第 3 章　流程控制语句的应用

语句是程序中最小的程序指令，即程序完成一次完整正操的基本单位。在 C#中，可以使用多种类型的语句，每一种类型的语句又可以通过多个关键字实现。通过这些语句可以控制程序代码的逻辑，提高程序的灵活性，从而实现比较复杂的程序逻辑。

本章主要内容：

- 选择语句的应用
- 迭代语句的应用
- 跳转语句的应用

3.1　选择语句的应用

选择语句也叫作分支语句，选择语句根据某个条件是否成立来控制程序的执行流程。选择语句包括 if 语句和 switch 语句，除此之外，也可把第 2 章中提到的三元运算符作为分支技术的一种。下面就详细介绍这三种分支技术。

3.1.1　简单的 if 条件语句

C#中的 if 语句比较简单，其语法格式如下：

```
if(条件表达式)
{
    【代码块】
}
```

if 语句的执行过程如图 3-1 所示。

如果条件表达式的值为 true，执行代码块，否则跳过 if 语句继续向下执行其他程序代码。

3.1.2　if…else…条件语句

if…else…语法格式如下：

```
if(条件表达式)
{
    【代码块 a】
}
else
```

```
{
    【代码块 b】
}
```

if…else…语句的执行过程如图 3-2 所示。

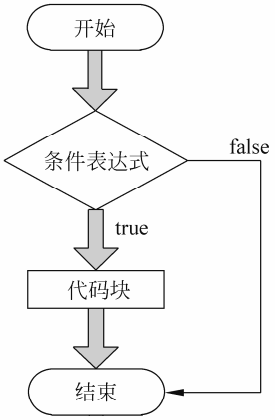

图 3-1　if 语句的执行过程

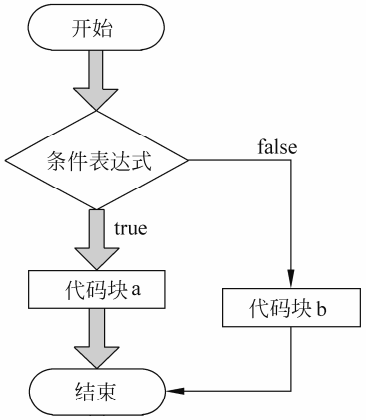

图 3-2　if…else…语句的执行过程

如果条件表达式的值为 true，那么执行代码块 a 中的代码，否则执行代码块 b 中的代码。

3.1.3　if…else if…else 多分支语句

if…else if…else 多分支语句的语法格式如下：

```
if(条件表达式 1)
{
    【代码块 a】
}
else if(条件表达式 2)
{
    【代码块 b】
}
else if(条件表达式 3)
{
    【代码块 c】
}
else
{
    【代码块 d】
}
```

if…else if…else 多分支语句的执行过程如图 3-3 所示。

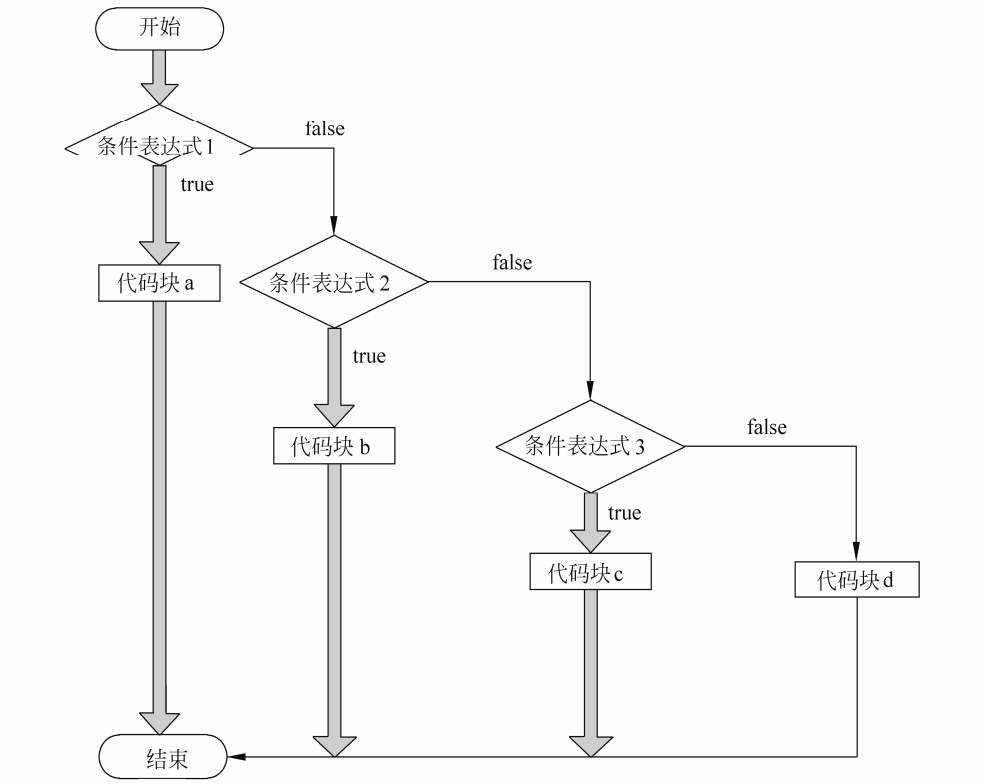

图 3-3　if…else if…else 多分支语句的执行过程

如果条件表达式 1 的值为 true，那么执行代码块 a 中的代码，否则判断条件表达式 2 的值是否为 true，如果为 true，则执行代码块 b 中的代码，否则继续判断条件表达式 3 的值是否为 true，如果为 true，则执行代码块 c 中的代码，否则执行代码块 d 中的代码。

3.1.4　嵌套 if 语句

嵌套 if 语句的语法格式如下：

```
if(条件表达式 1)
{
     if(条件表达式 2)
     {
         【代码块 a】
     }
     else
     {
         【代码块 b】
     }
}
else
{
    if(条件表达式 3)
    {
```

```
            【代码块 c】
        }
        else
        {
            【代码块 d】
        }
    }
```

嵌套 if 语句的执行过程如图 3-4 所示。

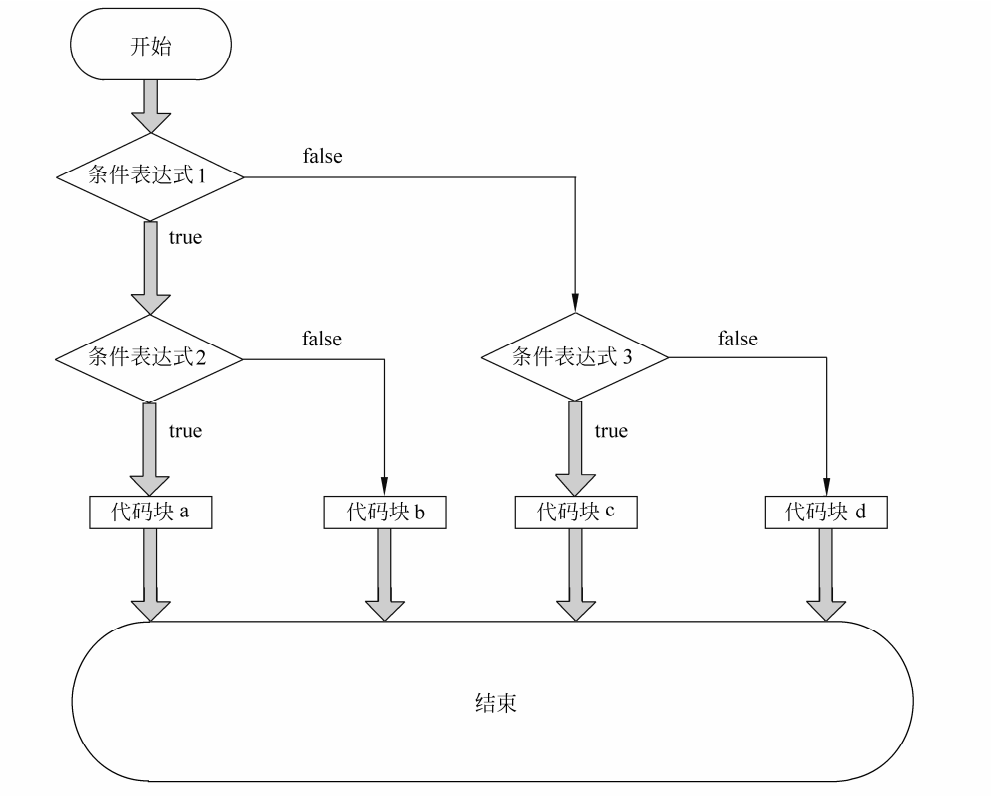

图 3-4　嵌套 if 语句的执行过程

如果条件表达式 1 的值为 true，接着判断条件表达式 2 的值，如果为 true，就执行代码块 a 中的代码，否则执行代码块 b 中的代码。如果条件表达式 1 的值为 false，接着判断条件表达式 3 的值是否为 true，如果为 true，则执行代码块 c 中的代码，否则执行代码块 d 中的代码。

3.1.5　switch 多分支语句

switch 语句与 if 语句非常类似，是通过将控制传递给其内部的一个 case 语句来处理多个选择的流程控制语句。C#中要求每个 case 和 default 语句中都必须有 break 语句，除非两个 case 中间没有其他语句，那么前一个 case 可以不包含 break。此外，判断的表达式或变量可以是 int、char 或 string 等类型。

switch 语句的基本格式如下：

```
switch(条件表达式)
{
    case 常量值 1:
        代码块 1;
        break;
    case 常量值 2:
        代码块 2;
        break;

    …
    case 常量值 N:
        代码块 N;
        break;
    default:
        缺省代码块;
        break;
}
```

switch 关键字后面的括号中是条件表达式，大括号中的代码是由若干个 case 子句所组成。条件表达式的值与每个常量值进行比较，如果有一个匹配，就执行为该匹配提供的代码块语句。如果没有匹配，就执行 default 部分中的代码。不论是否执行 default 最后都会执行 break 语句，使程序跳出 switch 语句。switch 语句可以包含任意数目的 case 子句，但是任何两个 case 语句都不能具有相同的值。一个 switch 语句中只能有一个 default 标签。

一个 case 语句处理完成后，不能再进入下一个 case 语句了，但有一种情况例外，代码如下。

```
switch(条件表达式)
{
    case 常量值 1:
    case 常量值 2:
        代码块;
        break;

    …
    case 常量值 N:
        代码块 N;
        break;
    default:
        缺省代码块;
        break;
}
```

把若干个 case 子句放在一起，在其后加一个代码块，实际上是一次检查多个条件，如果满足其中一个条件，就会执行代码块中的代码以及 break 语句。

3.1.6　三元运算符

三元运算符有三个操作数，其语法格式如下：

```
<test>?<resultIfTrue>:<resultIfFalse>
```

其中，第一个操作数<test>可得到一个 bool 值，如果这个值为 true，则结果为<resultIfTrue>，否则为<resultIfFalse>。

例如：

```
String resultStr=(age>=18)?"你已经成年！":"你尚未成年！";
```

此例判断 int 类型的变量 age 是否大于等于 18，如果是，则 resultStr 值为“你已经成年！”，否则为“你尚未成年！”。

3.2 迭代语句的应用

迭代语句又称为循环语句，使用迭代语句可以让程序多次执行相同的代码或代码块，这些代码或代码块称为循环体。对于任何一个循环体来说，都应该提供一个跳出循环的条件，不同的循环语句提供不同的条件。在 C#中，常用的迭代语句有 for 语句、while 语句、do…while 语句和 foreach 语句。下面就详细介绍这几种迭代语句的用法。

3.2.1 for 循环语句

for 循环可以执行指定的次数，并维护自己的计数器。其语法格式如下：

```
for(表达式 1;表达式 2;表达式 3)
{
    代码块
}
```

其中，表达式 1 由一个局部变量声明或由一个逗号分隔的多个表达式组成。其变量的作用域范围为从声明开始，直到代码块的结尾。

表达式 2 规定必须是一个布尔表达式。

表达式 3 必须包含一个用逗号分隔的表达式列表。

for 循环语句的执行过程如图 3-5 所示。

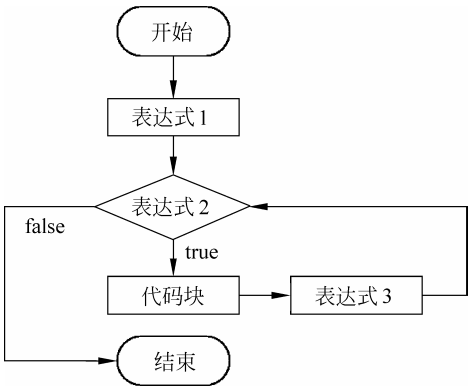

图 3-5　for 循环语句的执行过程

从图 3-5 中不难看出 for 循环的执行顺序如下。

（1）如果存在表达式 1，则先执行表达式 1。

（2）如果存在表达式 2，则计算表达式 2；如果不存在表达式 2，则程序转移到代码块执行，若执行到了代码块的结束点，则按顺序计算表达式 3，然后从上一步骤中表达式 2 的计算开始，执行下一次的循环。

下面通过一个具体例子来详细讲解下 for 循环。

例 3-1：使用 for 循环输出 0～9（ConsoleForLoop）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleForLoop
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine(i);
            }
            Console.ReadLine();
        }
    }
}
```

运行结果如图 3-6 所示。

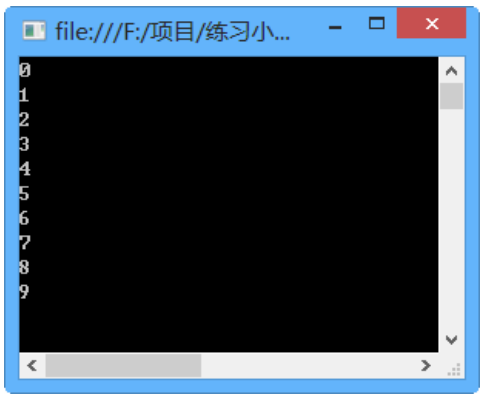

图 3-6　使用 for 循环输出 0～9

在此例中，先初始化 i 为 0，然后判断 i 的值是否小于 10，若小于则输出 i 的值并换行，接着 i 自增 1，然后开始下一次判断 i 的值是否小于 10，若小于则输出 i 的值并换行，接着 i 自增 1，重复以上步骤，直到 i 的值为 10 时，再判断 i 的值是否小于 10，此时循环条件

不成立，循环结束。

3.2.2 while 循环语句

while 语句执行一个语句或语句块，直到指定的表达式计算为 false。

while 循环的语法格式如下：

```
while (条件表达式)
{
    代码块
}
```

while 循环语句的执行过程如图 3-7 所示。

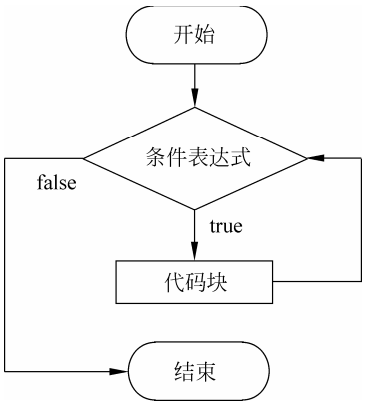

图 3-7 while 循环语句的执行过程

下面通过一个具体例子来详细讲解下 while 循环。

例 3-2：使用 while 循环输出 0～9（ConsoleWhileLoop）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleWhileLoop
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            while (i < 10)
            {
                Console.WriteLine(i);
                i++;
            }
            Console.ReadLine();
        }
    }
```

```
}
```

运行结果如图 3-8 所示。

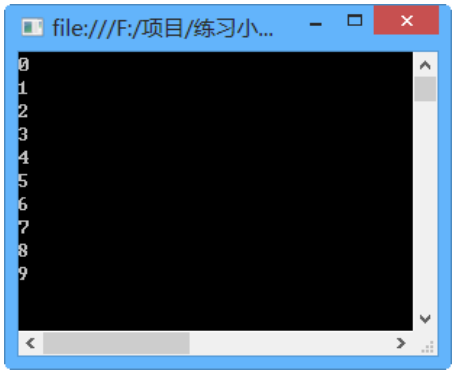

图 3-8　使用 while 循环输出 0～9

在此例中，先初始化 i 为 0，然后判断 while 的条件表达式 i<10 是否成立，若成立则输出 i 的值并换行，接着 i 自增 1，然后开始下一次的判断，判断 while 的条件表达式 i<10 是否成立，直到 i<10 不成立，循环终止。

3.2.3　do…while 循环语句

do…while 语句与 while 类似，do…while 语句是先执行循环体内的语句，再判断循环条件表达式是否成立，所以至少会执行一次循环体内的语句。

do…while 循环的语法格式如下：

```
do
{
    代码块
}
while (条件表达式);
```

do…while 循环语句的执行过程如图 3-9 所示。

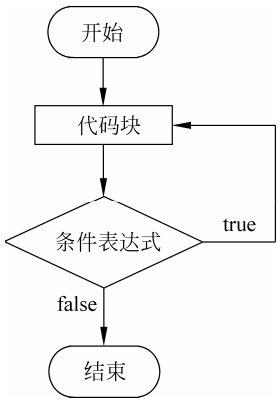

图 3-9　do…while 循环语句的执行过程

下面通过一个具体例子来详细讲解下 do…while 循环。

例 3-3：使用 do…while 循环输出 0～9（ConsoleDoWhileLoop）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleDoWhileLoop
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            do
            {
                Console.WriteLine(i++);
            }
            while (i < 10);
            Console.ReadLine();
        }
    }
}
```

运行结果如图 3-10 所示。

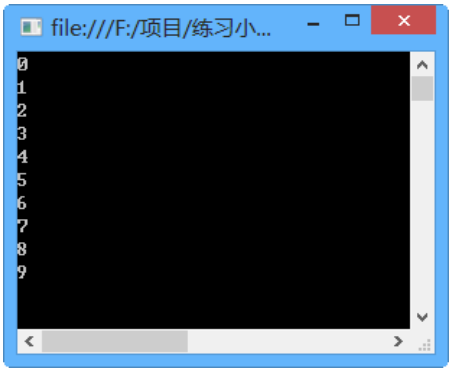

图 3-10　使用 do…while 循环输出 0～9

在此例中，先初始化 i 为 0，然后执行循环体内的语句输出 i 的值并换行，接着 i 自增 1，之后判断条件表达式 i<10 是否成立，若成立则继续下一次循环输出 i 的值并换行，接着 i 自增 1，然后开始下一次的判断，判断条件表达式 i<10 是否成立，直到 i<10 不成立，循环终止。

3.2.4　foreach 循环语句

foreach 循环语句用于枚举一个集合的元素，并对该集合中的每个元素执行一次循环体中的语句。

foreach 循环语句的语法格式如下：

```
foreach(类型 循环变量名 in 集合或数组)
{
    代码块
}
```

其中，类型和循环变量名用于声明循环变量，循环变量在整个代码块内有效。在循环过程中，循环变量表示当前正在执行循环的集合元素或数组元素。

foreach 循环语句的执行过程如图 3-11 所示。

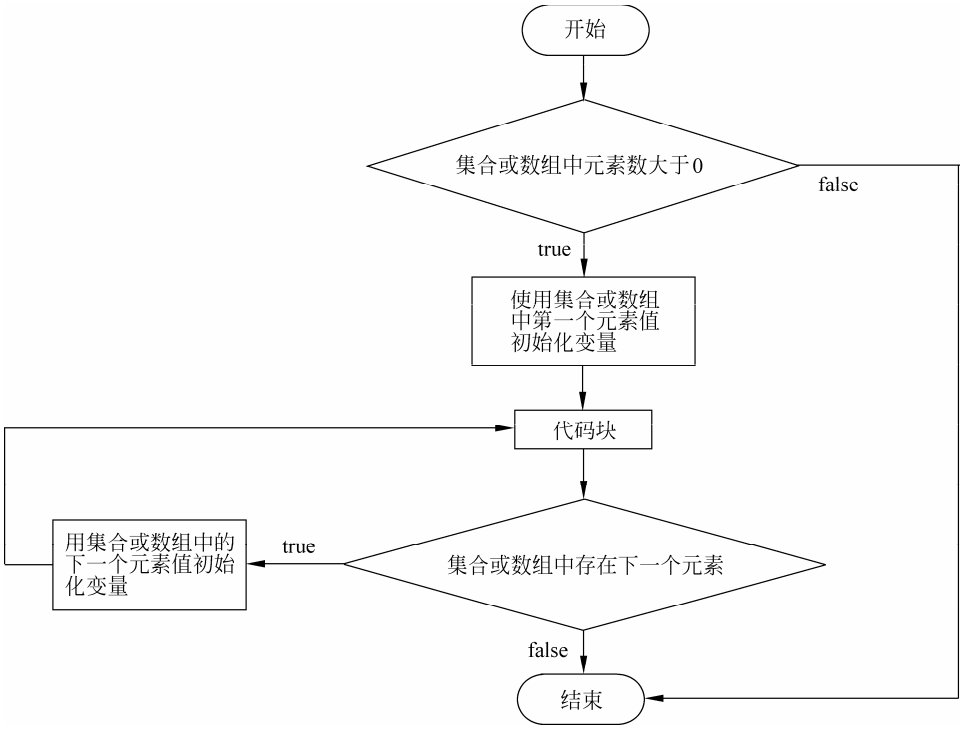

图 3-11　foreach 循环语句的执行过程

下面通过一个具体例子来详细讲解下 foreach 循环语句。

例 3-4：使用 foreach 循环语句输出数组中的元素（ConsoleForEachLoop）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleForEachLoop
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] strs = new string[] { "I", "am", "a", "programmer,", "I",
```

```
            "speak", "for", "himself." };
            foreach (string s in strs)
            {
                Console.Write(s + " ");
            }
            Console.ReadLine();
        }
    }
}
```

运行结果如图 3-12 所示。

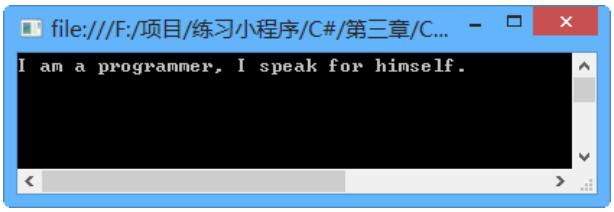

图 3-12　使用 foreach 循环语句输出数组中的元素

3.2.5　for、foreach、while 和 do…while 的区别

（1）for 循环语句一般用在对于循环次数已知的情况下，而 while 语句和 do…while 语句则一般用在对于循环次数不确定的情况下。for 循环必须使用整型变量作为循环的计数器，通过条件表达式限定计数器变量值来控制循环。

（2）用 while 语句和 do…while 语句时，对循环变量的初始化操作应该放在 while 语句和 do…while 语句之前，而 for 语句则可以在初始化语句中完成。

（3）while 语句和 do…while 语句实现的功能相同，唯一的区别就是 do…while 语句先执行后判断，无论表达式的值是否为 true，都将执行一次循环；而 while 语句则是首先判断表达式的值是否为 true，如果为 true 则执行循环语句；否则将不执行循环语句。

（4）while 语句和 do…while 语句，只在 while 后面指定循环条件，但是需要在循环体中包括使循环趋于结束的语句，而 for 语句则可以在迭代语句中包含使循环趋于结束的语句。

（5）foreach 语句只能对数据进行读操作，在其作用域内不能对进行遍历的值做修改，其遍历顺序只能递增。自动遍历给定集合的所有值。

3.2.6　双重循环

所谓双重循环就是循环中再嵌套循环进去。下面通过一个小例子简单介绍下双重循环的应用。

例 3-5：使用双重 for 循环语句输出如下样式的图案（ConsoleDoubleForLoop）

```
*
***
*****
*******
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleDoubleForLoop
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 1; i <= 4; i++)
            {
                for (int j = 1; j <= (i * 2) - 1; j++)
                {
                    Console.Write('*');
                }
                Console.WriteLine();
            }
            Console.ReadLine();
        }
    }
}
```

运行结果如图 3-13 所示。

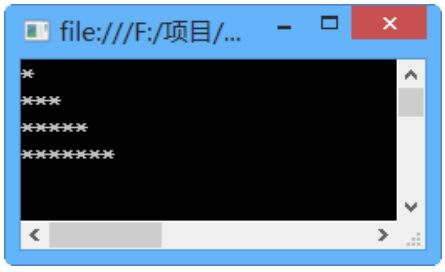

图 3-13　使用双重 for 循环语句输出指定样式的图案

3.3　跳转语句的应用

跳转语句主要用于无条件地转移控制到某个位置，这个位置就称为跳转语句的目标。跳转语句主要包括 break 语句、continue 语句、return 语句和 goto 语句 4 种。下面将对这几

种跳转语句做简单的介绍。

3.3.1 break 跳转语句

在之前讲解的 switch 语句中用到了 break 语句， break break 语句不仅可以使用在此，还可以把它用于 for、while、do…while 以及 foreach 循环语句中。break 语句使程序跳出当前循环，并继续执行循环之后的代码。当 break 出现在嵌套循环语句中时，break 语句只能出现在最里层的语句中。如果要穿越多个嵌套语句，则必须使用 goto 语句。

下面通过一个小例子来讲解下 break 语句的用法。

例 3-6：break 语句的使用（ConsoleBreak）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleBreak
{
    class Program
    {
        static void Main(string[] args)
        {
            int num = 0;
            string[] strs = new string[] { "LiLei", "HanMeimei", "LinTao",
            "Kate", "Uncle Wang", "Jim", "Tom", "Lily", "Lucy", "Ann",
            "polly" };
            foreach (string s in strs)
            {
                num++;
                if (s == "Tom")
                {
                    Console.WriteLine("找到 Tom 了,他在编号" + num + "的位置！");
                    break;
                }
                Console.WriteLine("位置" + num + "处没有 Tom");
            }

            Console.ReadLine();
        }
    }
}
```

运行结果如图 3-14 所示。

此例使用 foreach 语句遍历数组，查找“Tom”所在的位置，找到后使用 break 语句结束循环。

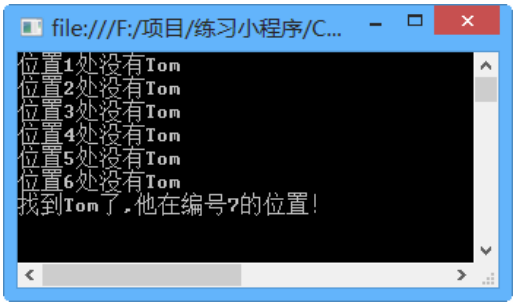

图 3-14　break 语句的使用

3.3. 2　continue 跳转语句

continue 语句与 break 语句类似，必须出现在 for、while、do…while 和 foreach 循环语句中，continue 语句的作用是退出当前循环结构的本次循环，并开始执行下一次循环，而不是退出当前循环结构。当嵌套循环语句存在时，continue 语句只能使直接包含它的循环语句开始一次新的循环。

下面通过一个小例子来讲解下 continue 语句的用法。

例 3-7：continue 语句的使用（ConsoleContinue）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleContinue
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 1; i <= 10; i++)
            {
                if (i % 2 != 0)
                    continue;
                Console.WriteLine(i);
            }
            Console.ReadLine();
        }
    }
}
```

运行结果如图 3-15 所示。

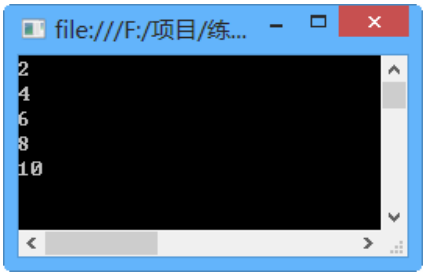

图 3-15　continue 语句的使用

此例使用 for 循环语句，对循环变量 i 取余，如果余数不为 0，则 continue 语句就终止当前循环，继续下一次循环，所以只输出 2、4、6、8、10。

3.3.3　return 跳转语句

return 语句终止它出现在其中的方法的执行并将控制返回给调用方法。 它还可以返回一个可选值。如果方法为 void 类型，则可以省略 return 语句。

如果 return 语句位于 try 块中，则将在控制流返回到调用方法之前执行 finally 块（如果存在）。

下面通过一个小例子来讲解下 return 语句的用法。

例 3-8：return 语句的使用（ConsoleReturn）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleReturn
{
    class Program
    {
        static void Main(string[] args)
        {
            while (true)
            {
                string num1, num2, choose;
                Console.WriteLine("请输入一个数字(输入“e”退出程序)");
                num1 = Console.ReadLine();
                if (num1 == "e")
                {
                    break;
                }
                Console.WriteLine("请输入另一个数字");
                num2 = Console.ReadLine();
                Console.WriteLine("请选择要进行的运算：");
                Console.WriteLine("1.+\t2.-\t3.*\t4./");
                choose = Console.ReadLine();
                Console.WriteLine("计算结果为：{0}", Operational(Convert.
                ToDouble(num1), Convert.ToDouble(num2), choose));
```

```
                Console.WriteLine("————————————————");
            }
            Console.WriteLine("您已经成功退出,再见！");
            Console.ReadLine();
        }

        public static string Operational(double num1, double num2, string
        choose)
        {
            switch (choose)
            {
                case "1":
                    return (num1 + num2).ToString();
                case "2":
                    return (num1 - num2).ToString();
                case "3":
                    return (num1 * num2).ToString();
                case "4":
                    return (num1 / num2).ToString();
                default:
                    return "输入有误！";
            }
        }
    }
}
```

运行结果如图 3-16 所示。

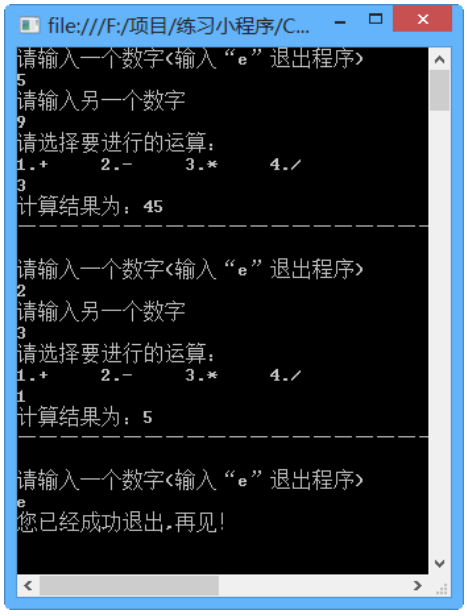

图 3-16　return 语句的使用

此实例实现了一个简易的加减乘除计算器的效果，代码中根据输入的 choose 值判断做何种运算，switch 语句的每个 case 中使用 return 语句直接将转为 string 类型后的计算结果返回了函数调用处，这又引出了 case 中的一种特殊情况——当一个 case 中需要 return 语句时，break 语句要省略，因为两者共存没有任何意义（位于后面的一个语句将永远无法被程

序执行到）。关于函数的相关知识将在后续章节进行讲解，此处不做具体讲解。注意此实例未进行异常处理，旨在讲解 return 的用法。

3.3.4 goto 语句

goto 语句用于将控制转移到事先定义的标签标记的语句。goto 语句可以应用于 switch 语句中的 case 标签和 default 标签以及事先定义的标记语句所声明的标签。goto 语句的三种形式及介绍如下。

```
goto case 表达式:
```

goto case 语句的目标为它所在 switch 语句中的某个语句列表。

```
goto default:
```

goto default 语句的目标是它所在的 switch 语句的 default 标签。

```
goto 【标签】:
```

goto 【标签】语句的目标是具有给定标签的标记语句。此用法可用于跳出多层嵌套循环语句。但是，不能使用 goto 语句从外部进入循环。

下面通过一个小例子来讲解下 goto 语句的用法。

例 3-9：goto 语句的使用（ConsoleGoto）

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleGoto
{
    class Program
    {
        static void Main(string[] args)
        {
            while (true)
            {
                Console.WriteLine("请输入任意字符，(输入"e"退出)");
                string str = Console.ReadLine();
                if (str == "e")
                    goto exit;
                Console.WriteLine("您输入了" + str);
            }
        exit:
            Console.WriteLine("您已经成功退出循环！");
            Console.ReadLine();
        }
    }
}
```

运行结果如图 3-17 所示。

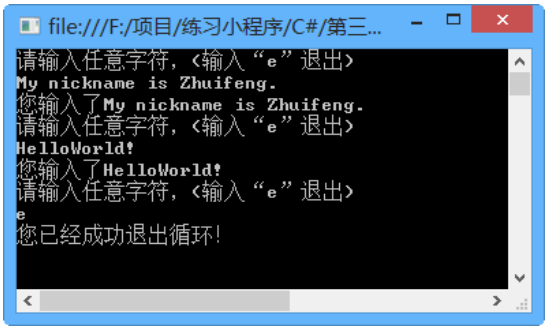

图 3-17　goto 语句的使用

此实例演示了 goto 【标签】的用法，当输出“e”时，程序直接跳出了 while 循环，并在定义为 exit 的标签处继续执行程序。

goto 在使用时存在一些争议，有人建议避免使用它，因为它会打乱程序代码的执行顺序，使程序代码的可维护性较差，所以应尽量避免使用 goto 语句。

3.3.5　各跳转语句的区别

（1）break 语句：跳出（终止）循环。
（2）continue 语句：跳出（终止）当前的循环，继续执行下一次循环。
（3）return 语句：跳出循环以及其包含的函数。
（4）goto 语句：可以跳出循环到已标记好的位置上。

小结

本章主要学习了流程控制语句，其中包括选择语句、迭代语句、跳转语句等。这些控制语句将出现在任何一个成熟健壮的应用程序中，读者应熟练掌握这些语句的用法，为日后开发打下良好的基础。