第 3 章　流程控制语句

王忠民

内容简介

和 Java 、 C++ 、 Pascal 等其他编程语言一样， C# 语言也是由各种各样的语句有序构建而成的，它提供了控制程序步骤的基本手段。如果没有流程控制语句，整个程序将按照线性的顺序来执行，不能根据用户的输入条件来决定执行的顺序。本章将详细介绍 C# 语言中的流程控制语句，包括基本语句、选择语句、循环语句和跳转语句等内容。

学习本章后，用户能够熟练掌握流程控制语句的语法形式和使用规则，为后面的程序开发奠定基础。

本章学习要点

- 熟悉空语句、语句块、声明语句和表达式语句的作用
- 掌握 if 语句、 if else 语句和嵌套 if 语句的使用方法
- 掌握 for 循环语句的使用方法
- 掌握 while 循环语句的使用方法
- 掌握 do while 语句的使用方法
- 掌握 foreach 语句的使用方法
- 熟悉 while 语句和 do while 语句的异同点
- 熟练使用多重语句输出各种图形
- 了解各种跳转语句的作用

3.1 基本语句

C# 语句中包含多种常用的基本语句，如空语句、声明语句和语句块等。本节详细介绍这些语句的使用方法。

3.1.1 空语句

只有分号“;”组成的语句就称为空语句，它在应用程序中什么都不做，且不包含任何实际性的语句。如果要求有语句的上下文中不执行任何操作时，使用空语句。

```
class Program
{
    static bool isCheck = false;
        public static void Test()
    {
        while (isCheck) ;
    }

    public static void Main(string[] args)
    {
        Test();
        Console.ReadLine();
 }  }
```

上述代码中无论 isCheck 的值是什么， while 语句都不执行任何操作

3.1.2 语句块

语句块中可以包含多个语句，一个语句块可以看作是多个语句的组合。它的语法形式如下所示：

```
{
   statement-list  // 语句列表
   …… 
   …… 
}
```

上面语法中 statement-list 表示一个或多个语句组成的列表，如果不存在 statement-list ，则称该块是空的。

语句块的执行规则如下：

- 如果语句块的内容为空，控制转到块的结束点
- 如果语句块的内容不为空，控制转到语句列表。
- 当控制到达语句列表的结束点时，控制转到块的结束点

3.1.2 语句块

例：创建一个语句块，该语句块包含 4 条语句。第一条语句声明 3 个 int 类型的变量；中间两条语句分别为变量 firstnum 和 secondnum 赋值。最后将这两个变量的值相乘，结果保存到 total 变量中。

```
{
    int  firstnum, secondnum, total;
   firstnum = 100;
   secondnum = 10;
   total = firstnum * secondnum;
}
```

3.1.3 声明语句

声明语句主要用来声明常量或者变量。

例如下面语句演示了常量和变量的声明：

```
static  string  name = "Jack“;      // 声明字符型的静态变量
double  money = 300.23;                    // 声明 double 类型的实例变量
const  double  Radius = 3.14;              // 声明 double 类型的常量
const  int  Size = 100;                    // 声明整型的常量
```

3.1.4 表达式语句

表达式语句是 C# 语言中最常见的语句之一。实际上它的值只是一个中间结果，常用的表达式语句主要包括 7 类，如下所示：

- 赋值表达式 如“ num=2002”
- 创建对象表达式 如“ object obj = new object()”
- 前缀递增表达式 如“ ++num”
- 前缀递减表达式 如“ --num”
- 后缀递增表达式 如“ num++”
- 后缀递减表达式 如“ num--”
- 调用表达式 如“ count()”

3.2 选择语句

选择语句是指根据表达式的值从若干个给定的语句中选择一个来执行的语句。本节介绍常用的判断语句，如 if 语句、 if else 语句和 switch 语句等。

3.2.1 if 语句

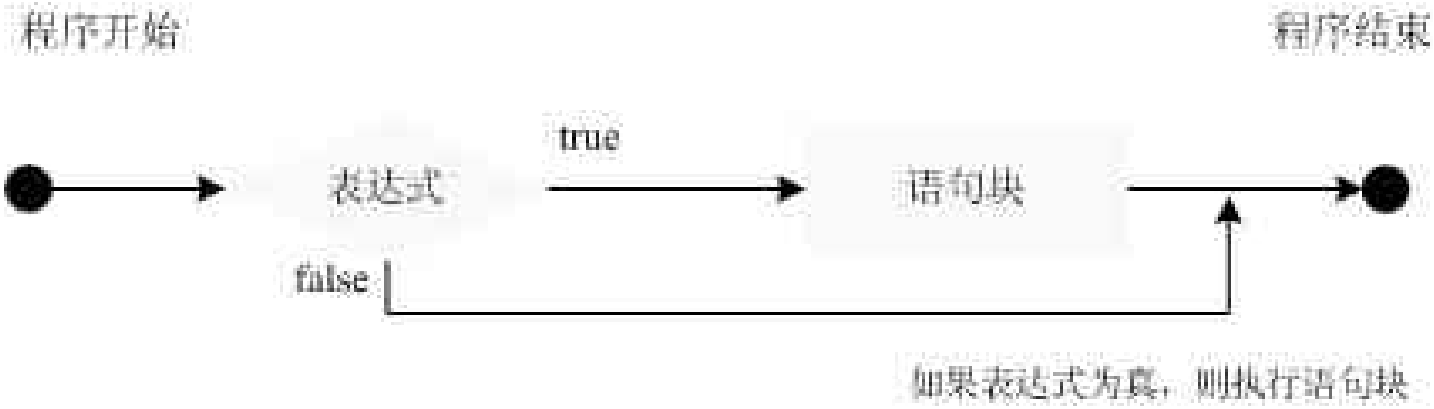
if 语句是使用最多的条件分支结构，它属于选择语句，也可以称为条件语句。根据布尔表达式的值判断是否执行语句块的内容。

```
if ( 表达式 )
{
    语句块
}
```

```
// 如果只有一条语句

if ( 表达式 )  语句；
```

```
// 如果只有一条语句

if ( 表达式 )
    语句；
```

其中表达式可以是任何一种逻辑表达式。
if 语句的执行流程如图所示。



例：从控制台输入用户名和密码，首先判断用户名中是否包含“ @” 符号，然后再判断密码是否为空或密码长度是否小于 6 。代码如下：

```
Console.Write(" 请输入用户名： ");
string  userName = Console.ReadLine();
Console.Write(" 请输入用户密码： ");
string  userPass = Console.ReadLine();
if (userName.Contains("@"))
{
   Console.WriteLine(" 对不起，您的用户名包含非法字符！ ");
}
if (string.IsNullOrEmpty(userPass) || userPass.Length < 6)
{
   Console.WriteLine(" 用户密码不能为空，且长度不能小于 6 位字符 ");
}
Console.ReadLine();
```

```
请输入用户名：张萱妍@mail.com
请输入用户密码：000
对不起，您的用户名包含非法字符!
用户密码不能为空，且长度不能小于6位字符
```

3.2.2 if else 语句

if else 语句是 if 语句的扩展体，它主要控制两个语句块。

```
if ( 条件表达式 )
{                    // 如果只包含一条语句，则可省略 {}
   语句块 1
}
else
{                    // 如果只包含一条语句，则可省略 {}
   语句块 2
}
```

上面语句的执行过程是：首先判断 if 语句后面的表达式的值，如果该值为 True 则执行语句块 1 的代码，否则执行语句块 2 的代码。

表达式为真则执行语句块1，否则执行语句块2

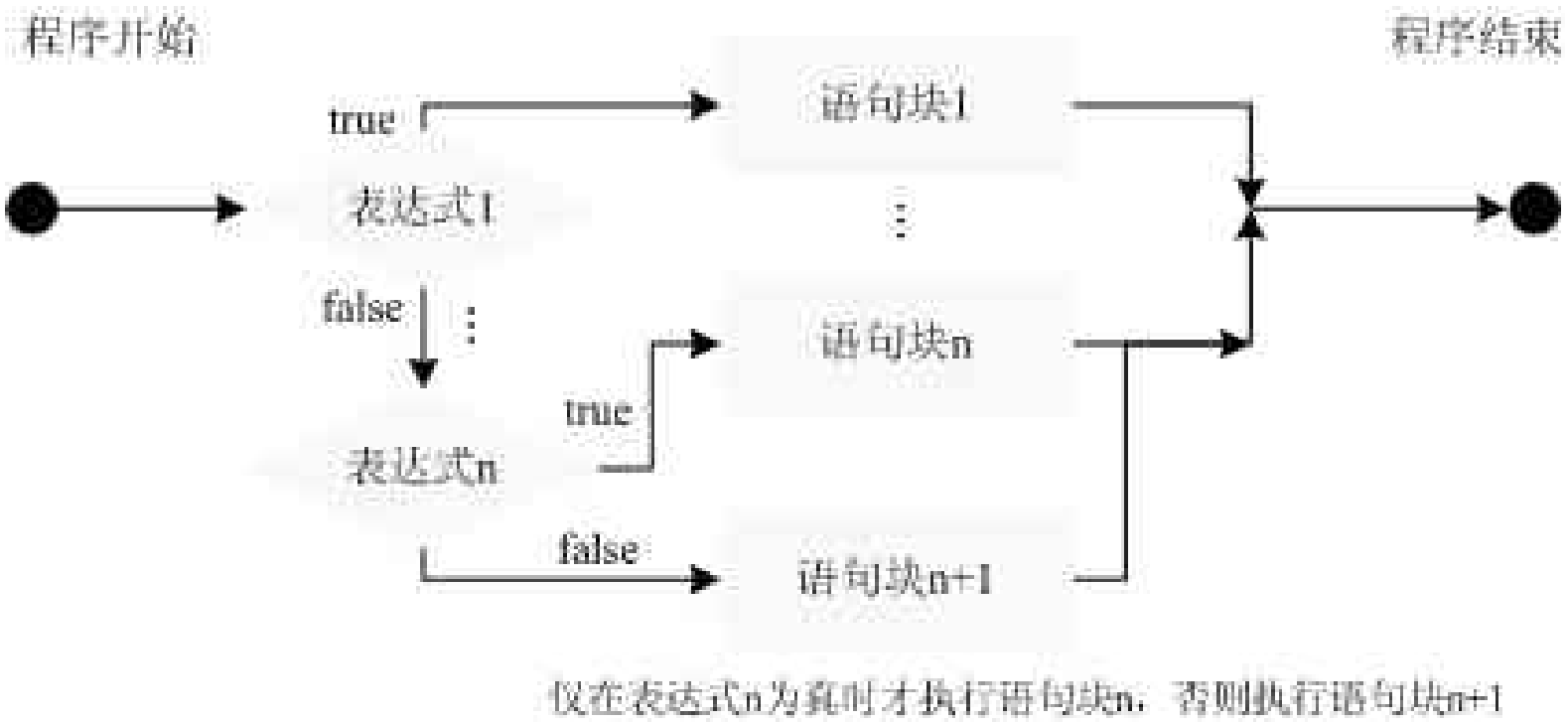
```
public  static  void  Main(string[] args)
  {
     DateTime   dt  =  DateTime.Now;
     if  ( dt.Hour > 4  && dt.Hour < 9)
          Console.WriteLine(" 早晨好！ ");
     else
          Console.WriteLine(" 努力工作吧！ ");

     Console.ReadLine();
   }
```

3.2.3 if -- else if -- else

无论是 if 语句还是 if else 语句，它们给应用程序只提供了一个分支。但是，C# 中的应用程序分支可以有多个，这时候可以使用 if else if else 语句，它由多个 if else 语句组合而成。

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

- 首先判断表达式 1 的值，如果它的值为 true 则执行语句块 1 ；否则依次判断 else if 中表达式的值。
- 当某个条件表达式的值为 true 时则执行该分支相应的语句块。
- 如果所有分支的值都为 false 则执行语句块 n+1 ，然后继续执行程序后面的代码。


程序开始
程序结束
true
语句块1
表达式1
false
语句块n
true
表达式n
false
语句块n+1
仅在表达式n为真时才执行语句块n，否则执行语句块n+1

```
public static void Main(string[] args)
   {
      Console.WriteLine(" 请选择你的身份： ");
      Console.WriteLine("1 、普通管理员  2 、超级管理员  3 、游客  ");
      int  i = int.Parse(Console.ReadLine());
      if (i == 1)
         Console.WriteLine(" 你好，普通管理员可以查看系统全部信息。 ");
      else  if (i == 2)
       { Console.WriteLine(" 你好，超级用户可以查看和修改、删除。 "); }
      else  if (i == 3)
         Console.WriteLine(" 你好，游客只能查看自己的信息。请注册！ ");
      else
         Console.WriteLine(" 选择的编号错误 ");

      Console.ReadLine();
    }
```

3.2.4 嵌套 if 语句

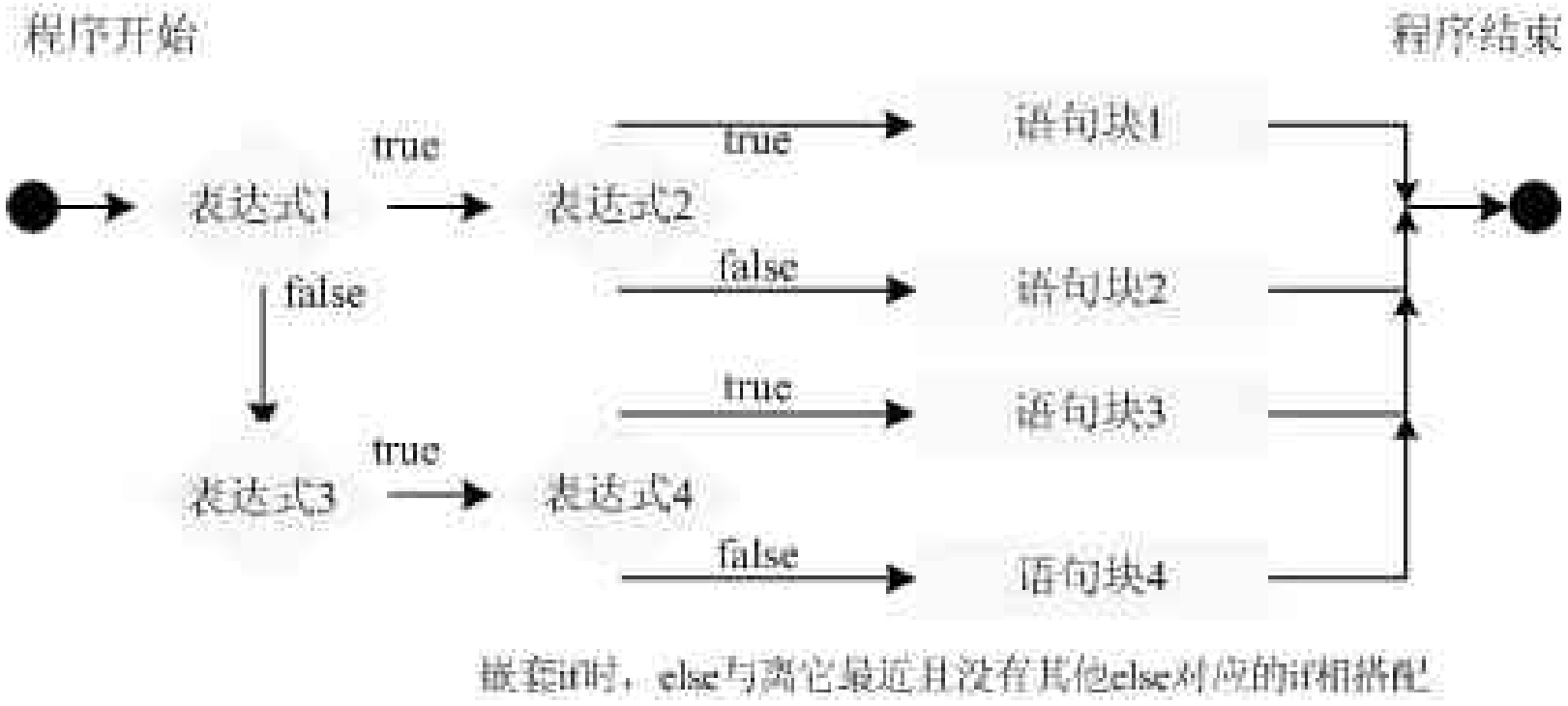
嵌套 if 语句就是指在 if 语句或 else 语句中，嵌套了多个 if else 语句。

if 语句、 if else 语句和 if else if else 语句之间可以相互嵌套，并且可以嵌套多个 if 语句。

```
if ( 表达式 1)
{
  if ( 表达式 2)
  {
    语句块 1
  }
  else
  {
    语句块 2
  }
}
else
{
  if ( 表达式 3)
  {
    语句块 3
  }
  else
  {
    语句块 4
  }
}
```

3.2.4 嵌套 if 语句

上述语法中 if else 语句又分别嵌套了一个 if else 语句。在执行嵌套 if 语句时程序会首先执行外层的 if else 语句，然后再执行内层的 if else 语句。嵌套 if 语句的执行流程如图所示。



嵌套if时，else与离它最近且没有其他else对应的if相搭配

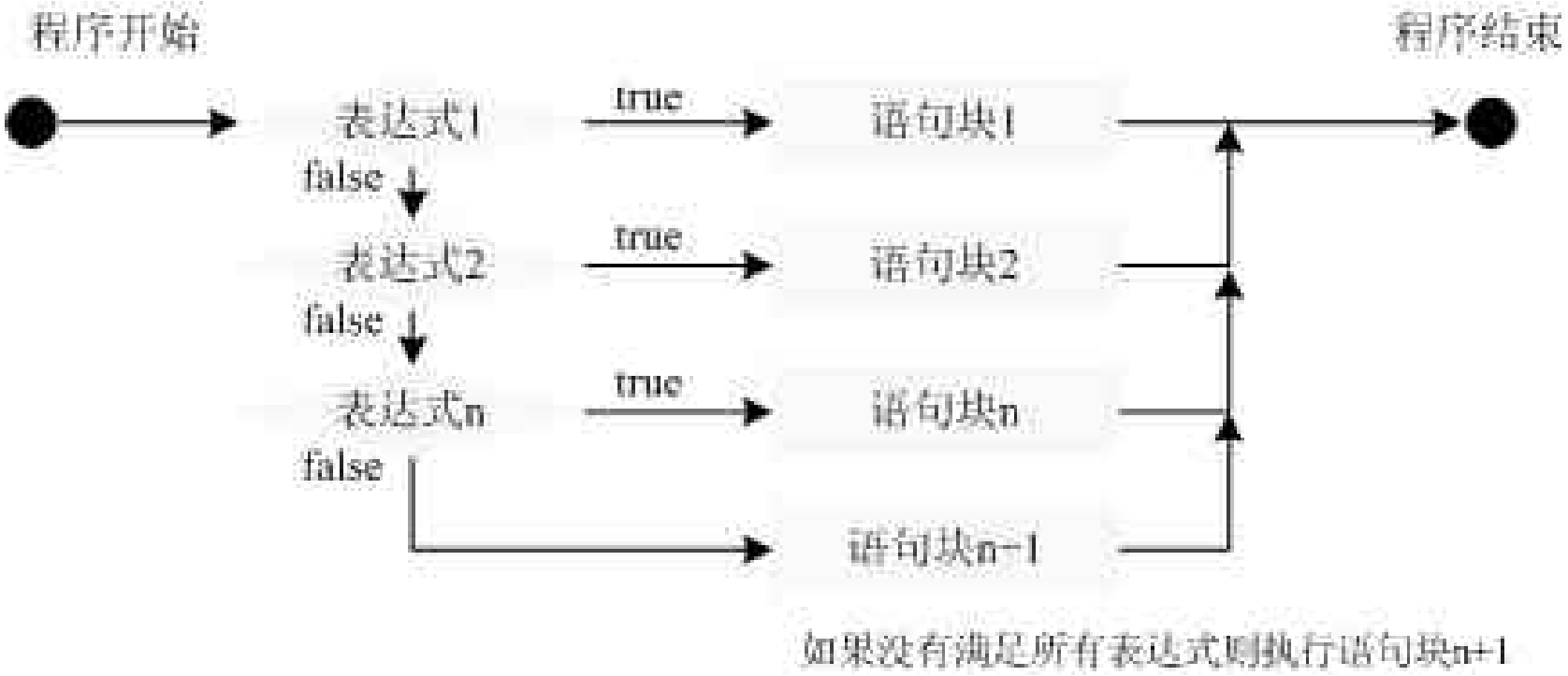
3.2.5 switch 语句

switch 语句提供多路分支。可以把它看作是 if else 语句的另一种实现方式。如果需要比较有很多值的变量，可以使用 switch 语句。

```
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;
    break;
}
```

- 表达式将和每个 case 子句的值进行比较，如果匹配成功则执行相关语句块。
- 如果都不匹配则执行 default 中的语句块。
- 每个语句块以 break 语句结尾，否则编译错误。
- 可以没有 default 语句块。

3.2.5 switch 语句



例：输入出生的月、日，计算出星座

```
public static void Main(string[] args)
  {
     Console.Write(" 请输入出生的月、日（ MMDD ）： ");
     int  monthday = Convert.ToInt32(Console.ReadLine());
     int  month = monthday / 100;
     int  day = monthday % 100;
     string  xingzuo="";
     switch (month)
     {
        case 1:
          xingzuo = day < 21 ? " 魔蝎座 " : " 水瓶座 ";
          break;
        case 2:
          xingzuo= day <20 ? " 水瓶座 ":" 双鱼座 ";
          break;
     }
     Console.WriteLine(xingzuo);
     Console.ReadLine();
  }
```

下面程序执行时，若输入２，则输出结果是？

```
int  i = Convert.ToInt32(Console.ReadLine());
switch (i++)
{
    case 1:
        Console.WriteLine("1:"+i);
        break;
    case 2:
        Console.WriteLine("2:" + i);
        break;
 }
```

找错：

```
switch（i*3）
 {
     case 1：
         Console.WriteLine("1:"+i);
         break;
     case 2 ：
         Console.WriteLine("2:" + i);
         break;
 }
```

3.3 循环语句

循环语句也叫迭代语句，它可以重复执行嵌入语句的语句块。被重复执行的语句称之为循环体，能否继续重复执行取决于循环条件。本节就详细介绍 C# 中的 4 种循环语句。

3.3.1 for 语句

for 循环语句通常用在确定循环次数的情况下，语法格式如下：

```
for ( 初始化表达式 ; 条件表达式 ; 迭代表达式 )
{
   语句块
}
```

上述语法格式中程序首先**执行**初始化表达式的代码，然后**判断**是否满足条件表达式，如果满足条件则执行语句块的代码。执行语句块完毕后**执行**迭代表达式的代码，如果条件表达式仍然成立则继续执行语句块的代码，再重新执行迭代表达式。重复上面的步骤直到不满足条件表达式为止。

例：计算 1~100 所有整数的和、所有奇数的和、所有偶数的和。

```
public static void Main(string[] args)
  {
     int  totalSum=0,eventSum = 0,oddSum = 0;
     for (int  i = 1; i <= 100;i++ )
     {
        totalSum  += i;
        if (i % 2 == 0)
           eventSum += i;
        else
           oddSum += i;
     }
     Console.WriteLine(“ 总和 {0} ，奇数和 {1} ，偶数和 {2}",totalSum,oddSum,eventSum

     Console.ReadLine();
   }
```

长语句可书写成多行

Console.WriteLine(“ 总和 {0} ，奇数和 {1} ，偶和 {2}",totalSum,oddSum,eventSum

```
Console
        .
        WriteLine
        ( “ 总和 {0} ， 奇数和 {1} ， 偶和 {2}“ ,
        totalSum
        ,
        oddSum
        ,eventSum);
```

for 语句可以单独使用，也可以嵌套使用，构成多重循环。

例：根据输入的行数和列数，绘制长方形。

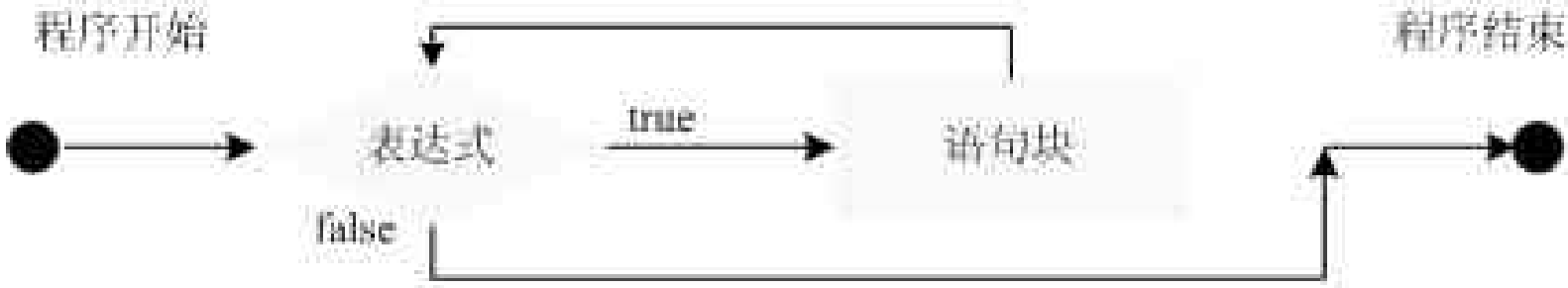
```
public static void Main(string[] args)
  {
     Console.WriteLine(" 请分别输入行数、列数： ");
     int  row = Convert.ToInt32(Console.ReadLine());
     int  col = Convert.ToInt32(Console.ReadLine());
     for  ( int i = 1;  i <= row; i++ )
     {
        for  (int  j = 1;  j <= col;  j++)
           Console.Write("*");
        Console.WriteLine();
     }
     Console.ReadLine();
   }
```

3.3.2 while 语句

for 语句是用户确定循环次数时使用的循环语句。如果不知道程序要循环的次数，怎么办？在 C# 中有另外一种循环： while 循环。 while 循环语句能够按照不同的条件执行一个嵌入语句零次或多次。语法如下：

```
while ( 条件表达式 )
{
    语句块
}
```

在 while 语句的语法中首先判断表达式是否满足条件，如果满足条件则执行语句块的内容。否则直接执行 while 语句后面的内容。



当表达式为真则执行语句块，直到为假

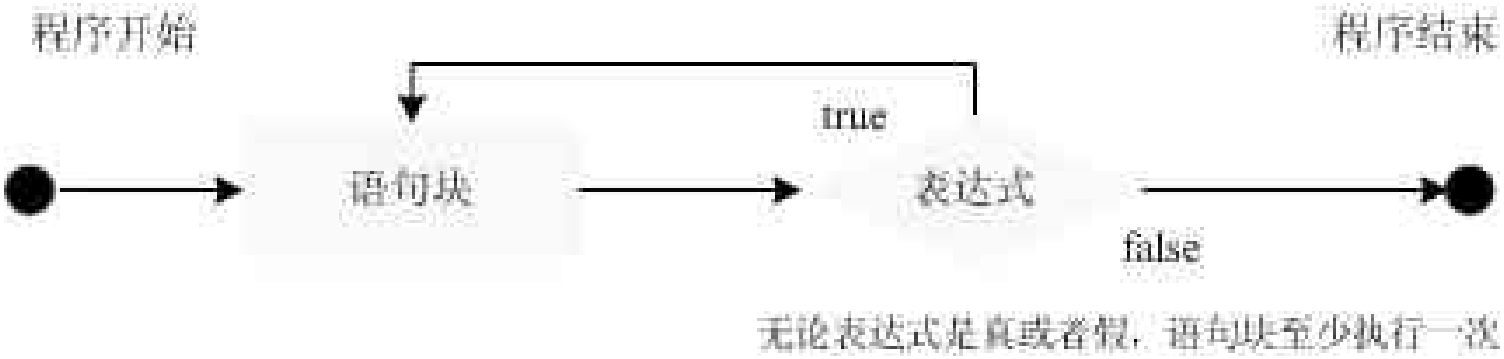
例：求 1~100 之间所有 5 的倍数的和。

```
public  static  void  Main(string[] args)
  {
     int  i = 1 , sum =0;
     while (i<=100)
     {
        if (i % 5 == 0)
        {
           sum += i;
           Console.WriteLine("{0},  {1}", i, sum);
        }
        i++;
     }
     Console.ReadLine();
   }
```

3.3.3 do while 语句

do while 语句和 while 语句相似，它是 while 循环语句的扩展。 do while 语句能够按照不同的条件执行嵌入语句一次或多次。特点是：先执行，后判断。

```
do
{
   语句块
}while( 条件表达式 );
```



do while 语句也是由循环条件和循环体组成的，但它 while 语句略有不同。do while 语句的特点为：先执行循环体，然后判断循环条件是否成立。而 while 语句先判断是否满足条件，如果满足条件，再执行循环体。

例：猜数游戏，电脑生成 0~3 的随机数。与输入数做比较。若输入负数结束。

```
public static void Main(string[] args)
   {
      Random  rnd = new  Random();   // 用 Random 类生成随机数
      int  i,j;
      do
      {
         i = rnd.Next(4);  // 生成一个小于 4 的非负随机数
         j = Convert.ToInt32(Console.ReadLine());
         if  (i == j )
            Console.WriteLine(" 猜对了 !");
         else
            Console.WriteLine(" 错 ！答案是 {0}, 你猜的是 {1}",i, j);
      } while (j >= 0);

      Console.ReadLine();
    }
```

Random 的用法

①int Next() ; 返回大于等于零且小于 Int32.MaxValue 的 32 位带符号整数。
② int Next(int maxValue); 返回大于等于零且小于 maxValue 的 32 位带符号整数。
③ int Next(int minValue, int maxValue) ; 返回一个大于等于 minValue 且小于 maxValue 的 32 位带符号整数。

```
int k;
Random rnd = new Random();
k = rnd.Next();    // 返回非负随机数 , 整数
k = rnd.Next(20);  //  返回非负随机数 , 整数 ， 0<= k < 20
k = rnd.Next(-11, 3); // 返回两值之间的随机整数 ， -11<= k < 3
```

3.3.4 foreach 语句

foreach 语句是一个特殊的 for 循环语句，它用于枚举一个集合元素，迭代出集合中的每一项，但是不能修改集合中的每一项（即：循环体内不能修改 item 的值）。

```
foreach (type  item  in  collection)
{
   语句块
}
```

可用于数组、 Arraylist 等

```
public static void Main()
{
    string[] numbers = new  string [4]{"one","two","three","four"};

    foreach(string  s  in  numbers)
    {
        foreach(char  ch  in  s)
            Console.WriteLine(ch);
        Console.WriteLine("---");
    }

    Console.ReadLine();
}
```

3.4 跳转语句

除了基本语句、选择语句和循环语句外，C# 中还有另外的一种语句：跳转语句。跳转语句用于无条件的转移，主要包括 break 、 goto 、 return 和 continue 等语句。

3.4.1 break 语句

break 语句从最近层的循环语句中（ switch 、 while 、 do 、 for 、 foreach ）跳出，从而执行循环语句下面的语句。（与 if 语句无关）

```
static void Main(string[] args)
{
    string[]  famous = { " 红楼梦 ", " 三国演义 ", " 水浒传 ", " 西游记 " };
    foreach (string  fam  in  famous)
    {
        if (fam == " 水浒传 ")
          {    break;   }
        else
          {  Console.Write(fam);   }
        Console.WriteLine("!");
    }
    Console.ReadLine();
}
```

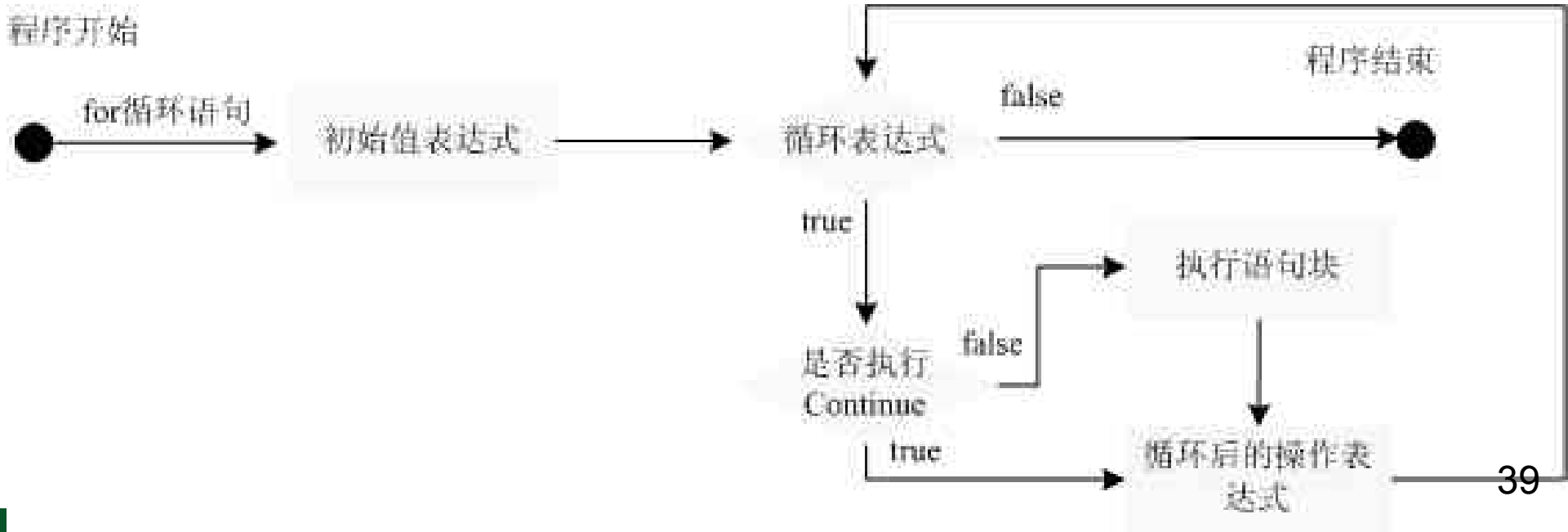
控制台输出的结果是？

3.4.2 continue 语句

continue 语句只能出现在循环体中，它表示跳出本次循环，继续执行下一次循环。 continue 语句只能用在 while 、 do 、 for 、 foreach 语句中，否则会发生编译时错误。

continue 语句和 break 语句相似，但是它们最大的区别在于： continue 语句是跳过循环体中剩余的语句而强制执行下一次循环，而 break 语句是直接跳出循环语句。

continue 语句的在 for 循环语句中的执行流程：



把上例的 break 改成 continue ：

```
static void Main(string[] args)
{
    string[]  famous = { " 红楼梦 ", " 三国演义 ", " 水浒传 ", " 西游记 " };
    foreach (string  fam  in  famous)
    {
        if (fam == " 水浒传 ")
        {
            continue;
        }
        else
        {
            Console.Write(fam);
        }
        Console.WriteLine("!");
    }
    Console.ReadLine();
}
```

3.4.3 return 语句

return 语句用于终止其执行的方法，并将控制返回给调用方法。

1.如果方法没有返回类型，可以省略 return 语句。

2.如果方法有返回类型， return 语句必须返回这个类型的值。

例：裴波那契数列又称黄金分割数列，它指的是这样的数列：1、1、2、3、5、8、13、21……。数列中后一个数值总是等于前两个数字的和。下面实现裴波那契数列：

1、1、2、3、5、8、13、21……

```
public static int Fibonacci(int number)
{
   if (number == 1 || number == 2)
      return  1;
   else
      return  Fibonacci(number - 1) + Fibonacci(number - 2);
}

static  void  Main(string[] args)
{
   int  inputnum  =  Fibonacci(14);
   Console.WriteLine(" 第 14 个数字的值应该是： " + inputnum);
}
```

运行输出的结果是：

```
第 14 个数字的值应该是： 377
```

3.4.4 goto 语句

goto 语句也叫无条件转移语句，它可以跳转到指定的标签位置。

```
goto   语句标号；
```

goto 语句多用于 switch 语句，实现由某个 switch 的 case 标签或 default 标签跳转到另一个 case 标签或 default 标签。

goto 也可以用在嵌套语句中，使程序跳出多层循环。

goto 不能跳出类的范围。

goto 不能跳到类似于 for 循环那样的语句块中。

例：如果选择的不是 C 则 goto 跳转到 Choose 标签，否则跳转到 Right 标签。

```
static  void  Main(string[] args)
{
   Choose:
   Console.WriteLine(" 下列四个选项中，哪项不是中国古代四大发明？ ");
   Console.WriteLine("A. 造纸术  B. 指南针  C. 医药  D. 火药  E. 印刷术 ");
   Console.Write(" 您选择 :");string answer = Console.ReadLine();
   if (answer == "C")
   { goto  Right; }
   else
   {
      Console.WriteLine(" 选择答案错误，请重新选择！ ");
      goto  Choose;
   }
 Right:
 Console.WriteLine(" 恭喜您，回答正确！ ");
 Console.ReadLine();
}
```

3.5 实现简单的计算器

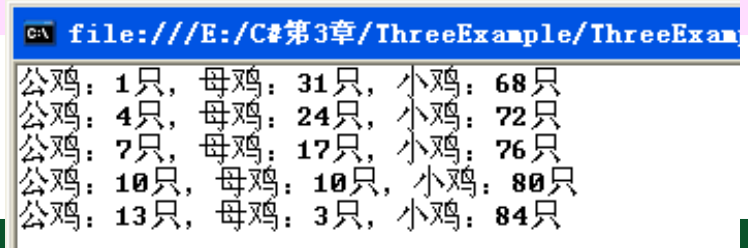
本节将前几节所讲的部分内容相结合实现一个简单的计算器。首先提示用户输入两个操作数，然后输入操作符根据用户输入的操作符输出操作的结果。

```
 static  void  Main(string[] args)
{
   OperNum:
      Console.Write(" 请输入第一个操作数： ");
      double num1 = Convert.ToDouble(Console.ReadLine());
      Console.Write(" 请输入第二个操作数： ");
      double num2 = Convert.ToDouble(Console.ReadLine());
      Console.Write(" 请输入运算符号（ + 、 - 、 * 、 / 、 % ） ： ");
      string  oper = Console.ReadLine();
      switch (oper)
      {
         case "+":
            Console.WriteLine("{0} 加上 {1} 等于 {2}", num1, num2, num1 + num2);
            break;
          // 省略其他情况
         default:
            Console.WriteLine(" 您输入的操作符号不合法！ ");
            goto OperNum;
            break;  // 此句执行不到，可省略
      }
      Console.Read();
 }}
```

3.6 百钱买百鸡

100 元钱买 100 只鸡。小鸡 1 元 2 只、母鸡 2 元 1 只、公鸡 4 元 1 只。如果 100 元钱单独买公鸡能买 25 只，单独买母鸡能买 50 只。把公鸡数量作外循环，母鸡数量作内循环。

```
static void Main(string[] args)
{
   for (int gong = 0; gong <= 25; gong++)
   {
      for (int mu = 0; mu <= 50; mu++)
      {
         int  xiao = 100 - gong - mu;
         if ( 100 - mu *2 - gong*4) * 2 == xiao)
            Console.WriteLine(" 公鸡： {0} 只，母鸡： {1} 只，小鸡： {2} 只 ", gong,
mu, xiao);
      }
   }
}
```



作业：发至 ustb100@163.com ，在主题里写明：班级、学号、姓名

1 、编写控制台程序，利用循环输出 9*9 乘法表。

2 、循环输入班里每个人的英语成绩，输入 -1 时结束。最终输出全班平均分。

3 、本文件第 17 页的程序，把 if 改为 switch 实现。