2）生存期的结束。当过程终止时，不再保留该过程的局部变量值，并回收局部变量所使用的内存。

例如，上例在 Write()过程中 myString 变量的存亡过程是，执行完 Dim 语句后变量 myString 被赋值存储在某一段内存中，执行完 End Sub 语句后内存被收回，同时变量的值被清除。每次调用 Write()过程都会重复这个过程。

2.4　运算符与表达式

运算符是表达进行何种运算操作的记号。它的操作对象一般称为算子，算子可以是常量、变量或表达式。根据所需算子数目的不同，还可将运算符分为单目和双目运算符两类。单目运算符用于算子之前或之后，如++x 或 x++；双目运算符用于两个算子中间，如 x+y。

如果在一个运算式中混合使用了多种运算符，则进行运算时，运算符的操作顺序将遵循一定的规则，即不同的运算符有不同的优先级，运算符的优先级在 Visual Basic.NET 中有严格的定义。

2.4.1　算术运算符

算术运算符专门用于数字运算，运算结果也是数字类型，如表 2-8 所示。

表 2-8 算术运算符

运算符	运算符定义	举例	结果
+	加法符号	12+12	24
-	减法符号	12-2	10
*	乘法符号	12*2	24
/	除法符号	12/2	6
\	整数除法	7\2	3
^	求幂	2^2	4
MOD	取余数	7MOD2	1

补充说明以下几点：

（1）“+”运算符也用于连接两个字符串。使用“+”运算符时，有可能无法确定是做加法还是做字符串连接。而“&”运算符用于生成两个表达式的字符串连接。

（2）“-”运算符也可以表示数值表达式的负值。

（3）“\”运算符返回的结果为整数，任何小数部分都被截断。

（4）“MOD”运算符是将两个数相除并只返回余数。如果除数或被除数有一个数是浮点数，则结果是表示余数的浮点数。

2.4.2 赋值运算符

赋值运算符的主要作用是给变量赋值，常用的运算符如表 2-9 所示。

表 2-9 赋值运算符

运算符	功能	举例	结果
=	用于将值赋给变量或属性	x = 42	42
^=	计算以该变量值为底、以表达式为指数的幂，并将结果赋回给该变量	10^=3	1000
*=	变量值乘以表达式值，并将结果赋给该变量	10*=3	30
/=	变量值除以表达式值，并将结果赋给该变量	12/=3	4
\=	变量值除以表达式值，并将整数结果赋给该变量	10/=3	3
+=	将变量值与表达式值相加，并将结果赋给该变量；也可以连接 String 变量与 String 表达式，并将结果赋给该变量	10+=3 "10"+="3"	13 "103"
-=	从变量值中减去表达式值，并将结果赋给该变量	10-=3	7
<<=	对变量值执行数学左移位，并将结果赋给该变量	10<<=3	80
>>=	对变量值执行数学右移位，并将结果赋给该变量	10>>=2	2
&=	连接 String 变量与 String 表达式，并将结果赋给该变量	Hello&=Word	HelloWord

2.4.3 比较运算符

比较运算符用于对表达式进行比较，常用的比较运算符如表 2-10 所示。

表 2-10　比较运算符

运算符	运算符名称
<	小于
<=	小于或等于
>	大于
>=	大于或等于
=	等于
<>	不等于

补充以下几点：

（1）数值的比较：当 Single 类型的表达式与 Double 类型的表达式比较时，Single 表达式被转换为 Double，此行为与 Visual Basic 6 中的行为相反；与此相似，当 Decimal 类型的表达式与 Single 或 Double 类型的表达式比较时，Decimal 表达式将被转换为 Single 或 Double。

（2）字符串的比较：比较字符串时，根据字符串表达式的字母排序顺序对其进行计算，字符串排序顺序是根据 Option Compare 设置计算的。

（3）比较运算符进行无类型编程（不必先声明变量即可引用变量，必须将 Option Strict 设为 Off，才可以进行比较），如表 2-11 所示。

表 2-11　无类型编程的操作数比较

操作数类型	比较结果
都为字符串	根据字符串排序特征进行排序比较
都为数值	对象将被转换为 Double，并执行数值比较
一个是数值而另一个是 String	String 被转换为 Double 并执行数值比较。如果 String 不能转换为 Double，将引发 InvalidCastException
有一个或两个都是 String 以外的引用类型	将引发 InvalidCastException

2.4.4　串联运算符

串联运算符将多个字符串联接为一个字符串。有两种串联运算符：+和&，如表 2-12 所示。

表 2-12　串联运算符

运算符	功能	举例	结果
&	生成两个表达式的字符串连接	Hello" & " World"	"Hello World"
+	主要用作两个数相加，也可用于连接两个字符串	Hello" + " World"	"Hello World"

2.4.5　逻辑运算符

逻辑运算符比较 Boolean 表达式，并返回 Boolean 结果，如表 2-13 所示。

表 2-13 逻辑/按位运算符

运算符	含义
And	如果两个位均为 True，则结果位为 True；否则结果位为 False
Or	如果任何一个位为 True，则结果位为 True；否则结果位为 False
Xor	两个位不同时为 True 或不同时为 False，则结果位为 True；如果同时为 False 或 True，则结果为 False
Not	求反运算
AndAlso	和 And 功能一样，只是比 And 比较的速度更快
OrElse	和 Or 功能一样，只是比 Or 比较的速度更快

2.4.6 移位运算符

对操作数执行移位操作，如表 2-14 所示。

表 2-14 移位运算符

运算符	功能	举例	结果
<<	使操作数中的数位向左移动指定的位数	192<<4	3072
>>	使操作数中的数位向右移动指定的位数	192>>4	12

2.4.7 其他运算符

（1）GetType 运算符：用于返回指定类型的 Type 对象。例如：GetType(Integer)。

（2）Is 运算符：Is 运算符确定两个对象引用是否引用同一个对象。但是，它不执行值比较，不会抛出异常。如果 object1 和 object2 都引用同一个对象，则比较结果为 True；否则为 False。

例如：

```
Dim myObject As New Object
Dim otherObject As New Object
Dim yourObject, thisObject, thatObject As Object
Dim myCheck As Boolean
yourObject = myObject
thisObject = myObject
thatObject = otherObject
myCheck = yourObject Is thisObject   '此时 myCheck 为 True
myCheck = thatObject Is thisObject   '此时 myCheck 为 False
```

（3）Like 运算符：比较两个字符串。Like 运算符的行为取决于 Option Compare 语句。内置的模式匹配为字符串比较提供了一种多功能工具。模式匹配功能允许使用通配符、字符列表或字符范围的任何组合来匹配字符串。例如：“?”代表任何单个字符；“*”代表零或更多字符；“#”代表任何单个数字（0 到 9）；“[]”代表[]中的任何单个字符；“[!]”代表不在[]中的任何单个字符。

2.4.8 运算符的优先级

运算符的优先级决定同一算式中所有运算符的运算次序，当表达式包含不止一种运算符

时，则按照下列规则对其进行计算。算术运算符和串联运算符的优先级均高于比较运算符、逻辑运算符和位运算符；比较运算符高于逻辑运算符和位运算符；逻辑运算符和位运算符优先级最低。具有相同优先顺序的运算符将按照它们在表达式中出现的顺序从左至右进行计算。

表 2-15 列出了 Visual Basic 规定的运算符。位于同一表格行中的运算符优先级相同，位于较上方表格行中的运算符的优先级较高。

表 2-15　运算符优先级

类别	运算符
求幂	^
求负	–
乘法和除法	*、/
整数除法	\
取模	Mod
加法和减法	+、-
字符串串联	&
数学移位	<<、>>
比较运算符	=、<>、<、<=、>、>=、Like、Is、TypeOf...Is
逻辑非	Not
逻辑与	And、AndAlso
逻辑或	Or、OrElse
逻辑与或	Xor

2.5　语句

语句是程序完成一次完整操作的基本单位。在 Visual Basic.NET 中可分为声明语句、赋值语句和 Option 语句三种。

2.5.1　声明语句

声明语句用于命名和定义过程、变量、数组和常数，同时指明了它们的类型，定义了范围，并申请了需要的内存空间。

代码示例：下面的示例包含三个声明。

```
Sub ApplyFormat()
  Const limit As Integer = 33
  Dim myString As String
End Sub
```

代码分析：

（1）Sub…End Sub 语句：声明了 ApplyFormat 是一个过程，每当运行该过程，都执行包含在 Sub…End Sub 中的所有语句。

（2）Const 语句：声明 limit 是一个常量，类型为 Integer，值为 33，其实该句包含两类语

句，“=”前是声明语句，“=”后是赋值语句。

（3）Dim 语句：声明 myString 是一个变量，类型为 String。

2.5.2 赋值语句

1. 简单赋值语句

简单赋值语句由变量+“=”+表达式（值）构成，变量用于存储表达式的结果。“=”两端的类型要匹配，或者说声明语句和赋值语句的类型要匹配。在运行时“=”右侧的表达式先于变量表达式计算。如果被赋值的变量是一个引用类型的数组元素，将执行运行时检查，以确保表达式同数组元素类型兼容。

例如：

```
#01: Dim i As Integer =10
#02: Dim oa(2) As String
#03: oa(0) = "Hello"
#04: oa(1) = i
```

代码分析：

#01：声明一个 Intege 类型变量 i，并赋值为 10。

#02：声明一个 String 类型的数组（在第 3 章有详细介绍）。

#03：给数组 oa 的第一个元素赋值，类型为 String，值为 Hello。

#04：给数组 oa 的第二个元素赋值，类型为 Integer，值为 10。编译代码会发现该句会出错，因为声明语句中数组 oa 是 String 类型的而赋值语句中 i 是 Integer 类型的，类型不匹配。

需要指出的是，赋值语句的语义取决于被赋值的变量类型：如果被赋值的变量为值类型，则赋值语句将表达式的值复制到变量中；如果被赋值的变量为引用类型，则赋值语句将引用而不是值本身复制到变量中；如果变量类型为 Object，则赋值语义取决于表达式是值类型还是引用类型。

2. 复合赋值语句

Visual Basic.NET 支持复合赋值语句。与完全展开的表达式不同，复合赋值语句左侧的变量仅计算一次。这意味着在运行时，变量表达式先于赋值语句右侧的表达式计算。

例如：

```
Dim i As Integer
i += 10
```

上例代码相当于 i=i+10，“+”运算先于“=”运算，且仅计算一次。可以使用复合赋值语句的运算符包括：“^”、“*”、“/”、“\”、“+”、“-”、“&”。

3. Mid 赋值语句

Mid 赋值语句将一个字符串赋给另一个字符串。

Mid 语句为：

```
Mid(ByRef Target As String,ByVal Start As Integer,
    Optional ByVal Length As Integer) = StringExpression
```

参数说明：

- Target：要修改的 String 变量的名称。
- Start：Target 中文本替换开始的字符位置。

- Length：要替换的字符数。若省略该参数，则使用所有字符串。
- StringExpression：替换 Target 部分的 String 表达式。

顺便指出，在 Mid 赋值语句中，所替换的字符数始终少于或等于 Target 中的字符数，每个空格算一个字符。

示例：

```
MyString = "The dog jumps"              '定义一个 MyString 变量
Mid(MyString, 5, 3) = "fox"             ' MyString = "The fox jumps"
Mid(MyString, 5) = "cow"                ' MyString = "The cow jumps"
Mid(MyString, 5) = "cow jumped over"    ' MyString = "The cow jumpe"
Mid(MyString, 5, 3) = "duck"            ' MyString = "The duc jumps"
```

4. 委托赋值语句

如果赋值语句包含关键字 AddressOf，则赋值语句右侧的表达式必须为调用目标，并且变量的类型必须是签名同调用目标兼容的委托（相关概念详见 3.8 节）。例如给一个按钮的单击事件赋值可以这样声明：

```
Dim myButton As Forms.Button AddHandler myButton.Click, AddressOf myButton_Click
```

上面的委托赋值语句的含义是：将 myButton 这个按钮的单击事件 myButton.Click，委托给了事件处理程序 myButton_Click 来处理。

2.5.3　Option Strict、Option Explicit、Option Compare 语句

Visual Basic.NET 中的 Option Explicit 和 Option Strict 是针对编译器的语句。一段程序编译成功了，并不意味着这段程序没有错误，也可能会在运行时出现错误；程序运行一次没有错误也不表明下次运行时一定不会发生错误。因此，如果潜在的运行时错误在编译期被及时发现，程序会更加稳定。

Option Explicit 语句决定编译器是否要求所有的变量被显式地声明，语法如下：

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

如果选择 On，则编译器要求所有的变量被显式地声明；如果选择 Off，则编译器允许变量被隐式声明。变量的隐式声明可能会产生错误，如在代码的编写过程中，不小心敲错了变量名，由于编译器允许变量的隐式声明，所以在程序编译时不会发生错误，但在运行时可能会发生不可预期的错误。因此在代码中最好不要用 Option Explicit Off。

Option Strict 语句决定编译器类型的隐式转换方面的要求，语法如下：

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

如果选择 On，则编译器允许类型的扩大转换；选择 Off，编译器不仅允许扩大转换，还允许收缩转换。收缩转换可能使程序产生不可预知的错误。

Option 语句必须在程序的开始位置。例如：

```
Option Strict On
Module Module1
    Dim x As Integer
    Dim y As Double
    Sub Main()
        x = y
    End Sub
```

```
End Module
```

编译结果生成一个错误为：Option Strict On 不允许从 Double 到 Integer 的隐式转换。若将 Option Strict On 改为 Option Strict Off 则编译通过。

Option 语句也可以在 Visual Studio 2005 中设置，方法为：在解决方案资源管理器中右击“项目”，选择“属性”→“编译”，如图 2-7 所示。

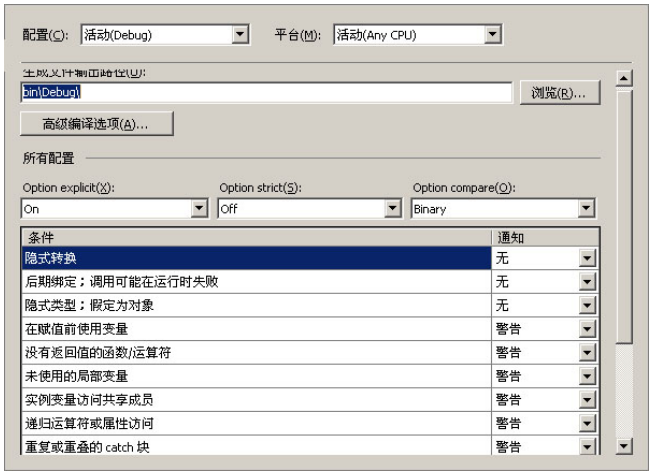

图 2-7　Option Strict 设置

Option Compare 语句的作用是声明字符串比较时默认使用的方法。图 2-7 中 Option compare 选项中选择 Binary，则说明默认字符串比较方法采用二进制比较；如果选择 Text，则说明默认字符串比较方法采用区分大小写的文本顺序排序。

2.6　基本控制结构

2.6.1　顺序结构

Visual Basic.NET 代码的执行顺序是从第一个可执行语句开始，依次执行，直到程序结束语句为止。顺序结构程序中的任何一个可执行语句，在程序运行过程中，都必须运行一次，而且也只能运行一次。这样的程序结构最简单、最直观、最易于理解。顺序执行是程序执行的基本规则，控制语句（如选择语句、循环语句、条件语句和暂停语句）能改变程序执行顺序。

2.6.2　选择结构

1. If...Then...Else 语句

该语句是根据表达式的值，有条件地执行一组语句。

声明方式：

```
If 布尔条件表达式
Then [语句 1]
[Else 语句 2]
End If
```

该语句由 If、Then、End If 组成，缺一不可，Else 是可选的。该语句执行流程如图 2-8 所示。

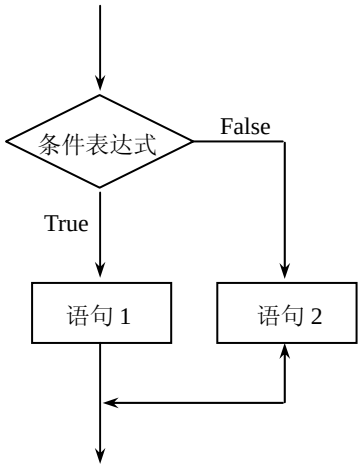

图 2-8　If 语句执行流程

图 2-8 所示程序执行顺序为：若条件表达式的值为 True 时，程序执行语句 1；否则执行语句 2。其中语句 1 和语句 2 可以是简单语句，也可以是复合语句。

示例：

```
Dim x As Integer=4
Dim y As Boolean
If x>5 Then
y=true
Else
y=false
End if
```

代码执行结果是 y=false，因为 x=4，当判断条件 x>4 时为 false，所以代码执行 else 代码块的内容，即 y=false。

2. Select Case 语句

该语句是根据表达式的值，执行若干组语句中的某一组。该语句可以一次将测试变量与多个值进行比较，而不是仅测试一个条件。

声明方式：

```
Select  Case  控制表达式
Case 常量表达式
      语句1
Case 常量表达式
      语句2
Case Else
     其他语句
End Select
```

说明：

（1）控制表达式是任意的数值或字符串表达式，若控制表达式的值与 Case 语句中常量表

达式相匹配，则执行该 Case 后的语句；若任何一个 Case 语句都不匹配，则执行 Case Else 代码块。

（2）比较时按照 Select…Case 块的出现顺序进行，因此 Case Else 语句必须放在 Case 语句的最后，因为在任何情况下，Case Else 语句都会执行。

（3）可以有任意数量的 Case 语句，每个 Case 语句可以包括多个值。

（4）Case Else 语句可以省略。

（5）To 关键字：常量表达式匹配值范围的边界。

例如：

```
Case x To y  '常量表达式的值在 x 和 y 之间且包括 x 和 y,x<=y
```

（6）Is 关键字：指定对常量表达式的匹配值的限制

例如：

```
Case Is <a  '常量表达式的值小于 a
```

Select Case 语句执行流程如图 2-9 所示。

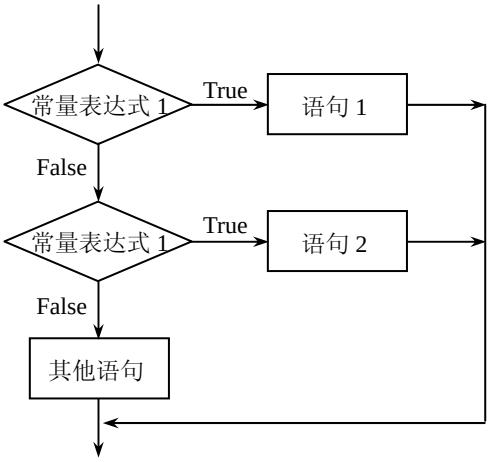

图 2-9 Select Case 语句执行流程

代码示例：建立控制台应用程序 CaseExam，程序的功能是练习使用 Case 语句。

```
#01:    Dim Number As Integer = 10
#02:    Select Case Number
#03:            Case 1 To 5
#04:            Console.WriteLine("Between 1 and 5")
#05:            Case 6, 7, 8
#06:            Console.WriteLine("Between 6 and 8")
#07:            Case Is < 20
#08:            Console.WriteLine("Greater than 8 and Less than 20")
#09:            Case Else
#10:            Console.WriteLine("Not between 1 and 20")
#11:    End Select
```

程序运行结果如图 2-10 所示。

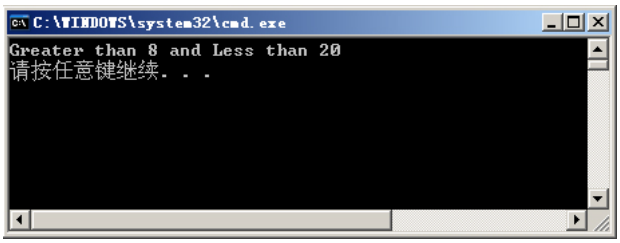

图 2-10　程序运行结果

代码分析：

#01：将变量赋值，Number=10。然后代码会从上到下顺序执行，Number 会依次和 Case 表达式进行比较。

#03：变量 Number 的值是否是 1～5 中间的一个数，若条件符合，则执行#04；否则执行#05。

#05：变量 Numberd 的值是否是 6～8 中间的一个数，若条件符合，则执行#06；否则执行#07。

#07：变量 Numberd 的值是否小于 20，该条件符合，执行#08。

#09：当所有的 Case 子句都不能满足要求时，执行该子句。

2.6.3　循环结构

1. Do...Loop 语句

使用 Do...Loop 语句，可以使代码块执行不确定的次数。循环条件通常从两个值的比较中得到，也可以是任何值为 Boolean（True 或 False）的表达式。语法如下：

```
Do While 布尔条件表达式
…
[Exit Do]
…
Loop
```

或

```
Do
…
[ Exit Do ]
…
Loop  While 布尔条件表达式
```

（1）Do While...Loop：在进入循环前检查条件，在条件保持为 True 时，循环会继续下去，程序执行流程如图 2-11 所示。

代码示例：创建控制台程序 DoWhileLoop，在进入循环前先测试条件，只有 Number 的值大于 6，循环中的语句才会运行。代码如下所示。

```
Module Module1
    Sub Main()
        Dim Counter As Integer = 0
        Dim Number As Integer = 10
        Do While Number > 6
            Number = Number - 1
```

```
            Counter = Counter + 1
        Loop
        Console.WriteLine("循环运行了 " & Counter & " 次")        '运行了 4 次
    End Sub
End Module
```

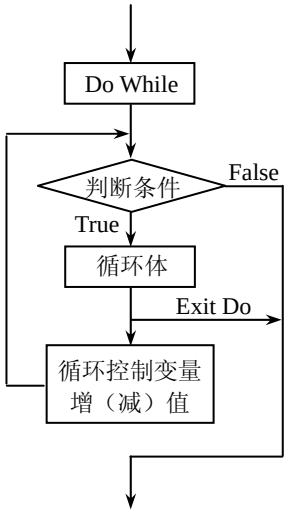

图 2-11　Do While…Loop 语句执行流程

（2）Do Loop...While：在循环至少运行一次后检查条件，在条件保持为 True 时循环会继续下去。

代码示例：创建控制台程序 DoLoopWhile，循环中的语句在测试条件前运行一次，该条件在第一次测试时为 False。代码如下所示。

```
Module Module1
    Sub Main()
        Dim Counter As Integer = 0
        Dim Number As Integer = 5
        Do
            Number = Number - 1
            Counter = Counter + 1
        Loop While Number > 6
        Console.WriteLine("循环次数为 " & Counter & " 次")        '循环了 1 次
    End Sub
End Module
```

（3）Exit Do：该语句将控制立即传送到 Loop 语句后面的语句。可以在 Do 循环中的任何位置放置任何数量的 Exit Do 语句。Exit Do 经常用在计算某个条件（如 If...Then...Else）之后。

代码示例：创建控制台程序 ExitDoExam。代码如下所示。

```
Module Module1
    Sub Main()
        Dim Counter As Integer = 0
        Dim Number As Integer = 8
        Do Until Number = 10
            If Number <= 0 Then Exit Do
```

```
            Number = Number - 1
            Counter = Counter + 1
        Loop
        Console.WriteLine("循环次数为 " & Counter & " 次。")        '循环了 8 次
    End Sub
End Module
```

顺便指出，若要停止无限循环，可以按 Esc 或 Ctrl+Break 组合键退出。

2. For ...Next 语句

Do 循环适合于事先不知道需要执行多少次循环的代码，For...Next 循环适合于执行特定循环次数的代码。

语法如下所示。

```
For counter [ As datatype ] = start To end [ step ]
……
Next [ counter ]
```

几点说明：

（1）变量 counter 必须是支持“>=”、“<＝”、“+”运算符的数据类型，通常为 Integer。Counter 变量在每次重复循环的过程中递增或递减。

（2）迭代值 start、end 和 step 的数据类型通常为 Integer，或者是 Integer 的计算表达式。

（3）可选项 step 可以为正也可以为负。如果省略的话，则默认为 1。

（4）执行语句块之前，先将 counter 与 end 进行比较。如果 counter 已经超过了结束值，则 For 循环终止，控制传递给 Next 语句后面的语句；否则执行该语句块。

（5）每次 Visual Basic 遇到 Next 语句时，都按 step 递增计数器，然后返回到 For 语句；再次将计数器与 end 进行比较，并根据结果执行块或者终止循环……直到计数器超过 end 或执行了 Exit For 语句。

（6）可以使用 Exit For 语句在计数器超过其结束值之前退出 For...Next 循环。

For…Next 语句执行流程如图 2-12 所示。

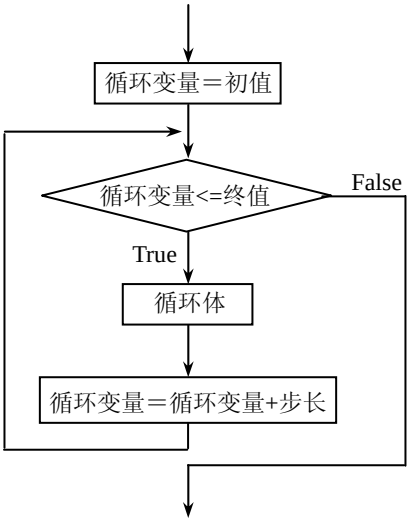

图 2-12　For…Next 语句执行流程

代码示例：创建控制台程序 ForNextExam。

```
Module Module1
   Sub Main()
      Dim J, Total As Integer
      For J = 2 To 10 Step 2
         Total = Total + J
      Next J
      Console.WriteLine("The total is " & Total)
      'total=0+2+4+6+8+10=30
   End Sub
End Module
```

3. For Each...In…Next 语句

该循环类似于 For...Next 循环，但它为集合中的每个元素执行语句块，而不是执行指定次数。

语法如下所示。

```
For Each elementvariable [ As datatype ] In collection
……
Next [ elementvariable ]
```

几点说明：

（1）collection 的元素可以是任何数据类型。

（2）elementvariable 的数据类型必须是集合的每个元素都可以转换得到的数据类型。

（3）如果未在循环外声明 elementvariable，可以在 For Each 语句中使用 As 子句声明该变量。

（4）对于循环的每次迭代，变量 elementvariable 设置为集合中的元素之一，然后执行语句块。当集合中的所有元素都分配给 elementvariable 后，For Each 循环终止，控制传递给 Next 语句后面的语句。

代码示例：创建控制台程序 ForEachNextExam。

```
Module Module1
   Sub Main()
      Dim A() As Integer = {3, 4, 6}
      Dim Elt As Integer
      Dim Sum As Integer = 0
      For Each Elt In A
         Sum += Elt
      Next Elt
      Console.WriteLine(Sum)       'Sum=3+4+6=13
   End Sub
End Module
```

2.6.4 嵌套控制结构

控制语句可以自身嵌套使用，也可以放在其他控制语句中嵌套使用。

例如，If 语句可以这样使用：

```
If …Then
If …Then
End If
Else
If…Then
End If
End If
```

Select 语句可以这样使用：

```
Select x
   Case a
      Select x
         Case c
……
      End Select
   Case b
…
End Select
```

此外，控制语句还可以相互嵌套，例如：For...Next 循环中嵌套 If...Then...Else 语句。Visual Basic 中的控制语句可以根据需要嵌套任意多级。为了使嵌套决策结构和循环可读性更好，通常的做法是缩进每个控制体。

示例：建立控制台应用程序 Nesting，实现的功能是将矩阵每一行的正元素加在一起，代码如下：

```
Public Sub SumRows(ByVal A(,) As Double, ByRef R() As Double)
   Dim I, J As Integer
   For I = 0 To UBound(A, 1)
      R(I) = 0
      For J = 0 To UBound(A, 2)
         If A(I, J) > 0 Then
            R(I) = R(I) + A(I, J)
         End If
      Next J
      Console.Write("矩阵 A 第" & I)
      Console.WriteLine("行的所有正元素之和为：" & R(I))
    Next I
   End Sub
Sub Main()
   Dim R(4) As Double
   Dim A(2, 2) As Double
   A(0, 0) = 0
   A(0, 1) = 1
   A(0, 2) = 2
   A(1, 0) = 3
   A(1, 1) = 4
   A(1, 2) = 5
   A(2, 0) = 6
```

```
    A(2, 1) = 7
    A(2, 2) = 8
    SumRows(A, R)
End Sub
```

运行结果如图 2-13 所示。

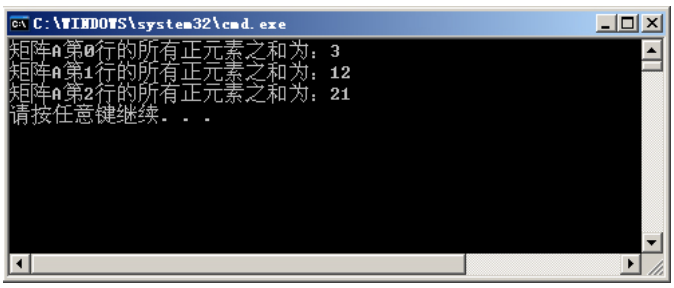

图 2-13　Nesting 程序运行结果

2.6.5　其他辅助控制语句

（1）GoTo 语句：转跳到过程内的指定行，它只能转跳到它所在过程中的行。

语句声明：

```
GoTo lineNumber
```

代码示例：

```
Dim myBoolean As Boolean=false
Dim MyString As String
   If myBoolean=false Then GoTo Line1 Else GoTo Line2
Line1:
   MyString = " Go To Line1"
 Line2:
   MyString = "Go To Line2"
```

GoTo 语句使代码的阅读和维护变得困难，在编码过程中尽量少使用。

（2）Exit 语句：该语句可以直接从任何决策结构、循环或过程中退出，并将执行转移到最后一个控制语句后面的语句。Exit 语句的语法指定要转移出哪种类型的控制语句，可能有下列版本的 Exit 语句：

```
Exit Select
Exit Try
Exit Do
Exit While
Exit For
```

还可以直接从 Function、Sub 或 Property 过程退出，语法类似于 Exit For 和 Exit Do，语法如下所示。

```
Exit Sub
Exit Function
Exit Property
```

Exit Sub、Exit Function 和 Exit Property 可根据需要出现任意多次，可出现在过程体内的

任何地方，甚至可以出现在控制语句（如 If...Then...Else）中。

（3）Stop 语句：中止代码的执行。可以将 Stop 语句放在过程的任何地方，以中止执行。Stop 语句类似于代码中设置断点的功能，它不关闭任何文件或清除任何变量。

代码示例：

```
Dim I As Integer
For I = 1 To 6   ' Start For...Next loop
  Console.WriteLine (I)
  Stop
Next I
```

代码遇到 Stop 语句，该过程并没有退出循环，变量没有被清除，还可以继续运行循环。Stop 语句有助于程序员查看循环体中变量的变化。

（4）End 语句：结束代码执行。End 语句可以放在过程的任何位置，以结束代码执行，并且清除变量。它提供一种强迫程序停止的方法。

代码示例：

```
Dim I As Integer
For I = 1 To 10   ' Start For...Next loop
  Console.WriteLine (I)
  End
Next I
```

该循环仅仅运行了一次，因为 End 关键字使循环停止，清除了变量。

（5）With…End With：执行重复引用单个对象或结构的一系列语句。如果有一系列都对同一对象进行操作的语句，则可使用 With...End With 语句一次为所有语句指定该对象。这样可使过程运行得更快，并避免重复键入。

示例：

```
With Label1
    .ForeColor = Color.Red
    .Font = New Font(.Font, FontStyle.Bold Or FontStyle.Italic)
End With
```

2.7　编码规则

1. 基本规则

（1）代码执行遵循顺序结构，控制语句改变代码执行顺序。

（2）行结尾没有特殊字符，按下 Enter 键即开始新的一行。

（3）空行和缩进被忽略。

（4）大小写不敏感。

（5）在语句之间插入“:”可以让多条语句处于同一行。

2. 注释

注释是一段被编译器忽略的代码，可以起到对源代码解释的作用。Visual Basic 有两种类型的注释：

（1）以关键字 REM 开始的一行是注释。

例如：REM Console.WriteLine("这是一行注释。")

（2）单引号后的文本是注释。

例如： 'Console.WriteLine("这是一行注释。")

以上两种方式都是对代码的单行注释，若要对代码块注释，可以使用 Ctrl-K Ctrl-C（Ctrl-E Ctrl-C），取消块注释可以使用 Ctrl-K Ctrl-U（Ctrl-E Ctrl-U）。

3. 代码的续行

当代码较长时，可打断为几行，此时可以使用续行符。续行符是指在行的末尾同时键入“空格”、“_”、“Enter”，这样编译器会将这些行当作一行处理。但是，续行符不能出现在双引号中。

4. Visual Basic 命名约定

（1）最大长度为 255 个字符。

（2）名称的首字符必须为字母或汉字，接下来的字符可以是字母、0～9 的数字或下划线。

（3）不能使用 Visual Basic.NET 的关键字。

（4）名称中各单词首字母均为大写，如 FindLastRecord 和 RedrawMyForm。

（5）函数名和方法名以动词开始，如 InitNameArray 和 CloseDialog。

（6）类名和属性名以名词开始，如 EmployeeName 和 CarAccessory。

（7）接口名称以前缀 I 开始，后面接一个名词或名词词组，如：IComponent；或者接一个描述接口行为的形容词，如：IPersistable。不要使用下划线，不要过多使用缩写，因为缩写会引起混淆。

（8）事件处理程序的名称以一个描述事件类型的名词开始，后面接后缀 EventHandler，如：MouseEventHandler。

（9）事件参数类的名称里要加 EventArgs 后缀。

（10）如果某事件含有“之前”或“之后”的概念，请以现在时或过去时形式使用前缀，如：ControlAdd 或 ControlAdded。

（11）对于长项或常用项，可使用缩写使名称长度适中，例如，可以使用 HTML 代替 HyperText Markup Language。通常，多于 32 个字符的变量名在低分辨率的监视器上难以阅读。同时，确保缩写在整个应用程序中保持一致。

2.8 程序举例

2.8.1 开发 Visual Basic.NET 应用程序的一般步骤

1. 了解 Visual Basic.NET 编码约定

很多人（特别是非专业人士）不太重视编码约定。事实上，编码的标准化会使代码更加容易阅读和维护。

2. 定制自己的 IDE

根据自己的喜好，定制自己的 IDE，如在代码编辑中显示行号、自动换行、自定义字体的大小和颜色等。

3. 合理组织程序文件

如果应用程序较大，生成的文件数量较多时，文件组织不合理会使应用程序的可维护性很差。最简单的办法是使用“解决方案资源管理器”将文件组织成“树状目录”，就是将文件分门别类组织到不同的文件夹中，如：可以建立 Images 文件夹，将图片统一到 Images 中。

4. 创建应用程序界面（控制台应用程序除外）并设置控件属性

用户界面由各种控件组成，所有的控件都放在窗体上，因此程序中的所有信息都要通过窗体显示出来。控件可以用从工具箱中拖拽或编写代码两种方式生成，控件的属性可以在属性窗口中设置或编写代码设置。

5. 编写事件处理程序

可以用下面的方法之一进入事件过程编辑状态：

（1）双击已建立好的控件。

（2）执行“视图”菜单中的“代码”命令。

（3）按 F7 键。

（4）单击“解决方案资源管理器”中的“查看代码”窗口。

6. 保存应用程序

将编写的应用程序保存到适当的位置。保存方法可以使用菜单命令“文件”→“全部保存”，或者使用快捷菜单中的“保存”按钮。

7. 测试和调试应用程序

使用 Visual Studio 开发环境提供的调试工具，检查并排除程序中的错误，保证所开发的程序能实现预定的功能，使之正确工作。

8. 编译和执行应用程序

使用 Visual Studio 2005 集成开发环境或在命令行方式下编译应用程序，生成可执行程序。

9. 部署应用程序

将已完成的应用程序安装到其他计算机上。

2.8.2　控制台程序举例

下面以在屏幕上输出“Hello World!”字符串为例，说明控制台应用程序的编写步骤。

（1）启动 Visual Studio 2005。

（2）依次选择菜单“文件”→“新建”→“项目”后，打开“新建项目”对话框。

（3）将“项目类型”设置为“Visual Basic 项目”。

（4）将“模板”设置为“控制台应用程序”。

（5）在“名称”文本框中输入“Hello World”。

（6）在“位置”的文本框中输入“E:\教材\vb2008.11.4\程序\CH02”，然后单击“确定”按钮，则 Visual Studio 2005 会按照上面设定的参数创建一个控制台应用程序项目，如图 2-14 所示。

（7）在“解决方案资源管理器”窗口中，双击 Module1.vb 文件，进入 Module1.vb 文件的编辑界面，在系统创建的 Main()函数中加入一行代码：

```
Console.WriteLine("Hello World!")
```

（8）Visual Studio 2005 已经自动产生了 Main()函数，下面是所有的代码。

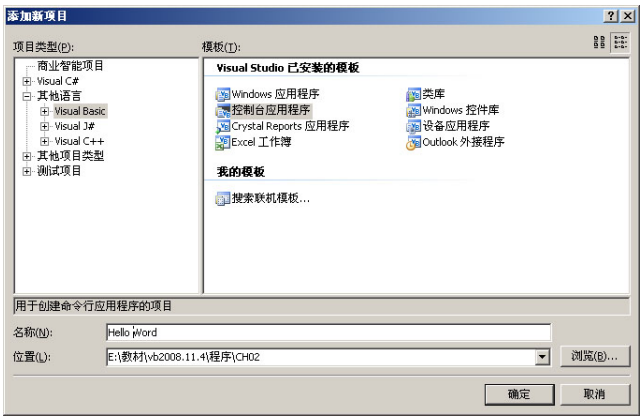

图 2-14　新建控制台应用程序界面

```
#01: Module Module1
#02:    Sub Main()
#03:     '在计算机屏幕上显示信息
#04:     System.Console.WriteLine("Hello World!")
#05:    End Sub
#06: End Module
```

代码分析：

#01、#06：是 Module 语句，必须成对出现，在 Module 语句里面可以编写代码。一个项目中可以有多个模块。

#02：Main 过程，每个 Visual Basic 程序均必须包含一个称为 Main 的过程，该过程为应用程序的起始点，并为应用程序提供总体控制。加载模块时，将调用该过程。

#03：注释语句。“'”表示注释符号，“'”后面的语句将被编译器忽略。

#04：输出语句，使用 Console 类中的 WriteLine 方法。Console 类包含从控制台读取每个字符（Console.Read()）或整个行的方法（Console.ReadLine()）。此类还包含若干写方法，这些方法可自动将值类型的各个实例、字符数组或对象组转换为格式化字符串或无格式字符串，然后将该字符串写入控制台，该字符串后面还可以带行终止字符串。

此外还有一种占位符的写法。例如：Console.WriteLine("{0}:{1}"，str1,str2)，这样每对花括号都表示一个占位符，花括号里面的整数从 0 开始，占位符和后面的变量相对应。若 str1="a"，str2="b"，则输出的结果是 a:b。

#02、#05：是一个过程声明语句，Sub 和 End Sub 也必须成对出现。

（9）选择“文件”→“保存 Module1.vb”，保存所做的修改。

（10）编译和执行。

有两种编译方式：使用命令行和 Visual Studio 2005 集成开发环境。

①使用命令行方式编译和运行程序。单击“开始”→“所有程序”→Visual Studio 2005 .NET →Visaul Studio .NET Tools→“Visual Studio 2005 命令提示”进入命令行窗口，然后输入 vbc.exe /?并回车，出现该命令的帮助信息，如图 2-15 所示。

将控制台应用程序 Hello Word 编译成 exe 文件的命令如下所示。

```
vbc.exe /out:c:/HelloWord.exe /t:exe /r:MSCorLib.dll E:\教材\vb2008.11.4\程
        序\CH02\HelloWord\Module1.vb
```

图 2-15　vbc.exe 帮助信息

说明：

- /out：告诉编译器产生一个可执行文件。上例中是在 c 盘根目录下产生一个 HelloWord.exe 文件。
- /t：编译文件生成的目标类型，上例编译的目标是生成一个.exe 文件。
- /r：告诉编译器到 MSCorLib.dll 程序集中查找外部类型，即调用 Console 类需要到外部类型中寻找。
- E:\教材\vb2008.11.4\程序\CH02\HelloWord\Module1.vb:是要进行编译的文件位置和文件名。

若要运行程序，在命令行窗口中输入正确的路径后，输入 HelloWord.exe 即可。

②从 IDE 编译并运行程序。编译程序可以单击“生成”→“生成解决方案”。运行程序依次单击“调试”→“开始执行（不调试）”，就可以得到如图 2-16 所示的运行界面。

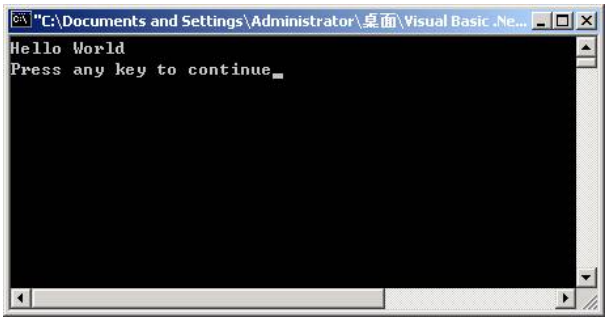

图 2-16　Hello World 控制台程序的运行界面

2.8.3　Windows 应用程序举例

下面以在屏幕上输出“Hello World!”字符串为例，说明 Windows 应用程序的编写步骤。

（1）启动 Visual Studio 2005。

（2）选择菜单命令“文件”→“新建”→“项目”，打开“新建项目”对话框。

（3）将“项目类型”设置为“Visual Basic 项目”。

（4）将“模板”设置为“Windows 应用程序”。

（5）在“名称”文本框中输入“Hello World”。

（6）在“位置”的文本框中输入“E:\教材\vb2008.11.4\程序\CH02”，然后单击“确定”按钮，则 Visual Studio 2005 会按照上面设定的参数创建一个 Windows 应用程序项目，如图 2-17 所示。

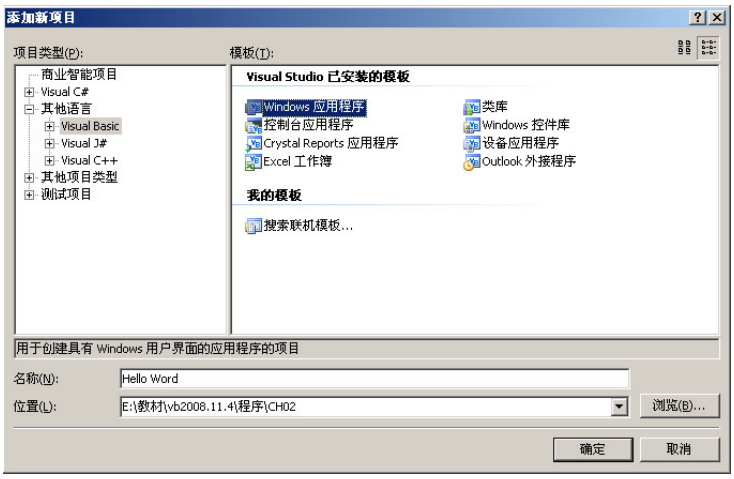

图 2-17　新建 Windows 应用程序界面

（7）选择“工具箱”中的“Windows 窗体组件”，并从中拖入一个 Button 组件到 Form1.vb 窗口，此组件名称为 button1，双击拖入的 button1 组件，Visual Studio 2005 会切换窗口到 Form1 的设计窗口，并自动产生 button1 的 Click 事件对应的代码。

（8）在 button1 的 Click 事件的代码区添加下列代码，下列代码的功能是弹出“Hello World!”提示框。

```
MessageBox.Show("Hello World!")
```

（9）选择“文件”→“保存 Form1.vb”菜单或者按快捷键 Ctrl＋S，保存所做的修改。

（10）选择“生成”→“生成 Hello World 程序”菜单，则 Visual Studio 2005 会自动编译、连接源程序，并生成 Hello World 程序的执行文件。

（11）选择“调试”→“启动”菜单或者按快捷键 F5，则开始运行 Hello World 程序，单击程序中的 Button1 按钮，则弹出“Hello World!”提示框，如图 2-18 所示。

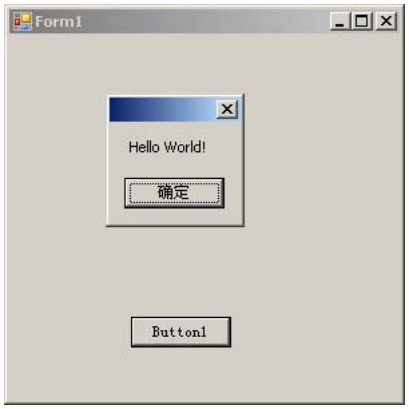

图 2-18　Hello World 程序运行界面

2.8.4　Visual Basic.NET 的应用程序主要的文件类型

Visual Basic.NET 应用程序主要包括如下几种类型的文件：

（1）.vbproj 文件：项目文件，用于描述项目细节，如内容、名称、版本等。

（2）.vbproj.user 文件：项目中的用户选项文件。

（3）.sln 文件：解决方案文件，描述解决方案的信息，包括解决方案中的项目细节。

（4）.vb 文件：代码文件，包含应用程序所有的源代码。

（5）.resx 文件：程序集资源文件，包含程序使用的一些资源（如图片），当应用程序发布后该文件可以删除，该文件是完全可读和可维护的。

（6）.pdb 文件：程序数据库文件，包含应用程序调试和项目状态信息。

（7）.exe 文件：可执行文件，该文件可以在装有.NET Framework 的系统中运行。

（8）.dll 文件：库文件，程序编译后生成的文件，可以被其他程序使用。

本章小结

（1）数据类型可分为值类型和引用类型，值类型数据执行效率较高，数据在堆栈中分配；引用类型执行效率较低，数据在托管堆中分配，堆栈中只分配指向堆的引用。如果数据中只是值类型，则需要的堆栈空间较大；如果只有引用类型，则执行的效率较低，所以值类型和引用类型都不可少。值类型又可进一步分为多个类型，它们之间存在收缩转换和扩大转换，一般扩大转换是比较安全的转换，而收缩转换是不安全的，需要显式声明，可以使用类型关键字、Ctype 方法和 Convert 类三种方法进行转换。值类型和引用类型之间的转换被称作装箱和拆箱。

（2）变量的值在程序运行过程中可以改变，而常量不可改变，它们遵守一套命名规则。另外变量有自己的生命周期和作用域，与变量的声明位置有关。变量的生命周期和作用域是十分重要的概念，需要很好地理解。

（3）运算符用于对操作数进行特定的运算，表达式是由运算符和操作数组成的式子，语句是程序完成一次完整操作的基本单位。这些都是 Visual Basic 最基本的语法，应该熟练掌握。

（4）程序一般是顺序执行的，若要改变程序执行顺序，需要控制结构，控制结构可分为顺序、选择和循环三个主要的控制结构，语句在三种结构中如何执行需要很好地掌握。

（5）任何编程语言都有一套编码规范，遵守代码规范编写程序是一个好的习惯，对后期代码的维护和重构都有非常大的帮助。

（6）本书所有示范代码都是基于 Visual Studio 2005 开发的控制台应用程序或 Windows 应用程序，但是也要掌握在没有 Visual Studio 2005 的情况下的代码编写、编译和运行的方法。

习题

一、填空题

1．_____年 BASIC 语言问世，_____年 Visual Basic.NET 诞生。

2．Visual Basic 3.0 将_____数据库驱动集成到了 Visual Basic 中，Visual Basic 5.0 引

入_____的概念。Visual Basic 6.0 集成了_____，提供了一种访问数据库的全新方法。

3．使用 vbc.exe 编译.vb 文件时，/out 的作用是_____，/t 的作用是_____，/r 的作用是_____。

4．Visual Basic 源代码遵循最基本的顺序结构，语句从_____向_____，每行一条，按序执行。

5．当代码较长时，可以将较长的打断为几行，此时可以使用续行符。续行符是指在行的末尾同时键入_____。

6．若变量 str = "Hello Visual Basic.NET"，则 Mid(str,6,3)= "Hell Word"，str=_____。

7．.vbproj 文件是_____文件，.sln 文件是_____文件。

8．程序代码有两种注释的方法，分别使用_____和_____。

9．基本控制结构有_____、_____、_____嵌套控制结构和其他结构。

10．“Const limit As Integer=33”代码中包含了_____和赋值两种语句。

二、选择题

1．在 Visual Studio.NET 中创建的 VB 项目，其项目文件 MyProject.vbproj 的存储格式为（　　）。

A）一般 TXT 格式　　B）二进制格式

C）HTML 格式　　D）XML 格式

2．以下可以作为变量名的是（　　）。

A）a#B　　B）vara

C）14b　　D）?ccc

3．下面是一段 For…Next 循环语句：

```
Dim J As Integer
      For J = 2 To 10 Step 2
         Dim i As Integer = 0
         i = i + J
      Next J
```

程序运行结束 i 的值是（　　）。

A）30　　B）10

C）20　　D）40

4．以下能从字符串“Visual Basic.NET”中取出“Basic”字符串的函数是（　　）。

A）Left　　B）Mid

C）String　　D）Instr

5．以下不属于 Visual Basic.NET 系统的文件类型是（　　）。

A）.cs　　B）.vb

C）.vbproj　　D）.sln 文件

6．有如下一段程序：

```
Dim a As Integer = 5
      Dim c As Integer = 4
      Dim b As Integer = 3
```

```
        Dim d As Integer = 2
        Dim bl As Boolean
    bl = a > b + 1 Or c < d And c Mod d
```

则 bl 的值是（　　）。

A）True　　B）False

C）1　　D）0

7．假定有以下循环结构：

```
Do until 条件
循环体
Loop
```

则下列说法正确的是（　　）。

A）如果“条件”是一个为 0 的常数，则一次循环体也不执行

B）如果“条件”是一个为 1 的常数，则至少执行一次循环体

C）如果“条件”是 False，则至少执行一次循环体

D）不论“条件”是否为 True，至少要执行一次循环体

8．阅读如下程序：

```
Dim n As Integer = 10
    Dim x As Integer
        Select Case n
            Case 1 To 20
                x = 10
            Case 2, 4, 6
                x = 20
            Case Is < 10
                x = 30
            Case 10
                x = 40
        End Select
Console.WriteLine("x=" & x.ToString())
```

程序运行结果为（　　）。

A）x=10　　B）x=20

C）x=30　　D）x=40

9．退出 For 循环可使用的语句为（　　）。

A）Exit For　　B）Exit Do

C）End For　　D）End Do

10．字符&在 Visual Basic.NET 中用作（　　）。

A）注释引导标志符　　B）字符串连接符

C）字符串限定符　　D）行继续符

11．关于值类型，下列说法正确的是（　　）。

A）每个值类型的变量都含有它自己的数据拷贝

B）Integer 类型属于值类型，因此值相同的整型变量在内存中的存放地址也相同

C）Visual Basic.NET 中的 variant 类型不是值类型

D）System.Object 类型是一种最基本的值类型

12．关于引用类型，下列说法正确的是（　　）。

A）引用类型变量包含指向对象实例的引用或指针

B）两个或多个引用类型变量可以引用同一个对象

C）Visual Basic.NET 中的 Variant 类型是一种引用类型

D）引用类型比值类型更安全

13．在下列 Visual Basic.NET 类型中，选出所有的值类型（　　）。

A）Integer 类型　　B）Char 类型

C）String 类型　　D）Struct 类型

14．在 Visual Basic.NET 中，Integer 类型的变量在计算机中占用（　　）位。

A）16 位　　B）32 位

C）64 位　　D）与具体的计算机系统相关

15．Visual Basic.NET 中，取字符串“this is a test”的第一个字符，并将其赋值给字符变量 ch。下列代码（　　）组正确（假设已经定义变量 ch）。

A）ch = Char(Mid("this is a test",0,1))

B）ch = CChar(Mid("this is a test",0,1))

C）ch = Char(Mid("this is a test",1,1))

D）ch = CChar(Mid("this is a test",1,1))

16．Visual Basic.NET 编译器开关 Option Strict On 的作用是（　　）。

A）在使用变量之前必须声明

B）变量声明语句中不能省略 as 子句

C）禁止对变量进行任何形式的类型窄转换

D）禁止对变量进行任何形式的类型转换

17．下列（　　）属于合法的变量名。

A）$money　　B）@Email

C）_Home　　D）Integer

18．在过程 showMyOfficeInfo 中，表示地址的局部变量应起名为（　　）。

A）Dim myAddress as String　　B）Dim MyAddress as String

C）Dim aksdiqieqew as String　　D）Dim @myAddress as String

19．在 Visual Basic.NET 的过程 MySub 中，下列变量声明语句（　　）正确。

A）Dim x,y,z as Integer　　B）Dim x as Integer,y as String,z as Decimal

C）Private temp as Long　　D）Public temp as Long

20．下列（　　）Visual Basic.NET 类型之间的转换属于收缩转换。

A）Integer→Long　　B）Data→String

C）Double→String　　D）Double→Long

21．执行下列语句后，控制台输出结果为（　　）。

```
Dim res as Integer = CType("123.45",Integer)
Console.Writeln(res)
```

A）123　　B）12345

C）0　　D）res

22．执行下列语句后，变量 x 的值为（　　）。

```
Dim x as Integer = 0
x = 3^2
```

A）0　　B）5

C）6　　D）9

23．执行下列语句后，变量 x 的值为（　　）。

```
Dim x as Integer = 0
x = 3/2
```

A）0　　B）1

C）2　　D）3

24．下列布尔表达式的值为（　　）。

```
True and False OrElse True and True
```

A）True　　B）False

C）OrElse　　D）AndAlso

25．下列（　　）语句可以完成判断 n1/n2 是否等于 m 的任务（n1，n2，m 都是整数），并且在 n2 等于 0 时也能正常运行。

A）If n1/n2=m Then OK=True

B）If n2<>0 Then OK=(n1/n2=m)

C）If n2<>0 AndAlso n1/n2=m Then OK=True

D）If n2<>0 And n1/n2=m Then OK=True

26．在 Select-Case 语句中，当 Case 变量 x 取值为 0<x<100 时，在控制台上输出"OK"。则该段 Case 子句应该选择（　　）。

A）Case 1..99
　　Console.WriteIn("OK")

B）Case 1 to 99
　　Console.WriteIn("OK")

C）Case 1,99
　　Console.WriteIn("OK")

D）Case 0<x<100
　　Console.WriteIn("OK")

27．执行下列语句后，整型变量 counter 的值是（　　）。

```
For counter =0 to 4
counter +=1
Next Counter
```

A）3　　B）4

C）5　　D）6

28．在设计循环语句时，若有以下要求：

①在执行循环之前先测试条件表达式 expression；

②让代码循环执行到该条件表达式的值取“真”。

则应使用下列（　　）Do Loop 循环语句。

A）Do Until expression

　　Loop

B）Do

　　Loop Until expression

C）Do While expression

　　Loop

D）Do

　　Loop While expression

29．执行下列代码后，字符串 firstName 的值是（　　）。

```
Dim firstName as String = "Tom"
Dim secondName as String = firstName
if  secondName is firstName then
   secondName &= "Jerry"
end if
```

A）Tom　　B）Jerry

C）TomJerry　　D）JerryTom

三、判断题

1．下划线“_”在 Visual Basic.NET 中用作注释引导标志符。（　　）

2．在 Visual Basic.NET 中，下列语句是合法的。（　　）

```
Dim x as Object
x=100
x="this ia a test"
```

3．在 Visual Basic.NET 中，一个布尔型变量占用的内存位数为 16 位。（　　）

4．如果在模块文件 MyMoudle.VB 中第 1、2 行设置编译器开关如下：

```
Option Strict Off
Option Explicit On
```

则该文件中的下列过程定义是合法的。（　　）

```
Public Sub hey()
   Dim x, y
   x = 100
   y = "hello"
End Sub
```

5．编译器开关设置为 Option Strict On，语句 Const pi = 3.1415926 是正确的常量声明。（　　）

6．如下这段代码的语法是正确的。（　　）

```
Dim i as Integer = 100
```

```
If i Then
  MessageBox.Show("i is not Zero")
End if
```

7．如下所示的这段代码结果是 OK。　（　　）

```
Dim t as Boolean = True
DIm f as Boolean = False
if f=t Then
  Console.Writeln("OK")
else
  Console.Writeln("WRONG")
End if
```

8．执行如下代码后，x 是 False。　（　　）

```
Dim x as Boolean = True
x = False<True
```

9．具有相同优先顺序的运算符将按照它们在表达式中出现的顺序从右至左进行计算。（　　）

10．Visual Basic 语言有一套命名约定，但在程序开发时可以不遵守。　（　　）

四、简答题

1．简述开发 Visual Basic.NET 应用程序的基本步骤。

2．简述 Visual Basic.NET 的程序组成。

3．简述变量的命名规则。

4．简述值类型和引用类型的区别。

五、操作题

1．使用递归算法编写如下程序：对于任意给定的实数 X 和整数 k>0，计算 X^k。

2．编写一个函数，求任意整数数组的最大元素。

3．编写一个程序，完成以下要求：

（1）提示用户输入任意的 3 个小数。

（2）显示这三个小数。

（3）将这三个小数相加，并显示其结果。

（4）将结果按四舍五入方法转换成整数并显示。

4．开发一个控制台程序，使之能够读取从键盘录入的“Hello Visual Basic.NET”字符串，并能在屏幕上显示。

5．使用记事本编写一段代码，并使用 vbc.exe 命令编译、执行。