第3章 C#程序的流程控制

总体要求

- 理解分支的概念，掌握 if 语句和 switch 语句的使用方法。
- 掌握条件运算符和条件表达式的使用方法。
- 理解循环的概念，掌握 while、do…while、for、foreach 语句的使用方法。
- 理解分支嵌套、循环嵌套的概念，了解相关应用。
- 掌握 continue 和 break 语句的使用方法。

相关知识点

- VS 2012 中创建项目、编辑程序、生成和调试应用程序的方法。
- 变量的声明和使用。
- 关系运算符和关系表达式。
- 逻辑运算符和逻辑表达式。
- 一维数组和字符串的使用方法。

学习重点

- if 语句和 switch 语句的使用方法。
- while、do…while、for 和 foreach 语句的使用方法。

学习难点

- 分支结构中条件的分析。
- 循环的条件、循环的操作的分析。
- 分支嵌套和循环嵌套。

一段 C＃程序由若干条 C＃语句按先后顺序排列而成。语句的排列顺序体现了程序的执行流程。通常，程序段按语句的先后顺序执行，如果需要改变执行流程，必须使用分支或循环语句。本章将详细介绍有关分支和循环的概念及其实现方法。

3.1 C＃程序的分支语句

程序的基本结构有三种：顺序结构、分支结构和循环结构。顺序结构一般为简单的程序，执行程序时按语句的书写顺序依次执行，但大量实际问题需要根据条件判断以改变程序执行顺序或重复执行某段程序，前者称为分支结构，后者称为循环结构。本节将介绍 C＃的两个分支语句：if 语句和 switch 语句。

3.1.1　if 语句

1. if 语句的一般形式

if 语句也称为条件语句或选择语句，用于实现程序的分支结构，根据条件是否成立来控制执行不同的程序段，完成相应的功能。

if 语句的一般形式如下：

```
if (表达式)
{
    语句块 1
}
else
{
    语句块 2
}
```

其中，表达式必须是布尔型的，通常由关系型表达式或逻辑表达式组成。

if 语句的逻辑意义为：如果表达式的值为 true，则选择执行“语句块 1”，否则选择执行“语句块 2”，如图 3-1 所示。

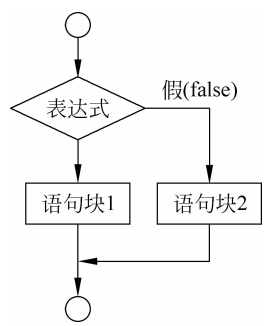

图 3-1　if 语句

“if…else…”的结构通常称为双分支结构。实际编程时，可省略 else 子句，构成单分支结构。当“语句块 1”或“语句块 2”只有一条语句时，可以省略花括号{}，还可以在同一行书写。

例如，设 x 为 int 型变量，下面的语句就是典型的单分支结构：

```
if(x % 2 == 0) Console.Write("x 为偶数");
```

2. 条件运算符和条件表达式

在 C＃中，如果双分支结构比较简单，可使用条件表达式来替代 if 语句。条件表达式的一般格式如下：

```
(表达式 1)?表达式 2:表达式 3
```

其中，条件运算符“?:”是 C＃语言中仅有的一个三目运算符。

条件表达式完成的运算是：

(1) 如果关系表达式 1 的值为真(true)，那么整个条件表达式的值为表达式 2 的值；

(2) 否则，为表达式 3 的值。

例如，设 x，y 为 int 型变量，则下面语句将 max 赋值为 x 和 y 的最大值：

```
int max = (x > y)?x:y;
```

该语句相当于：

```
int max;
if(x > y)
```

```
    max = x;
else
    max = y;
```

可见,使用条件表达式来构造一些逻辑比较简单的双分支结构,要比 if 语句更加简练。

【实例 3-1】 创建一个 Windows 应用程序,输入一个整数,判断该数是偶数还是奇数,并显示判断结果,运行效果如图 3-2 所示。

图 3-2 运行效果

(1) 首先在 Windows 窗体中添加两个 Label、一个 TextBox 和一个 Button 控件。各控件的主要属性设置见表 3-1。

表 3-1 需要修改的属性项

控 件	属 性	属 性 设 置
Label1	Text	请输入一个整数:
Label2	Name	lblShow
TextBox1	Name	txtNum
Button1	Name	btnOk
	Text	确定

(2) 然后编写“确定”按钮的 Click 事件方法,主要代码如下。

```
using System;
using System.Windows.Forms;

public partial class Test3_1 : Form
{
    private void btnOk_Click(object sender, EventArgs e)
    {
        //提取用户输入并转换为整数
        int num = Convert.ToInt32(txtNum.Text);
        if (num % 2 == 0)
            lblShow.Text = num + "是偶数!";
        else
            lblShow.Text = num + "是奇数!";
    }
}
```

3.1.2 多分支 if…else if 语句

在一个比较复杂的判断逻辑中,条件可能不止一个,这时可以使用多分支的 if…else if 语句。其语法如下:

```
if(表达式 1)          {语句 1;}
else if(表达式 2)     {语句 2;}
else if(表达式 3)     {语句 3;}
…
else if(表达式 n)     {语句 n;}
else                  {语句 n + 1;}
```

该语句的功能是：首先计算表达式 1，如果其结果为真(true)，则执行语句 1；否则依次往下计算各表达式的值，直到某个表达式的值为真(true)，并且执行相应的语句。如果所有表达式的值都为假，则执行最后的 else 子句后的语句 n＋1。整个 if 语句的流程图如图 3-3 所示。

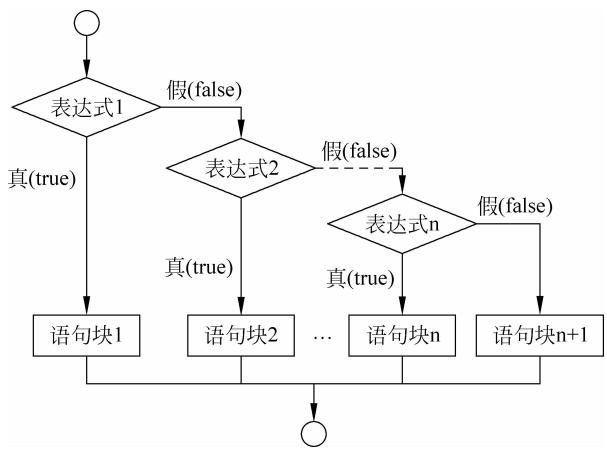

图 3-3　多分支结构

例如，设 x，y 为 int 型变量，则下面语句求出 x 和 y 的关系(大于、小于或等于)。

```
string result = "";
if(x > y)
    result = "x 比 y 大";
else if (x < y)
    result = "x 比 y 小";
else
    result = "x 和 y 相等";
```

注意事项：

(1) 整条 if 语句中只有一个分支能被执行。也就是说，当执行完某个分支后，整条 if 语句也就执行完毕了。

(2) else if 子句不能单独使用。

(3) 最后的 else 子句可省略，表示以上条件都不满足时，什么都不需要做。

【实例 3-2】 创建一个 Windows 应用程序，实现一个人的体型判断。医学上根据身高和体重，可以计算出“体指数”，从而实现对人肥胖程度的划分：

$$\text{体指数 } t = \text{体重 } w / (\text{身高 } h)^2$$

其中，w 单位是 kg，h 单位是 m，并且有如下判断依据。

(1) 当 t＜18 时，为偏瘦；

(2) 当 18≤t<25 时,为标准;

(3) 当 25≤t<27 时,为偏胖;

(4) 当 t≥27 时,为肥胖。运行效果如图 3-4 所示。

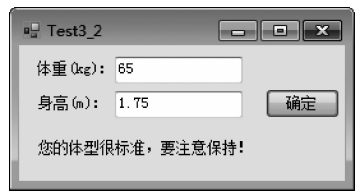

图 3-4 运行效果

(1) 首先在 Windows 窗体中添加三个 Label、两个 TextBox 和一个 Button 控件。各控件的主要属性设置见表 3-2。

表 3-2 需要修改的属性项

控 件	属 性	属 性 设 置
Label1	Text	体重(kg):
Label2	Text	身高(m):
Label3	Name	lblShow
TextBox1	Name	txtWeight
TextBox2	Name	txtHeight
Button1	Name	btnOk
	Text	确定

(2) 然后编写“确定”按钮的 Click 事件方法,主要代码如下。

```
using System;
using System.Windows.Forms;
public partial class Test3_2 : Form
{
    private void btnOk_Click(object sender, EventArgs e)
    {
        double h, w, t;
        w = Convert.ToDouble(txtWeight.Text);      //提取用户输入并转换为 double
        h = Convert.ToDouble(txtHeight.Text);
        t = w / (h * h);
        if (t < 18)
                lblShow.Text = "您的体型偏瘦,要注意营养!";
        else if ( t < 25)
                lblShow.Text = "您的体型很标准,要注意保持!";
        else if (t < 27)
                lblShow.Text = "您的体型偏胖,要注意多运动!";
        else
                lblShow.Text = "您的体型太胖了,要注意锻炼身体!"; }
}
```

3.1.3　switch 语句

当判断的条件较多，不止一两个分支时，也可使用 switch 语句。switch 语句主要用于实现多分支结构，其语法更简洁，能处理复杂的条件判断。

switch 语句的一般格式如下：

```
switch(表达式)

{
    case 常量 1:
        语句块 1;
        break;
    case 常量 2:
        语句块 2;
        break;
    …
    case 常量 n:
        语句块 n;
        break;
    default: 语句块 n + 1;
}
```

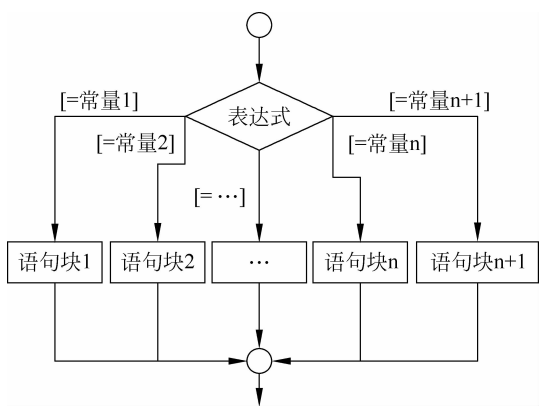

图 3-5　switch 语句

其中，switch 中的表达式通常是整型、字符型或字符串表达式，不能是关系表达式或逻辑表达式。case 后的常量不允许相同，其类型必须与表达式的值类型一致。

switch 语句的执行过程为：首先计算 switch 语句中表达式的值，再依次与每一个 case 后的常量比较，当表达式的值与某个常量相等时，则执行该 case 后的语句块，在执行 break 语句之后跳出 switch 结构，继续执行 switch 之后的语句，如图 3-5 所示。如果所有常量都不等于 switch 中表达式的值，则执行 default 之后的语句块，其中 default 子句可以省略，当表达式的值与 case 后的常量值都不相同时，则退出 switch 语句，执行该语句后面的语句。

可见，switch 语句中的 case 只是用来寻找分支的入口。程序在执行时一旦锁定某个分支，就执行该分支中的语句块，直到遇到 break 语句或到达 switch 结构的末尾为止。

C#不支持从一个 case 显式贯穿到另一个 case,因此在每一个 case 块的后面都必须有一个 break 语句,default 块的后面也必须有 break 语句,但当 case 语句中没有代码时例外,这时可以省略 break 语句,当表达式的值和一个 case 后的常量相同时,将直接顺序进入下一个 case 语句。

例如,已知整型量 a,b(b≠0),设 x 为实型量,计算分段函数:

$$y=\begin{cases} a+b\times x & 0.5\leqslant x<1.5 \\ a-b\times x & 0.5\leqslant x<2.5 \\ a\times b\times x & 2.5\leqslant x<3.5 \\ a/b\times x & 3.5\leqslant x<4.5 \end{cases}$$

使用 switch 语句求函数值的代码如下。

```
switch ((int)(x + 0.5))    //注意 switch 中的表达式只能是整型、字符型或字符串表达式
{
    case 1: y = a + b * x;break;
    case 2: y = a - b * x;break;
    case 3: y = a * b * x;break;
    case 4: y = a/(b * x);break;
    default: Console.WriteLine("x 值无效!");
}
```

【实例 3-3】 创建一个 Windows 应用程序,将考试的百分制成绩转换为等级:“优秀”、“良”、“中”、“及格”、“不及格”,其中,成绩大于等于 90 分的为优秀,80～89 分的为良,70～79 分的为中,60～69 分的为及格,60 分以下的为不及格,运行效果如图 3-6 所示。

图 3-6 运行效果

(1) 首先在 Windows 窗体中添加两个 Label、一个 TextBox 和一个 Button 控件。各控件的主要属性设置见表 3-3。

表 3-3 需要修改的属性项

控 件	属 性	属 性 设 置
Label1	Text	成绩:
Label2	Name	lblShow
TextBox1	Name	txtScore
Button1	Name	btnOk
	Text	确定

(2) 然后编写“确定”按钮的 Click 事件方法,主要代码如下。

```
using System;
using System.Windows.Forms;
public partial class Test3_3 : Form
```

```
{
    private void btnOk_Click(object sender, EventArgs e)
    {
        double score = Convert.ToDouble(txtScore.Text);
                                                    //提取用户输入的成绩并转换为浮点数
        switch ((int)score / 10)            //取出成绩的百位和十位,根据百位和十位确定等级
        {
                case 10:
                case 9:
                    lblShow.Text = "您的成绩等级为: 优";
                    break;
                case 8:
                    lblShow.Text = "您的成绩等级为: 良";
                    break;
                case 7:
                    lblShow.Text = "您的成绩等级为: 中";
                    break;
                case 6:
                    lblShow.Text = "您的成绩等级为: 及格";
                    break;
                default:
                    lblShow.Text = "您的成绩等级为: 不及格";
                    break;
            }
        }
    }
}
```

【注意】 “case 10：”中的常量 10 后因为没有语句，所以省略 break，当表达式“(int) score / 10”的值为 10 时，将贯穿到下一个 case 子句“case 9：”，执行语句“lblShow.Text = "您的成绩等级为：优";”。

上例也可以用多分支的 if…else if 语句来实现。如：

```
if (score >= 90)
    lblShow.Text = "您的成绩等级为: 优";
else if(score>=80)
    lblShow.Text = "您的成绩等级为: 良";
else if(score>=70)
    lblShow.Text = "您的成绩等级为: 中";
else if (score >= 60)
    lblShow.Text = "您的成绩等级为: 及格";
else
    lblShow.Text = "您的成绩等级为: 不及格";
```

可以看到，switch 语句和 if…else if 语句有异曲同工的效果，但它们也有不同。

(1) if…else if 后是关系、逻辑表达式，而 case 后是常量表达式；

(2) if…else if 更适合于条件是一个范围判定的情况，而 switch 只适合条件是相等判定的情况；

(3) if…else if 满足一个表达式后执行语句并退出，而 switch 满足一个表达式后执行语句，必须用 break 才能退出；

(4) if…else if 后的语句超过一句要用{},switch 中的 case 后,不管有多少条语句都不需要{}。

3.1.4 分支语句的嵌套

无论是 if 语句,还是 switch 语句,其中的语句可以是任何合法的 C# 语句,包括 if 语句或 switch 语句。如果 if 语句或 switch 语句中又包含 if 或 switch 语句,则称之为嵌套的分支语句。其中,嵌套的 if 语句也可以用来构建多分支结构的程序,以替代 switch 语句。

对于嵌套的 if 语句,从上到下,else 子句只与最近的尚未配对的 if 配对。为方便阅读和理解 if 和 else 的配对关系,要注意采用缩进格式书写代码或添加花括号{}。

例如,

```
if (x % 2 == 0) //①号 if
    if (x % 3 == 0) //②号 if
        Console.WriteLine("x 是能被 6 整除的偶数");
    else //①号 else
        Console.WriteLine("x 是不能被 6 整除的偶数");
else //②号 else
    Console.WriteLine("x 是一个奇数");
```

其中,①号 else 子句与最近的②号 if 配对,而②号 else 只能与①号 if 配对。

【实例 3-4】 创建一个 Windows 应用程序,使用嵌套的分支语句来判断用户输入的字符类型,运行效果如图 3-7 所示。

图 3-7 运行效果

(1) 首先根据表 3-4 在 Windows 窗体中添加窗体控件。

表 3-4 需要添加的控件及其属性设置

控 件	属 性	属性设置
Label1	Text	请输入一个字符:
Label2	Name	lblShow
TextBox1	Name	txtChar
Button1	Name	btnOk
	Text	确定

（2）然后在源代码视图中编辑如下代码。

```
using System;
using System.Windows.Forms;
public partial class Test3_4 : Form
{
    private void btnOk_Click(object sender, EventArgs e)
    {
        char c = Convert.ToChar(txtChar.Text);   //字符串转换为字符型
        if (Char.IsLetter(c))                    //判定指定的字符是否是一个字母
        {
            if (Char.IsLower(c))                 //判定指定的字符是否是一个小写字母
            {
                lblShow.Text = "它是一个小写字母.";
            }
            else if (Char.IsUpper(c))            //判定指定的字符是否是一个大写字母
            {
                lblShow.Text = "它是大写字母.";
            }
            else
            {
                lblShow.Text = "它是中文字符.";
            }
        }
        else if (char.IsNumber(c))               //判定指定的字符是否是一个数字
        {
            lblShow.Text = "它是数字.";
        }
        else
        {
            lblShow.Text = "它不是语言文字,也不是数字.";
        }
    }
}
```

该程序首先要求从键盘输入一个字符，然后对其进行判断，如果所输入的字符是文字字符，则进一步判断它是否为小写字母、大写字母或中文字符。如果不是文字字符，则进一步判断它是否为数字字符。注意，该程序的 if 语句最多为三重嵌套，为了理解嵌套关系，因此同时使用花括号和缩进格式来书写代码。

3.2　C＃程序的循环语句

循环结构是程序设计的基本结构之一。其特点是：在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。

C＃语言提供了多种循环语句，可以组成各种不同形式的循环结构，包括：while 语句、do…while、for 语句和 foreach 语句。本节将分别作介绍。

3.2.1 while 语句

while 语句表达的逻辑含义是：当逻辑条件成立时，重复执行某些语句，直到条件不成立时终止，从而不再循环。因此在循环次数不固定时 while 语句相当有用。while 语句的一般形式为：

```
while(表达式)
{
    语句块;
}
```

其中，表达式必须是布尔型表达式，用来检测循环条件是否成立，语句块为循环体。

while 语句执行过程如图 3-8 所示：首先计算表达式，当表达式的值为 true 时，执行一次循环体中的语句，重复上述操作到表达式的值为 false 时退出循环。如果表达式的值在开始时就为 false，那么不执行循环体语句直接退出循环。因此，while 语句的特点是：先判断表达式，后执行语句。

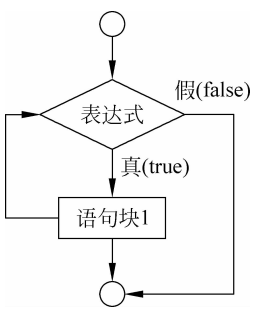

图 3-8 while 语句

while 语句在实际应用中，应该按照这样的思路进行设计：为了保证循环能正常进行，首先应在 while 语句之前增加一个控制循环的变量，并为其赋初值(当然该初始值应该符合循环的条件，即代入 while 语句中的表达式后，表达式的值为 true)；然后，在循环体中增加一条改变该变量值的语句，循环体每重复执行一次，其值就增加或减少一次。经过若干次循环后，其值将不符合循环条件，此时循环终止。

【实例 3-5】 求 $\sum_{n=1}^{100} n$，即 1+2+3+…+100。

【分析】 该题的求解实际是一个逐步累加的过程，就是做如下操作。

```
首先令 sum = 0;
sum = sum + 1;
sum = sum + 2;
sum = sum + 3;
…
sum = sum + 100;
```

最后的 sum 即是所求的累加和。从上面的操作中可以看到有相同的操作存在：增加、赋值…，直到所用的数用完。而循环结构正好适用于这样的重复过程。据此，可以制定这样的算法：

(1) 令 sum = 0，i = 1；

(2) 如果 i > 100，转到第(6)步；

(3) 否则，sum = sum+i；

(4) i = i + 1；

(5) 重复第(2)步；

(6) 输出 sum 变量的值,结束。

根据上述算法写成的程序如下所示。

```
using System;
using System.Windows.Forms;
public partial class Test3_5 : Form
{
    private void Test3_5_Load(object sender, EventArgs e)
    {
        int i, sum;
        i = 1;                                          //为循环变量赋初值
        sum = 0;
        while (i <= 100)                                //循环条件
        {                                               //循环体
            sum = sum + i;
            i++;                                        //改变循环变量的值
        }
        lblShow.Text = "1 到 100 的自然数之和是" + sum;   //显示计算结果
    }
}
```

该程序代码主要包含在窗体的 Load 事件方法 Test3_5_Load 中。程序先计算 1～100 的自然数之和,再使用 Label 控件显示计算结果,运行效果如图 3-9 所示。

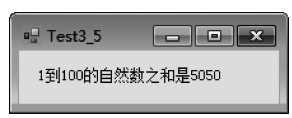

图 3-9　运行效果

3.2.2　do…while 语句

do…while 语句的特点是先执行循环体,然后判断循环条件是否成立,其一般形式为:

```
do
{
    语句块;
}
while (表达式);
```

其中,语句块为循环体,表达式必须是布尔型表达式,用来检测循环条件是否成立。

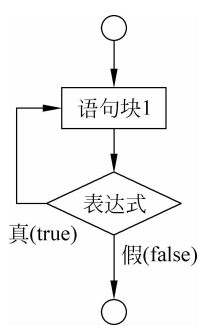

图 3-10　do…while 语句

do…while 语句执行过程如图 3-10 所示:首先执行一次循环体,然后再计算表达式,如果表达式的值为 true,则再执行一次循环体,重复上述操作,直到表达式的值为 false 时退出循环。如果条件在开始时就为 false,那么执行一次循环体语句后退出循环。例如:

```
i = 1;                                          //为循环变量赋初值
sum = 0;
```

```
do
{
    sum = sum + i;                                  //循环体
    i++;                                            //改变循环变量的值
} while (i <= 100);                                 //循环条件
lblShow.Text = "1～100 的自然数之和是" + sum; //显示计算结果
```

使用 do…while 语句需要注意以下几点。

(1) while 是先执行后判断,循环至少做一次。

(2) 一般情况下,while 和 do…while 均可以替换。但当第一次循环条件不满足的情况下,两者是不等的,不能互换。例如:

```
i = 101;
sum = 0;
while (i <= 100)
{
    sum = sum + i;
    i++;
}
```

循环体一次也不执行,sum 为 0。而

```
i = 101;
sum = 0;
do
{
    sum = sum + i;
    i++;
} while (i <= 100) ;
```

循环体将执行一次,sum 为 101。

可见,while 语句与 do…while 语句的区别在于:前者循环体执行的次数可能是 0 次,而后者循环体执行的次数至少是 1 次。

【实例 3-6】 创建一个 Windows 应用程序,统计从键盘输入的一行字符中英文字母的个数。

【分析】 很明显,从一行字符中数出英文字母的个数是一个循环判断的过程。为此,可以设置一个循环控制变量 i 和记录器变量 n。每一次循环先提取文本框中的第 i 个字符(与此同时变量 i 加 1),再判断该字符是否在 A～Z、a～z 之间,如果是就让 n＋＋。当 i 的值等于文本框的所输入的字符串长度时,循环结束。

(1) 首先根据表 3-5 在 Windows 窗体中添加窗体控件。

表 3-5 需要添加的控件及其属性设置

控　　件	属　　性	属 性 设 置
Label1	Text	请输入一行字符:
Label2	Name	lblShow
TextBox1	Name	txtSource
Button1	Name	btnOk
	Text	确定

(2) 然后在源代码视图中编辑如下代码。

```
public partial class Test3_6 : Form
{
    private void btnOk_Click(object sender, EventArgs e)
    {
        int n = 0, i = 0;
        do
        {
            char c = txtSource.Text[i++];
            if (c >= 'A' && c <= 'Z' || c >= 'a' && c <= 'z')
                n++;
        } while (i!= txtSource.Text.Length);
        lblShow.Text = String.Format("在该字符中,英文字母共: {0}个.", n);
    }
}
```

该程序运行效果如图 3-11 所示。

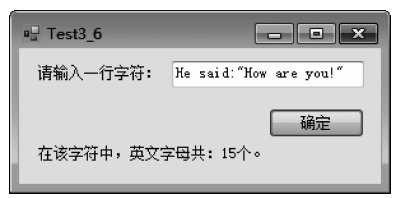

图 3-11　运行效果

其中 String.Format 方法的作用是将相应的变量采用指定格式字符串输出,其中的{0}将用后面的变量值替换,其中的 0 表示输出变量的序号,序号从 0 开始,如有多个变量,依次为{0},{1},{2},…

3.2.3　for 语句

for 语句与 while 语句、do…while 语句一样,可以循环重复执行一个语句或语句块,直到指定的表达式计算为 false 值。for 语句的一般形式为:

```
for(表达式 1; 表达式 2; 表达式 3)
{
    语句块;
}
```

其中,表达式 1 为赋值表达式,通常用于初始化循环控制变量;表达式 2 为布尔型的表达式,用来检测循环条件是否成立;表达式 3 为赋值表达式,用来更新循环控制变量的值,以保证循环能正常终止。

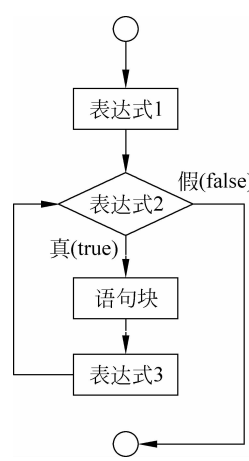

图 3-12　for 语句的执行过程

for 语句的执行过程(如图 3-12 所示)详细如下。

(1) 首先计算表达式 1,为循环控制变量赋初值;

(2) 然后计算表达式 2,检查循环控制条件,若表达式 2 的值为 true,则执行一次循环体语句,若为 false,终止循环;

(3) 执行完一次循环体语句后,计算表达式 3,对控制变量进行增量或减量操作,再重复第(2)步操作。

C#允许省略 for 语句中的三个表达式,但注意两个分号不要省略,同时要保证在程序中有起同样作用的语句。省略后的一般形式如下:

```
表达式 1;
for(;;)
{
    if(表达式 2 == false)
    {
        break;
    }
    语句;
    表达式 3;
}
```

【实例 3-7】 一个百万富翁遇到一个陌生人,陌生人找他谈一个换钱的计划,该项计划如下:我每天给你十万元,而你第一天只需给我一分钱,第二天我仍给你十万元,你给我二分钱,第三天我仍给你十万元,你给我四分钱,…,你每天给我的钱是前一天的两倍,直到满一个月(30 天),百万富翁很高兴,欣然接受了这个契约。请编写一个程序计算这一个月中陌生人给了百万富翁多少钱,百万富翁给陌生人多少钱。

【分析】 设第 i 天百万富翁给陌生人的钱为 t_i,则 $t_1=0.01$ 元,由题意可得,$t_i=t_{i-1}\times 2$。设第 i 天后百万富翁给陌生人的钱总数为 $s1_i$,则 $s1_1=t_1=0.01$,$s1_i=s1_{i-1}+t_i$。设第 i 天后陌生人给百万富翁的钱总数为 $s2_i$,则 $s2_1=100\,000$,$s2_i=s2_{i-1}+100\,000$。显然,这是一个循环过程。

```
using System;
using System.Windows.Forms;
public partial class Test3_7 : Form
{
    private void Test3_7_Load(object sender, EventArgs e)
    {
        int i;
        double t,s1, s2;
        s1 = t = 0.01;                          //百万富翁第一天给陌生人的钱为 1 分
        s2 = 100 000;                           //陌生人第一天给百万富翁的钱为十万元
        for (i = 2; i <= 30; i++)
        {
            t = t * 2;                          //百万富翁第 i 天给陌生人的钱
            s1 = s1 + t;                        //百万富翁第 i 天后共给陌生人的钱
            s2 = s2 + 100 000;                  //陌生人第 i 天后共给百万富翁的钱
        }
        lblShow.Text = String.Format("百万富翁给陌生人:{0:N2}元.\n"
                                    +"陌生人给百万富翁:{1:N2}元.",s1,s2);
        /*说明: 在格式字符"{1:N2}"中,"1"表示索引,
```

```
            * "N2"表示输出的数字带两位小数,整数每三位用逗号间隔
            * 如果参数不足两位小数,则自动补充显示 0
            */
        }
    }
```

该程序的运行结果如图 3-13 所示。

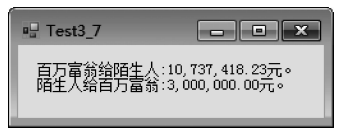

图 3-13　运行结果

3.2.4　foreach 语句

C# 的 foreach 语句提供了一种简单明了的方法来循环访问数组或集合的元素,又称迭代器。foreach 语句的一般形式如下:

```
foreach(类型 循环变量 in 表达式)
{
    语句块;
}
```

其中,表达式一般是一个数组名或集合名,循环变量的类型必须与表达式的数据类型一致。

foreach 语言的执行过程如下。

(1) 自动指向数组或集合中的第一个元素;

(2) 判断该元素是否存在,如果不存在结束循环;

(3) 把该元素的值赋给循环变量;

(4) 执行循环体语句块;

(5) 自动指向下一个元素,之后从第(2)步开始重复执行。

【实例 3-8】 创建一个 Windows 程序,实现如下功能。

(1) 输入联系人姓名和电话号码并保存到结构体数组中;

(2) 使用 foreach 语句迭代查询指定联系人的电话号码。

该程序运行效果如图 3-14 所示。

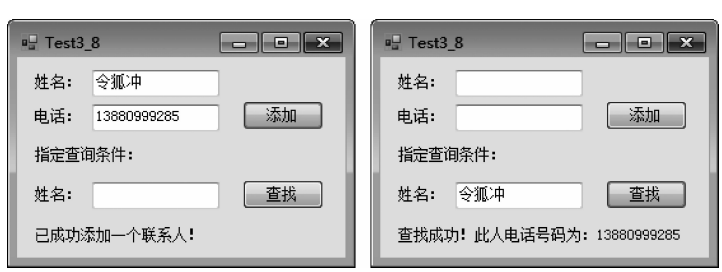

图 3-14　运行效果

(1) 首先根据表 3-6 在 Windows 窗体中添加窗体控件。

表 3-6 需要添加的控件及其属性设置

控 件	属 性	属性设置	控 件	属 性	属性设置
Label1	Text	姓名:	TextBox2	Name	txtTel
Label2	Text	电话:	TextBox3	Name	txtSearch
Label3	Text	指定查询条件:	Button1	Name	btnAdd
Label4	Text	姓名:		Text	添加
Label5	Name	lblShow	Button2	Name	btnSearch
TextBox1	Name	txtName		Text	查找

(2) 然后在源代码视图中编辑如下代码。

```
using System;
using System.Windows.Forms;
public partial class Test3_8 : Form
{
    struct Contacter                              //定义结构体
    {
        public string name;
        public string telphone;
    }
    Contacter[] persons = new Contacter[10];  //定义结构体数组,用于保存联系人信息
    int i = 0;                                    //用来记录已添加的联系人的个数
    private void btnAdd_Click(object sender, EventArgs e)
    {                                             //获得用户输入并保存到第 i 个数组元素中
        persons[i].name = txtName.Text;
        persons[i].telphone = txtTel.Text;
        i++;
        lblShow.Text = "已成功添加一个联系人!";
    }
    private void btnSearch_Click(object sender, EventArgs e)
    {
        bool isSearched = false;                  //定义标志变量,用于记录查找是否成功
        foreach (Contacter c in persons)          //迭代查找指定联系人
        {
            if (c.name == txtSearch.Text.Trim())
            {
                isSearched = true;                //修改标志变量,表示查找成功
                lblShow.Text = "查找成功!此人电话号码为: " + c.telephone;
            }
        }
        if (!isSearched)
            lblShow.Text = "查无此人!";
    }
}
```

【分析】 程序首先声明了一个名称为 Contacter 的结构体,该结构体包含两个成员 name 和 telephone,分别用来记录一个联系人的姓名和电话号码。在程序中,定义了一个结

构数组 persons 用于保存联系人信息，并用 i 记录当前数组的索引值。用户单击“添加”按钮，则将联系人信息添加到 persons[i]中，并显示“已成功添加一个联系人!”，当用户单击“查找”按钮，则使用 foreach 语句将 persons 中的每一个联系人取出来，判断该联系人的姓名和在查找文本框中输入的姓名是否一致，如果一致，则显示“查找成功! 此人电话号码为：……”。

在使用 foreach 时要注意以下几点。

(1) foreach 语句总是遍历整个数组。如果只需要遍历数组的特定部分(例如前半部分)，或者需要绕过特定元素(例如，只遍历索引为偶数的元素)，那么最好是使用 for 语句。

(2) foreach 语句总是从第一个元素遍历到最后一个元素。如果需要反向遍历，那么最好是使用 for 语句。

(3) 如果循环体需要知道元素索引，而不仅是元素值，那么必须使用 for 语句。

(4) 如果需要修改数组元素，那么必须使用 for 语句。这是因为 foreach 语句的循环变量是一个只读变量。例如，如果在本例的 foreach 的循环体中加上如下语句：

```
c. name = "乔峰";
```

则在编译时将出现如下错误：“c”是一个“foreach 迭代变量”，因此无法修改其成员。

3.2.5　循环语句的嵌套

在一个循环体内又包含另一个循环结构，称为循环嵌套。内层循环体中如果又包含新循环结构，则称之为多重循环嵌套。C＃没有严格规定多重循环的层数，但为了便于理解程序逻辑，建议循环嵌套不要超过三层。

C＃语言允许各种循环结构任意组合嵌套，一般说来，嵌套循环中涉及几个循环结构就称之为几重循环。下例示意了 for 和 while 嵌套形成的二重循环。

```
for(i = 1;i < 10;i++)
{
    while(j < 10)
    {
        Console.WriteLine("i = {0},j = {1}",i,j);
        j++;
    }
}
```

内循环　外循环

在使用循环嵌套时，有一些需要注意的地方。

(1) 在使用嵌套时，应使用复合语句(即多用花括号)以保证逻辑上的正确性。

(2) 内外层的循环变量名应不同，以避免造成混乱。

(3) 不允许循环交叉。即内循环必须完全包含于外循环内。

(4) 书写时最好养成右缩进的习惯，使得层次清晰，易于检查。

【实例 3-9】 创建一个 Windows 应用程序，打印如图 3-15 所示的九九乘法表。

【分析】 九九乘法表共 9 行，设行号为 i(i＝1,2,…,9)，设列号为 j(j＝1,2,…,i)，对于 i 来说，其值每增加 1，对应的 j 将周而复始地从 1 开始增加，直到等于行号 i 时结束。显然，如果用两个循环来分别产生行和列，那么产生行的循环必须包含产生列的循环，这是一个嵌

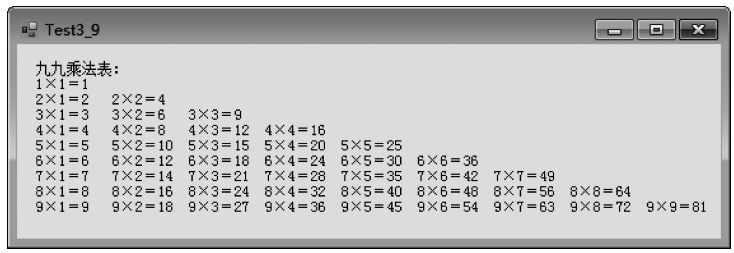

图 3-15 运行效果

套循环。当产生列的循环结束时,可使用“\n”实现换行显示。

主要源代码如下。

```
using System;
using System.Windows.Forms;

public partial class Test3_8 : Form
{
    private void Test3_8_Load(object sender, EventArgs e)
    {
        lblShow.Text = "九九乘法表: \n";
        for(int i = 1; i<=9;i++)
        {
            for (int j = 1; j <= i; j++)
            {
                lblShow.Text += String.Format("{0}×{1}={2,-2:D} ", i, j, i * j);
                /*说明: 在格式字符"{2,-2:D}"中,第一个"2"表示索引,
                        "-2: D"表示输出十进制数字,左对齐同时占两个字符位置,
                        如果参数不足两位,则自动补充显示空格
                 */
            }
            lblShow.Text += "\n";
        }
    }
}
```

3.3 跳转语句

前面讨论的循环语句,都是以某个布尔型表达式的结果作为循环条件,当表达式的值为false时,就结束循环。但有时希望在循环的中途直接控制流程转移。C#提供了两个跳转语句:break、continue,本节将详细介绍它们的使用方法。

3.3.1 break 语句

break 语句既可用于 switch 语句,也可用于循环语句。break 语句用于 switch 语句时,表示跳转出 switch 语句;用于循环语句时表示提前终止循环。在循环结构中,break 语句可与 if 语句配合使用,通常先用 if 语句判断条件是否成立,如果成立,则用 break 来终止循环,跳转出循环结构。

【实例 3-10】 创建一个 Windows 程序，输入一个整数，并判断该数是否是质数。

【分析】 质数是除了 1 和本身外没有其他因子的数，例如 3、17、41 等。根据定义，要确定一个数 m 是否为质数，就可以通过测试 m 有没有因子来确定。如果有，则不是质数；反之则是。可以让 m 一个个地去除以 2～$\sqrt{m}$之间的所有整数，只要其中一个能被整除，那么 m 肯定不是质数；如果所有的都不能被整除，则 m 一定是质数。算法如下：

(1) 给定数 m，令 n=⌊$\sqrt{m}$⌋(向下取整)；

(2) 令 i=2；

(3) 令 r = m % i；

(4) 如果 r = 0，则表明 m 不是质数，转到第(8)步；

(5) 否则，令 i++；

(6) 如果 i≤n，那么转向第(3)步；

(7) 否则，m 一定是质数；

(8) 结束。

操作过程如下。

(1) 首先根据表 3-7 在 Windows 窗体中添加窗体控件。

表 3-7　需要添加的控件及其属性设置

控　　件	属　　性	属 性 设 置
Label1	Text	整数：
Label2	Text	lblShow
TextBox1	Name	txtNum
Button1	Name	btnOk
	Text	判断

(2) 然后在源代码视图中编辑如下代码。

```
using System;
using System.Windows.Forms;
public partial class Test3_10 : Form
{
    private void btnOk_Click(object sender, EventArgs e)
    {
        int num = Convert.ToInt32(txtNum.Text);      //把输入的文本转换成对应的整数
        int n = (int)Math.Sqrt(num);                 //Math.Sqrt()方法求指定数字的平方根
        int i;
        for (i = 2; i <= n; i++)
        {
            if (num % i == 0)
                break;                               //不是质数,跳出循环体
        }
        if (i <= n)       //如果 i<=n,一定是在循环体内遇到 break 退出的,说明 num 不是质数
            lblShow.Text = num + "不是质数!";
        else
            lblShow.Text = num + "是质数!";
    }
}
```

该程序的运行效果如图 3-16 所示。

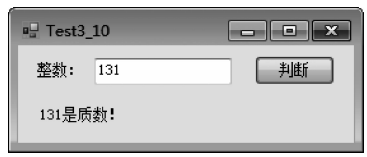

图 3-16 运行效果

3.3.2 continue 语句

continue 语句只能用于循环结构,与 break 语句不同的是,continue 语句不是用来终止并跳出循环结构的,而是忽略 continue 后面的语句,直接进入本循环结构的下一次循环操作。在 while 和 do…while 循环结构中,continue 立即转去检测循环控制表达式,以判定是否继续进行循环,在 for 语句中,则立即转向计算表达式 3,以改变循环控制变量,再判定表达式 2,以确定是否继续循环。图 3-17 展示了 break 和 continue 在 for 循环结构中的区别。

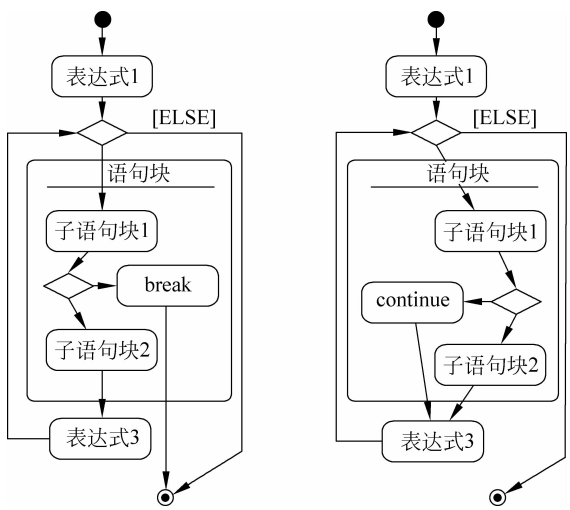

图 3-17 break 和 continue 在 for 语句中的区别

【实例 3-11】 创建一个 Windows 应用程序,过滤连续重复输入的字符。

(1) 首先根据表 3-8 在 Windows 窗体中添加窗体控件。

表 3-8 需要添加的控件及其属性设置

控 件	属 性	属 性 设 置
Label1	Text	字符串(相同字符将被过滤):
Label2	Text	lblShow
TextBox1	Name	txtSource
Button1	Name	btnOk
	Text	过滤

（2）然后在源代码视图中编辑如下代码。

```
using System;
using System.Windows.Forms;
public partial class Test3_11 : Form
{
    private void btnOk_Click(object sender, EventArgs e)
    {
        char ch_old, ch_new;
        ch_old = '';
        lblShow.Text = "过滤之后的结果如下：\n\n";
        for (int i = 1; i < txtSource.Text.Length; i++)
        {
            ch_new = (char)txtSource.Text[i];
            if (ch_new == ch_old) continue;            //前后两个字符相同,忽略后面的字符
            lblShow.Text + = ch_new.ToString();
            ch_old = ch_new;
        }
    }
}
```

程序使用 ch_new 获取输入的每一个字符，用 ch_old 记录该字符之前的字符，如果两者相等，则用 continue 结束本次循环，继续下一次的循环。如果不相同，则将字符 ch_new 添加到 lblShow.Text 中，同时，让 ch_old＝ch_new，继续下一次的循环。

该程序运行效果如图 3-18 所示。

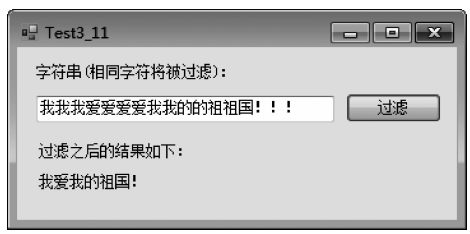

图 3-18　运行效果

习题

1. 简述 if…else…语句的逻辑意义。
2. 请列举 switch 语句的特点。
3. 在使用嵌套的 if 语句时，else 子句与 if 配对遵循什么原则？
4. 比较 while 语句和 do…while 语句的异同。
5. 请描述 for 语句的基本格式，并简述其执行流程。
6. 比较 for 语句和 foreach 语句的异同。
7. 指出以下循环体的执行次数：

```
for(int i = 1; i<=n; i++)
```

```
{
   for (int j = 1; j <= m; j++)
   {
      … // 循环体
   }
}
```

8. 比较 break 语句和 continue 语句的区别。

9. 设计一个 Windows 应用程序,实现如下功能:输入考试成绩,判断并显示优、良、中、及格或不及格的等级。

10. 有一函数:

$$y=\begin{cases}1-2x & (0\leqslant x<10)\\ x & (10\leqslant x<20)\\ 1+2x & (20\leqslant x<30)\end{cases}$$

设计一个 Windows 应用程序,输入 x,输出 y 值。

11. 设计一个 Windows 应用程序,显示所有水仙花数。所谓水仙花数是指一个三位数,其各位数字的立方和等于该数本身,例如,153 就是一个水仙花数,因为 $153=1^3+5^3+3^3$。

12. 设计一个 Windows 应用程序,计算以下分数序列前 20 项之和:

$$\frac{2}{1},\frac{3}{2},\frac{5}{3},\frac{8}{5},\frac{13}{8},\frac{22}{13},\cdots$$

13. 设计一个 Windows 应用程序,使用 for 语句输出杨辉三角的前 10 行,形式如下:

```
1
1  1
1  2  1
1  3  3   1
1  4  6   4   1
1  5  10  10  5  1
…
```

14. 设计一个 Windows 应用程序,将 1～1000 中能被 3 但不能被 5 整除的数输出。

15. 分析下列程序代码,请写出该程序运行时的输出结果。

```
using System;
public class Program
{
    static void Main(string[] args)
    {
        for (int i = 1; i < 20;i++)
        {
            if(i % 2 == 0 || i % 3 == 0)
                Console.WriteLine(i.ToString() + " ");
        }
    }
}
```

上机实验 3

一、实验目的

(1) 理解分支和循环的逻辑意义。
(2) 掌握 C# 的 if、switch 分支语句的使用方法。
(3) 掌握 C# 的 while、do…while、for、foreach 等循环语句的使用方法。

二、实验要求

(1) 熟悉 VS 2012 的基本操作方法；
(2) 认真阅读本章相关内容，尤其是案例；
(3) 实验前进行程序设计，完成源程序的编写任务；
(4) 反复操作，直到不需要参考教材、能熟练操作为止。

三、实验步骤

(1) 修改上机实验 2 的第 3 个实验任务，将输入的 n 个数字，通过 for 语句排序并输出。注意，不允许使用 Array.Sort( )方法排序。

(2) 设计一个 Windows 应用程序，实现如下功能。

① 输入学生姓名和考试成绩并保存到结构体数组中；

② 使用 foreach 语句求最高分并输出对应的姓名。

(3) 设计一个 Windows 应用程序，输入一行字符，检索是否存在重复的二字词汇(由两个字符组成的字符)，输出重复的次数，效果如图 3-19 所示。

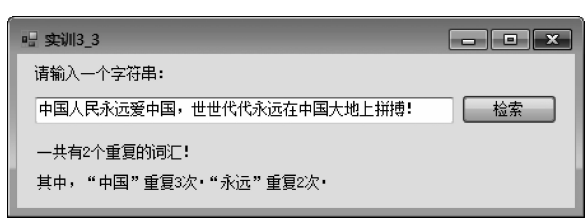

图 3-19　运行效果

核心代码如下：

```
private void btnSearch_Click(object sender, EventArgs e)
{
    int n = 0;                                          //记录出现重复的词汇的个数
    string[] words = new string[10];                    //保存出现重复的词汇
    int[] times = new int[10];                          //记录每一个重复的词汇的出现次数

    //寻找第 n 个出现重复的词汇
    for (int i = 0; i < txtSource.Text.Length - 2; i++)
```

```
    {
        bool isSame = false;                                    //记录是否发生重复
        string source = txtSource.Text.Substring(i, 2);         //提取二字源词
        int j = i + 2;
        while (j < txtSource.Text.Length - 2)
        {
            string target = txtSource.Text.Substring(j, 2); //提取二字目标词
            if (source == target)
            {
                times[n] ++;                                    //重复次数增加 1
                //如果是新出现的重复词汇,则保存
                if (Array.IndexOf(words, target) == -1)
                {
                    isSame = true;
                    words[n] = target;
                }
            }
            j++;
        }
        if (isSame) n++;                                        //出现重复的词汇的个数加 1
    }
    lblShow.Text = String.Format("一共有{0}个重复的词汇!\n\n其中,", n);
    for (int i = 0; i < 10; i++)
    {
        if(!String.IsNullOrEmpty(words[i]))
            lblShow.Text += String.Format("“{0}”重复{1}次·", words[i], times[i] + 1);
    }
}
```

四、实验总结

写出实验报告,报告内容包括:实验内容、任务分析、算法设计、源程序、实验体会等,并记录实验过程中的疑难点。