第 3 章 程序流程控制(一)

C＃程序包括 4 种基本控制结构：顺序结构、选择结构、循环结构和异常处理逻辑结构。本章主要介绍顺序结构和选择结构。

本章要点：

- 布尔数据类型
- 运算符：逻辑运算符、关系和类型测试运算符、条件运算符、其他运算符以及运算符优先级
- 顺序结构
- 选择结构：if 语句和 switch 语句

3.1 布尔数据类型

C＃的布尔(bool)数据类型用于逻辑运算，包含 bool 值 True 或 False，如表 3-1 所示。

表 3-1 bool 数据类型

名　　称	CTS 类型	说　　明	值
bool	System. Boolean	布尔类型	True 或 False

bool 值和整数值不能相互转换。如果变量(或函数的返回类型)声明为 bool 类型，就只能使用值 True 或 False。例如：

```
bool b1 = True;
bool b2 = 0;          //编译错误：常量值“0”无法转换为“bool”
```

3.2 运算符

3.2.1 逻辑运算符

逻辑运算符除了逻辑非(!)是一元运算符外，其余均为二元运算符，用于将操作数进行逻辑运算，结果为 True 或 False。表 3-2 按优先级从高到低的顺序列出了 C＃中的逻辑运算符。

表 3-2　C# 中的逻辑运算符

运算符	含　义	说　明	优先级	实　例	结　果
!	逻辑非	当操作数为 False 时返回 True；当操作数为 True 时返回 False	1	!True !False	False True
&	逻辑与	两个操作数均为 True 时，结果才为 True，否则为 False	2	True & True True & False False & True False & False	True False False False
^	逻辑异或	两个操作数不相同，即一个为 True、另一个为 False 时，结果才为 True，否则为 False	3	True ^ True True ^ False False ^ True False ^ False	False True True False
|	逻辑或	两个操作数中有一个为 True 时，结果即为 True，否则为 False	4	True | True True | False False | True False | False	True True True False
&&	条件与	两个操作数均为 True 时，结果才为 True。但仅在必要时才计算第二个操作数	5	True && True True && False False && True False && False	True False False False
||	条件或	两个操作数中有一个为 True 时，结果即为 True。但仅在必要时才计算第二个操作数	6	True || True True || False False || True False || False	True True True False

注意：

(1) 二元逻辑与(&)运算符。对于整型，“&”计算操作数的逻辑按位与；对于 bool 操作数，“&”计算操作数的逻辑与；另外，一元“&”运算符返回操作数的地址。

(2) 二元逻辑异或(^)运算符。对于整型，“^”计算操作数的按位异或；对于 bool 操作数，“^”计算操作数的逻辑异或。

(3) 二元逻辑或(|)运算符。对于整型，“|”计算操作数的按位或结果。对于 bool 操作数，“|”计算操作数的逻辑或结果。

(4) 条件与(&&)执行其 bool 操作数的逻辑与运算，但仅在必要时才计算第二个操作数。即“x && y”对应于操作“x&y”。不同的是，如果 x 为 False，则不计算 y(因为不论 y 为何值，与操作的结果都为 False)。这被称为“短路”计算。

(5) 条件或(||)运算符执行 bool 操作数的逻辑或运算，但仅在必要时才计算第二个操作数。即“x||y”对应于操作“x|y”。不同的是，如果 x 为 True，则不计算 y(因为不论 y 为何值，或操作的结果都为 True)。这被称为“短路”计算。

3.2.2　关系和类型测试运算符

关系和类型测试运算符是二元运算符。关系运算符用于将两个操作数的大小进行比

较。若关系成立，则比较的结果为 True，否则为 False。关系运算符的操作数可以是数值型、字符型、枚举类型。假设有如下定义：

```
int[] myArray = new int[]{1, 2};
```

表 3-3 列出了 C# 中的关系和类型测试运算符。

表 3-3 C# 中的关系和类型测试运算符

运 算 符	含 义	实 例	结 果
==	相等	"ABCDEF" == "ABCD"	False
!=	不等	"ABCD" != "abcd"	True
>	大于	"ABC" > "ABD"	False
>=	大于等于	123 >= 23	True
<	小于	"ABC" < "上海"	True
<=	小于等于	"123" <= "23"	True
x is T	数据 x 是否属于类型 T	myArray is int myArray is int[]	False True
x as T	返回转换为类型 T 的 x，如果 x 不是 T 则返回 null	myArray as int[] myArray as object	System.Int32[] System.Int32[]

注意：

(1) 关系运算符的优先级相同。

(2) 对于两个预定义的数值类型，关系运算符按照操作数的数值大小进行比较。

(3) 对于 string 类型，关系运算符比较字符串的值，即按字符的 ASCII 码值从左到右一一比较：首先比较两个字符串的第一个字符，其 ASCII 码值大的字符串大，若第一个字符相等，则继续比较第二个字符，依此类推，直至出现不同的字符为止。

(4) 对于 string 以外的引用类型，如果两个操作数引用同一个对象，则“==”返回 True。如果两个操作数引用不同的对象，则“!=”返回 True。

(5) int 和 System.Int32 是相同的数据类型。

3.2.3 条件运算符

使用条件运算符，可以更简洁、直观地表达某些 if-else 结构的计算。条件运算符是 C# 中唯一的三元运算符，由符号“?”和“:”组成，其一般形式为：

```
逻辑表达式? 表达式 1: 表达式 2;
```

首先计算“逻辑表达式”的值，如果为 True，则运算结果为“表达式 1”的值，否则运算结果为“表达式 2”的值。例如，计算 a 和 b 两个数中较大的数，并将其赋给变量 maxnum 中，语句为：

```
maxnum = (a > b)? a : b;
```

等价于：

```
if (a>b)
        maxnum = a;
else
        maxnum = b;
```

3.2.4 其他运算符

sizeof 用于获取值类型的字节大小，仅适用于值类型，而不适用于引用类型。sizeof 运算符只能在不安全代码块中使用。

typeof 用于获取类型的 System. Type 对象，例如“System. Type type = typeof(int);”。若要获取表达式的运行时类型，可以使用.NET Framework 方法 GetType()。例如：

```
int i = 0; System.Type type = i.GetType();
```

3.2.5 运算符优先级

表达式中的运算符按照运算符优先级(precedence)的特定顺序和结合性规则计算。C#语言定义的运算符优先级顺序和结合性规则如表 3-4 所示。表 3-4 按优先级从高到低的顺序列出各运算符类别，同一类别中的运算符优先级相同。

表 3-4 C#语言定义的运算符优先级顺序和结合性规则

类 别	运 算 符	说 明	结合性
基本	x. m	成员访问	→
	f(x)	方法和委托调用	
	a[x]	数组和索引器访问	
	−>	指针成员访问	
	x++	后增量	
	x−−	后减量	
	new	创建对象、数组或委托	←
	typeof(T)	获得 T 的 System. Type 对象	
	checked	在 checked 上下文中计算表达式	
	unchecked	在 unchecked 上下文中计算表达式	
一元	+	恒等	←
	−	求相反数	
	!	逻辑求反	
	~	按位求反	
	++x	前增量	
	−−x	前减量	
	(T)x	显式将 x 转换为类型 T	
	&	取操作数的地址	
	sizeof(T)	获取类型 T 的字节大小	
乘除	*	乘法	→
	/	除法	
	%	求余	

续表

类　别	运　算　符	说　明	结合性
加减	+	加法、字符串串联、委托组合	→
	−	减法、委托移除	
移位	<<	左移	→
	>>	右移	
关系和类型检测	<	小于	→
	>	大于	
	<=	小于或等于	
	>=	大于或等于	
	x is T	如果 x 属于 T 类型，则返回 true，否则返回 false	
	x as T	返回转换为类型 T 的 x，如果 x 不是 T 则返回 null	
相等	==	等于	→
	!=	不等于	
逻辑与	x & y	整型按位 AND，布尔逻辑 AND	→
逻辑异或	x ^ y	整型按位 XOR，布尔逻辑 XOR	→
逻辑或	x | y	整型按位 OR，布尔逻辑 OR	→
条件与	x && y	仅当 x 为 true 时才对 y 求值	→
条件或	x || y	仅当 x 为 false 时才对 y 求值	→
条件运算	x ? y : z	如果 x 为 true，则对 y 求值；如果 x 为 false，则对 z 求值	←
赋值和匿名函数	=	赋值	←
	+=，−=，*=，/=，%=，&=，|=，^=，<<=，>>=	复合赋值	
	=>	lambda 表达式	

当表达式中出现两个具有相同优先级的运算符时，则根据结合性进行计算。左结合运算符按从左到右（表中用符号“→”示意）的顺序计算，例如，$x * y / z$ 计算为 $(x * y) / z$。右结合运算符按从右到左（表中用符号“←”示意）的顺序计算。**注意**：C#语言中赋值运算符和条件运算符（?:）是右结合运算符，例如，$x = y = z$ 计算为 $x = (y = z)$；其他所有二元运算符都是左结合运算符。

优先级和结合性都可以用括号控制。例如，2 + 3 * 2 的计算结果为 2 + (3 * 2) = 8；而(2 + 3) * 2 的计算结果为 10。再如：

```
bool b = 16 + 2 * 5 >= 7 * 8 / 2 || "XYZ" != "xyz" && !(10 - 6 > 18 / 2);
```

相当于：

```
bool b = ((16 + (2 * 5)) >= ((7 * 8) / 2)) || (("XYZ" != "xyz") && (!((10 - 6) > (18 /
2))));
```

结果为：True。

3.3 顺序结构

C#程序中语句执行的基本顺序按各语句出现位置的先后次序执行，称之为顺序结构，如图 3-1 所示。先执行语句块 1，再执行语句块 2，最后执行语句块 3。三者是顺序执行关系。

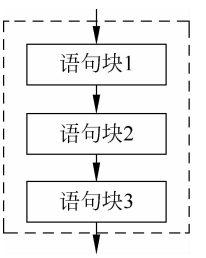


图 3-1 顺序结构示意图

【例 3-1】 顺序结构示例(T3Sequence.cs)：已知三角形的三条边(为简单起见，假设这三条边可以构成三角形)，求三角形的面积。

```
using System;
namespace CSharpBook.Chapter03
{   class Sequence
    {   static void Main()
        {   double a, b, c, p, area;
            Console.Write("请输入三角形的边 A: "); String s = Console.ReadLine();
            a = double.Parse(s);                    //将数字字符串转换为等效的双精度浮点数
            Console.Write("请输入三角形的边 B: "); b = double.Parse(Console.ReadLine());
            Console.Write("请输入三角形的边 C: "); c = double.Parse(Console.ReadLine());
            Console.WriteLine("三角形的三条边分别为: a={0}, b={1}, c={2}", a,b,c);
            p = (a + b + c) / 2;                                  //三角形周长的一半
            area = Math.Sqrt(p*(p-a)*(p-b)*(p-c));        //计算三角形面积
            Console.WriteLine("三角形的面积 = {0}",area); Console.ReadLine();
        }
    }
}
```

运行结果：

```
请输入三角形的边 A:  3
请输入三角形的边 B:  4
请输入三角形的边 C:  5
三角形的三条边分别为: a=3, b=4, c=5
三角形的面积 = 6
```

3.4 选择结构

选择结构可以根据条件来控制代码的执行分支，也叫分支结构。C#包括两种控制分支的条件语句：if 语句和 switch 语句。

3.4.1 if 语句

if 语句的选择结构包含多种形式：单分支、双分支和多分支，流程分别如图 3-2(a)、(b)、(c)所示。

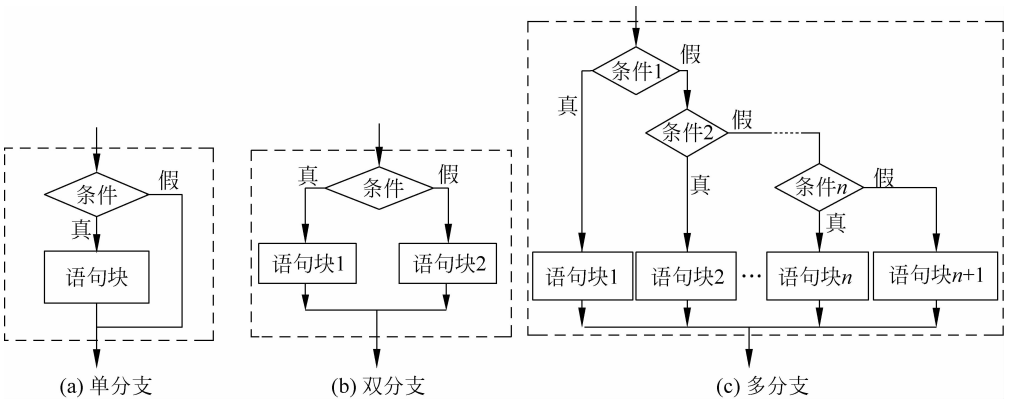


图 3-2　if 语句的选择结构

1. 单分支结构

if 语句单分支结构的语法形式如下：

```
if (条件表达式)
    语句/语句块 statement(s);
```

其中：

(1) 条件表达式：可以是关系表达式、逻辑表达式、算术表达式等。

(2) 语句/语句块 statement(s)：可以是单个语句，也可以是多个语句。如果要执行多个语句，则需要使用花括号({ ... })把这些语句组合为一个代码块。

该语句的作用是当条件表达式的值为真时，执行 if 后的语句(块)，否则不做任何操作，控制将转到 if 语句的结束点。其流程如图 3-2(a)所示。

【例 3-2】 单分支结构示例(T3SingleDecision.cs)：产生两个 0～100 之间的随机数 a 和 b，比较两者大小，使得 a 大于 b。

```
using System;
namespace CSharpBook.Chapter03
{   class SingleDecision
    {   static void Main()
        {   int a, b, t; Random rNum = new Random();
            a = rNum.Next(101); b = rNum.Next(101);     //产生 0～100 之间的随机整数 a、b
            Console.WriteLine("原始值: a = {0}, b = {1}", a, b);
            if (a < b) { t = a; a = b; b = t; }
            Console.WriteLine("降序值: a = {0}, b = {1}", a, b); Console.ReadLine();
        }
    }
}
```

运行结果(其中，a 和 b 随机生成)：

```
原始值: a = 51, b = 64
降序值: a = 64, b = 51
```

2. 双分支结构

if 语句双分支结构的语法形式如下：

```
if (条件表达式)
    语句/语句块 1;
else
    语句/语句块 2;
```

该语句的作用是当条件表达式的值为真(True)时，执行 if 后的语句(块)1，否则执行 else 后的语句(块)2，其流程如图 3-2(b)所示。

【例 3-3】 计算分段函数：

$$y=\begin{cases}\sin x+2\sqrt{x+e^4}-(x+1)^3 & x\geqslant 0\\ \ln(-5x)-\dfrac{|x^2-8x|}{7x}+e & x<0\end{cases}$$

此分段函数有以下几种实现方式，请读者自行编程测试。

(1) 利用单分支结构实现。

一句单分支语句：

```
y = Math.Sin(x) + 2 * Math.Sqrt(x + Math.Exp(4)) - Math.Pow(x + 1, 3);
if (x<0)
    y = Math.Log(-5 * x) - Math.Abs(x * x - 8 * x) / (7 * x) + Math.E;
```

或两句单分支语句：

```
if (x>=0)
    y = Math.Sin(x) + 2 * Math.Sqrt(x + Math.Exp(4)) - Math.Pow(x + 1, 3);
if (x<0)
    y = Math.Log(-5 * x) - Math.Abs(x * x - 8 * x) / (7 * x) + Math.E;
```

(2) 利用双分支结构实现。

```
if (x>=0)
    y = Math.Sin(x) + 2 * Math.Sqrt(x + Math.Exp(4)) - Math.Pow(x + 1, 3);
else
    y = Math.Log(-5 * x) - Math.Abs(x * x - 8 * x) / (7 * x) + Math.E;
```

(3) 利用条件运算符实现。

```
y = (x>=0)? Math.Sin(x) + 2 * Math.Sqrt(x + Math.Exp(4)) - Math.Pow(x + 1, 3): Math.Log
(-5 * x) - Math.Abs(x * x - 8 * x) / (7 * x) + Math.E;
```

3. 多分支结构

if 语句多分支结构的语法形式如下：

```
if (条件表达式 1)
   语句/语句块 1;
else if (条件表达式 2)
   语句/语句块 2;
…
else if (条件表达式 n)
   语句/语句块 n;
[else
   语句/语句块 n + 1;]
```

该语句的作用是根据不同的条件表达式以确定执行哪个语句(块),其流程如图 3-2(c)所示。

【例 3-4】 已知某课程的百分制分数(mark),将其转换为五级制(优、良、中、及格、不及格)的评定等级(grade)。评定条件如下:

$$\text{成绩等级} = \begin{cases} \text{优} & \text{mark} \geqslant 90 \\ \text{良} & 80 \leqslant \text{mark} < 90 \\ \text{中} & 70 \leqslant \text{mark} < 80 \\ \text{及格} & 60 \leqslant \text{mark} < 70 \\ \text{不及格} & \text{mark} < 60 \end{cases}$$

根据评定条件,有以下三种不同的实现方法。

方法 1:

```
if (mark >= 90)
    grade = "优";
else if (mark >= 80)
    grade = "良";
else if (mark >= 70)
    grade = "中";
else if (mark >= 60)
    grade = "及格";
else
    grade = "不及格";
```

方法 2:

```
if (mark >= 90)
    grade = "优";
else if (mark >= 80 && mark < 90 )
    grade = "良";
else if (mark >= 70 && mark < 80)
    grade = "中";
else if (mark >= 60 && mark < 70)
    grade = "及格";
else
    grade = "不及格";
```

方法 3：

```
if (mark >= 60)
    grade = "及格";
else if (mark >= 70)
    grade = "中";
else if (mark >= 80)
    grade = "良";
else if (mark >= 90)
    grade = "优";
else
    grade = "不及格";
```

其中：方法 1 中使用关系运算符“>=”，按分数从大到小依次比较；方法 2 使用关系运算符和逻辑运算符表达完整的条件，即使语句顺序不按比较的分数从大到小依次书写，也可以得到正确的等级评定结果；方法 3 使用关系运算符“>=”，但按分数从小到大依次比较。

上述三种方法中，方法 1、方法 2 正确，而且方法 1 最简捷明了，方法 2 虽然正确，但是存在冗余条件。方法 3 虽然语法没有错误，但是判断结果错误，根据分数(mark)所得等级评定结果只有“及格”和“不及格”两种，请读者根据程序流程自行分析原因。

【例 3-5】 已知坐标点(x,y)，判断其所在的象限。相关语句如下：

```
if (x > 0 && y > 0) Console.WriteLine("x = {0}, y = {1}, 位于第一象限", x, y);
else if (x < 0 && y > 0) Console.WriteLine("x = {0}, y = {1}, 位于第二象限", x, y);
else if (x < 0 && y < 0) Console.WriteLine("x = {0}, y = {1}, 位于第三象限", x, y);
else if (x > 0 && y < 0) Console.WriteLine("x = {0}, y = {1}, 位于第四象限", x, y);
else if (x == 0 && y == 0) Console.WriteLine("x = {0}, y = {1}, 位于原点", x, y);
else if (x == 0) Console.WriteLine("x = {0}, y = {1}, 位于 y 轴", x, y);
else Console.WriteLine("x = {0}, y = {1}, 位于 x 轴", x, y);
```

4. if 语句的嵌套

在 if 语句中又包含一条或多条 if 语句称为 if 语句的嵌套。一般形式如下：

```
if (条件表达式 1)
    if (条件表达式 11)
        语句 1;
    [else
        语句 2;]
[else
    if (条件表达式 21)
        语句 3;
    [else
        语句 4;]]
```

为了正确表达 if 语句的嵌套关系，建议读者使用花括号以及缩进格式确定 if 和 else 的配对关系。语句形式如下：

```
if (条件表达式 1)
{
    if (条件表达式 11)
    {
        语句/语句块 1;
    }
    [else
    {
        语句/语句块 2;
    }]
}
[else
{
    if (条件表达式 21)
    {
        语句 3;
    }
    [else
    {
        语句 4;
    }]
}]
```

【例 3-6】 计算分段函数：

$$y=\begin{cases}1 & x>0\\ 0 & x=0\\ -1 & x<0\end{cases}$$

此分段函数有以下几种实现方式，请读者判断哪些是正确的，并自行编程测试正确的实现方式。

方法 1：

```
if(x>0)
    y=1;
else if(x==0)
    y=0;
else
    y=-1;
```

方法 2：

```
if(x>=0)
{
    if(x>0)
        y=1;
    else
        y=0;
}
else  y=-1;
```

方法 3：

```
y=1;
```

```
if(x! = 0)
{    if(x < 0)
        y = -1;
}
else
    y = 0;
```

方法 4：

```
y = 1;
if(x! = 0)
    if(x < 0)
        y =- 1;
else
        y = 0;
```

请读者画出每种方法相应的流程图，并进行分析测试。其中，方法 1、方法 2 和方法 3 是正确的，而方法 4 是错误的，相当于如下语句：

```
y = 1;
if (x ! = 0)
    if (x < 0)
        y = -1;
    else
        y = 0;
```

因为在嵌套的 if 语句中，else 部分是和语法允许的、词法上最相近的上一个 if 语句相配对的。

【例 3-7】 已知字符变量 ch 中存放了一个字符，判断该字符是字母字符（并进一步判断是大写字母还是小写字母）、数字字符还是其他字符，并给出相应的提示信息。相关语句如下：

方法 1。利用系统提供的方法：

```
if (Char.IsLetter(ch))
{   if (Char.IsUpper(ch)) Console.WriteLine("字符 {0} 是大写字母", ch);
    else Console.WriteLine("字符 {0} 是小写字母", ch);
}
else if (Char.IsNumber(ch)) Console.WriteLine("字符 {0} 是数字字符", ch);
else Console.WriteLine("字符 {0} 是其他字符", ch);
```

方法 2。利用字符比较：

```
if (Char.ToUpper(ch) >= 'A' && Char.ToUpper(ch) <= 'Z')
{   if (ch >= 'A' && ch <= 'Z') Console.WriteLine("字符 {0} 是大写字母", ch);
    else Console.WriteLine("字符 {0} 是小写字母", ch);
}
else if (ch >= '0' && ch <= '9') Console.WriteLine("字符 {0} 是数字字符", ch);
else Console.WriteLine("字符 {0} 是其他字符", ch);
```

【例 3-8】 输入 3 个数，按从大到小的顺序排序。

方法 1：先 a 和 b 比较，使得 $a>b$；然后 a 和 c 比较，使得 $a>c$，此时 a 最大；最后 b 和 c

比较，使得 $b>c$。相关语句如下：

```
if (a < b){ t = a; a = b; b = t; }
if (a < c){ t = a; a = c; c = t; }
if (b < c){ t = b; b = c; c = t; }
```

方法 2：利用 Max 函数和 Min 函数求 a、b、c 3 个数中的最大数和最小数，而 3 个数之和减去最大数和最小数就是中间数。相关语句如下：

```
Nmax = Math.Max(Math.Max(a,b),c);
Nmin = Math.Min(Math.Min(a,b),c);
Nmid = a + b + c - Nmax - Nmin;
a = Nmax; b = Nmid; c = Nmin;
```

【例 3-9】 编程判断某一年是否为闰年。判断闰年的条件是：年份能被 4 整除但不能被 100 整除，或者能被 400 整除，其判断流程如图 3-3 所示。

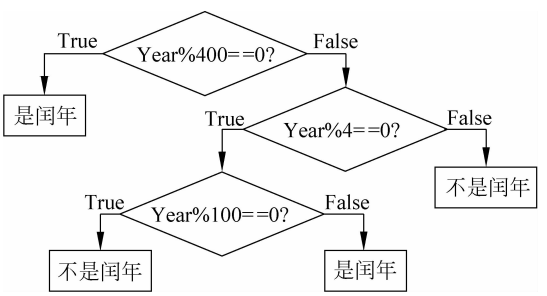


图 3-3 闰年的判断流程

方法 1：使用日期时间型变量的成员来判断闰年，相关语句如下：

```
if (DateTime.IsLeapYear(year)) Console.WriteLine("{0} year is a leap year!", year);
else Console.WriteLine("{0} year is not a leap year!", year);
```

方法 2：使用一个逻辑表达式包含所有的闰年条件，相关语句如下：

```
if ((year % 4 = = 0 && year % 100 != 0) || year % 400 = = 0)
    Console.WriteLine("{0} year is a leap year!", year);
else
    Console.WriteLine("{0} year is not a leap year!", year);
```

方法 3：使用嵌套的 if 语句，相关语句如下：

```
if (year % 400 = = 0)
    Console.WriteLine("{0} year is a leap year!", year);
else
{   if (year % 4 = = 0)
    {   if (year % 100 = = 0)Console.WriteLine("{0} year is not a leap year!", year);
        else Console.WriteLine("{0} year is a leap year!", year);
    }
```

```
    else Console.WriteLine("{0} year is not a leap year!", year);
}
```


方法 4：使用 if…else if 语句，相关语句如下：

```
if (year % 400 = = 0) Console.WriteLine("{0} year is a leap year!", year);
else if (year % 4 != 0) Console.WriteLine("{0} year is not a leap year!", year);
else if (year % 100 = = 0) Console.WriteLine("{0} year is not a leap year!", year);
else Console.WriteLine("{0} year is a leap year!", year);
```

3.4.2 switch 语句

对于多重分支，虽然可以使用嵌套的 if 语句实现，但是如果分支较多，则嵌套的 if 语句层次较多，结构比较复杂，可读性较差，此时可利用 switch…case 语句。switch 语句是一个控制语句，它通过将控制传递给其体内的一个 case 语句来处理多个选择和枚举，如图 3-4 所示。switch 语句的语法形式如下：

```
switch(控制表达式)
{
  case 常量表达式 1:
       语句序列 1;
       break;
  case 常量表达式 2:
       语句序列 2;
       break;
         .
         .
         .
  case 常量表达式 n:
       语句序列 n;
       break;
  default:
       语句序列 n + 1;
       break;
}
```

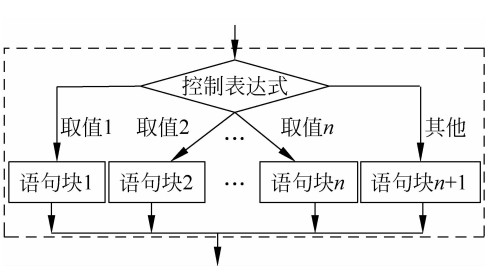


图 3-4　多分支选择结构(switch 语句)

【例 3-10】 根据考试成绩的等级输出百分制分数段的程序片断如下：

```
switch (grade)
{   case 'A':
        Console.WriteLine("'A' belongs to 90～100!");  break;
    case 'B':
        Console.WriteLine("'B' belongs to 80～90!");   break;
    case 'C':
        Console.WriteLine("'C' belongs to 70～80!");   break;
    case 'D':
        Console.WriteLine("'D' belongs to 60～70!");   break;
    case 'F':
        Console.WriteLine("'F' belongs to < 60!");     break;
    default:
        Console.WriteLine("Error character!");         break;
}
```

说明：

(1) switch 语句基于控制表达式的值选择要执行的语句分支。switch 语句按以下顺序执行：

① 控制表达式求值。

② 如果 case 标签后的常量表达式的值恰好等于控制表达式的值，控制将转到匹配的 case 标签后的语句序列。

③ 如果 case 标签后的常量表达式都不等于控制表达式的值，且如果存在一个 default 标签，则控制将转到 default 标签后的语句序列。

④ 如果 case 标签后的常量表达式都不等于控制表达式的值，且如果不存在 default 标签，则控制将跳出 switch 语句而执行后继语句。

(2) 控制表达式所允许的数据类型：整数类型(sbyte、byte、short、ushort、int、uint、long、ulong)，字符类型(char)，字符串类型(string)，或者枚举类型。

(3) 每一个 case 标签后的常量表达式的数据类型与控制表达式的类型相同，或可以隐式转换为控制表达式的类型。

(4) 每一个 case 标签后的常量表达式的值都不能相同。这包括值相同的不同常量。例如：

```
//假设 country 是 string 类型的字符串变量
const string England = "uk";
const string Britain = "uk";
switch(country)
{   case England:
    case Britain:                  //将导致编译错误!
        language = "English"; break;
}
```

(5) 各个 case 子句出现的次序不影响语句的执行结果。例如，可以先出现“case 'F'：……”，然后再是“case 'A'：……”。

(6) 一个 switch 语句中最多只能有一个 default 标签。虽然 switch 结构中的 case 可以任意顺序放置,但是一般建议将 default 情况放在最后。

(7) 和 if-else 语句不同的是,在执行一个 case 子句后面的语句序列后,switch 语句并不会终止,要退出 switch 语句,必须使用一个跳转语句。一般情况下,跳转语句是 break 语句。当到达 break 语句时,代码执行跳转到 switch 语句后面的语句。也可以使用 goto 语句达到同样的目的(但一般不建议使用这个语句)。合法的 goto 语句的例子诸如“goto default”、“goto case 'A'”等。如果没有指定 break 或其他跳转语句来结束 case 或 default 子句的执行,C#将产生编译错误。

(8) case 标签后的语句序列可以包含一条以上的执行语句,这些语句序列可以不加花括号,C#会自动顺序执行本 case 标签后所有的执行语句。

(9) 多个 case 标签可以共用同一组语句序列。例如:

```
…
    case 'A':
    case 'B':
    case 'C':
    case 'D':
        Console.WriteLine("'A' 'B' 'C' 'D' belong to > 60!");
…
```

grade 的值为'A'、'B'、'C'或'D'时均执行同一组语句。

习题 3

1. 单选题

(1) 以下关于 C#中 bool 数据类型的说法,不正确的是_____。

A. bool 数据类型用于逻辑运算　　B. bool 值和整数值不能相互转换

C. bool 值包括 true 和 false　　D. 可以使用 0 表示 false,非 0 值表示 true

(2) 下面程序段求两个数 x 和 y 中的大数,_____是不正确的。

A. maxNum= (x>y)? x:y;　　B. if (x>y) maxNum=x;
else maxNum=y;

C. maxNum=Math.Max(x,y);　　D. if (y>=x) maxNum=y;
maxNum=x;

(3) 下面 if 语句统计“成绩(mark)优秀的男生以及不及格的男生”的人数,正确的语句为_____。

A. if (gender =="男" && mark<60 || mark>=90) n++;

B. if (gender =="男" && mark<60 && mark>=90) n++;

C. if (gender =="男" && (mark<60 || mark>=90)) n++;

D. if (gender =="男" || mark<60 || mark>=90) n++;

2. 填空题

(1) 数学关系式 $2<x\leqslant 10$ 表示成正确的 C#表达式为_____。

(2) 判断整数 i 能否同时被 3 和 5 整除的表达式为_____。

(3) 语句“int a = 2, b = 4; int r = a < b ? a++ : b++;”执行后，r、a 和 b 的值分别是_____。

(4) 已知“int a=3; int b=5; int c=6;bool d=true;”，则表达式“$a >=0$ && $a+c>b+3$ || ! d”的值是_____。

(5) 语句“bool m = true, n = false, p = true; bool b1 = m | n ^ p; bool b2 = n | m ^ p;”执行后，$b1$ 和 $b2$ 的值分别是_____。

(6) “int i=3; bool b=(++i==4)”执行后，b=_____。“int i=3; bool b=(i++==4)”执行后，b=_____。

3. 思考题

(1) 说明以下 3 个 if 语句的区别：

①
```
if (i > 0)
   if (j > 0) n = 1;
    else n = 2;
```

②
```
if (i > 0)
   { if (j > 0) n = 1;}
   else n = 2;
```

③
```
if (i > 0) n = 1;
   else if (j > 0) n = 2;
```

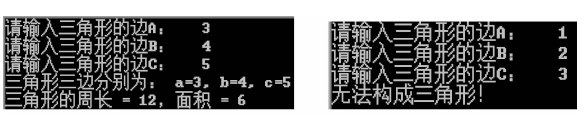
(2) 已知 i 为整型变量，阅读下面程序片段，请问输出结果是什么？

```
switch (i)
{   case 0:
        Console.WriteLine("0");
    case 1:
        Console.WriteLine("1");
    default:
        Console.WriteLine("Others");
}
```

实验 3

实验 3-1 求三角形周长和面积

实验要求：输入三角形三条边，先判断是否可以构成三角形，如果可以，则求三角形的周长和面积，否则报错。运行效果如图 3-5 所示。



(a) 三角形的周长和面积　　(b) 不能构成三角形

图 3-5　实验 3-1 运行效果

操作提示：

(1) 3 个数可以构成三角形必须满足如下条件：每条边长均大于 0，并且任意两边之和大于第三边。

(2) 已知三角形的三条边，则三角形的面积＝ $\sqrt{h*(h-a)*(h-b)*(h-c)}$，其中，$h$ 为三角形周长的一半。

实验 3-2 分段函数的实现

实验要求：输入 x，根据如下公式，计算分段函数 y 的值。要求参照课本例 3-3，分别利用“一句单分支语句”、“两句单分支语句”、“双分支结构”、“条件运算符”4 种方法实现。运行效果如图 3-6 所示。

$$y=\begin{cases}\dfrac{x^2-3x}{x+1}2\pi+\sin x & x\geqslant 0\\ \ln(-5x)+6\sqrt{|x|+e^4}-(x+1)^3 & x<0\end{cases}$$

实验 3-3 3 个数比较大小

实验要求：产生 3 个 0～100 之间(包含 0 和 100)的随机整数 a、b 和 c，按从小到大的顺序排序。运行效果如图 3-7 所示(其中，a、b、c 的值随机生成)。

```
请输入x:    5
方法一: x = 5, y = 6.99092769918312
方法二: x = 5, y = 6.99092769918312
方法三: x = 5, y = 6.99092769918312
方法四: x = 5, y = 6.99092769918312
```

图 3-6 实验 3-2 运行效果

```
原始值:   a=31,  b=4,  c=30
<方法一>升序值:   a=4,  b=30,  c=31
<方法二>升序值:   a=4,  b=30,  c=31
```

图 3-7 实验 3-3 运行效果

操作提示：

(1) 方法一：先 a 和 b 比较，使得 $a<b$；然后 a 和 c 比较，使得 $a<c$，此时 a 最小；最后 b 和 c 比较，使得 $b<c$。

(2) 方法二：利用 Max 函数和 Min 函数求 a、b、c 3 个数中的最大数和最小数，而 3 个数之和减去最大数和最小数就是中间数。

(3) 产生 0～100 之间(包含 0 和 100)随机整数的语句为：

```
Random rNum = new Random(); number = rNum.Next(101);
```

实验 3-4 求解一元二次方程

实验要求：输入一元二次方程的 3 个系数 a、b、c，求 $ax^2+bx+c=0$ 方程的解。运行效果如图 3-8 所示。

```
请输入系数a:    0
请输入系数b:    0
请输入系数c:    6
此方程无解!
```

(a) 无解

```
请输入系数a:    0
请输入系数b:    1
请输入系数c:    2
此方程的解为:  -2
```

(b) 一个实根

```
请输入系数a:    1
请输入系数b:    -2
请输入系数c:    1
此方程有两个相等实根:  1
```

(c) 两个相等实根

```
请输入系数a:    1
请输入系数b:    -1
请输入系数c:    -6
此方程有两个不等实根:  3 和 -2
```

(d) 两个不等实根

```
请输入系数a:    1
请输入系数b:    -1
请输入系数c:    0.5
此方程有两个不等实根:  0.5+0.5i 和 0.5-0.5i
```

(e) 两个共轭复根

图 3-8 实验 3-4 运行效果

操作提示：

方程 $ax^2+bx+c=0$ 的解有以下几种情况：

(1) $a=0$ and $b=0$，无解。

(2) $a=0$ and $b!=0$，有一个实根：$x=-\frac{c}{b}$。

(3) $b^2-4ac=0$，有两个相等实根：$x_1=x_2=-\frac{b}{2a}$。

(4) $b^2-4ac>0$，有两个不等实根：$x_1=\frac{-b+\sqrt{b^2-4ac}}{2a}$，$x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}$。

(5) $b^2-4ac<0$，有两个共轭复根：$x_1=-\frac{b}{2a}+\frac{\sqrt{4ac-b^2}}{2a}i$，$x_2=-\frac{b}{2a}-\frac{\sqrt{4ac-b^2}}{2a}i$。

实验 3-5 显示数字(1～7)所对应的星期(星期一～星期日)

实验要求：利用 switch 语句实现多重分支结构，输入一个数字(1～7)，用中文显示对应的星期(星期一～星期日)。运行效果如图 3-9 所示。

(a) 显示星期　(b) 错误提示

图 3-9　实验 3-5 运行效果

实验 3-6 党费计算器

实验要求：分别使用 if 语句和 switch 语句实现多分支结构，计算有固定工资收入的党员每月所交纳的党费。月工资收入 400 元及以下者，交纳月工资总额的 0.5%；月工资收入 401 元到 600 元者，交纳月工资总额的 1%；月工资收入在 601 元到 800 元者，交纳月工资总额的 1.5%；月工资收入在 801 元到 1500 元(税后)者，交纳月工资收入的 2%；月工资收入在 1500 元以上(税后)者，交纳月工资收入的 3%。即：

$$\text{党费 } f=\begin{cases}0.5\%\times\text{salary} & \text{salary}\leqslant 400\\ 1\%\times\text{salary} & 401\leqslant\text{salary}\leqslant 600\\ 1.5\%\times\text{salary} & 601\leqslant\text{salary}\leqslant 800\\ 2\%\times\text{salary} & 801\leqslant\text{salary}\leqslant 1500\\ 3\%\times\text{salary} & \text{salary}>1500\end{cases}$$

运行效果如图 3-10 所示。

图 3-10　实验 3-6 运行效果

操作提示：

为了使用 switch 语句，首先使用如下语句，将党费 f 的大范围区间数值转换为小范围区间数值(即 switch 语句中的控制表达式 c)：

```
if (salary > 1500) c = 15;
else c = (salary - 1)/100;
```

则党费 f 的计算公式变换为：

$$\text{党费 } f=\begin{cases}0.5\%\times\text{salary} & c=0\sim 3\\ 1\%\times\text{salary} & c=4\sim 5\\ 1.5\%\times\text{salary} & c=6\sim 7\\ 2\%\times\text{salary} & c=8\sim 14\\ 3\%\times\text{salary} & c=15\end{cases}$$

实验 3-7 袖珍计算器

实验要求：实现模拟袖珍计算器，要求输入两个操作数和一个操作符（+、-、*、/、%），根据操作符输出运算结果。运行效果如图 3-11 所示。

(a) 正常计算　　(b) 异常提示

图 3-11　实验 3-7 运行效果

操作提示：特别注意“/”和“%”运算符的零除异常问题。

实验 3-8 判断三角形

实验要求：输入三角形的三条边 a、b、c，判断此三条边是否可以构成三角形。若能，进一步判断三角形的性质：等边、等腰、直角或其他三角形。运行效果如图 3-12 所示。

(a) 不能构成三角形　(b) 能构成等边三角形　(c) 能构成等腰三角形　(d) 能构成直角三角形

图 3-12　实验 3-8 运行效果

操作提示：各类三角形的判断准则如表 3-5 所示。

表 3-5　各类三角形的判断准则

形　　状	满 足 条 件
三角形	任意两边之和大于第三边
等边三角形	三边均相等的三角形
等腰三角形	只有两边相等的三角形
直角三角形	勾股定理：斜边^2＝直角边 1^2＋直角边 2^2