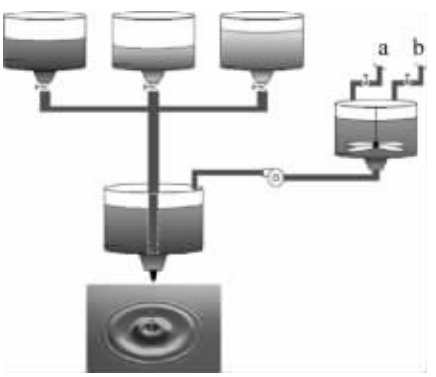
第 3 章　C# 控制语句

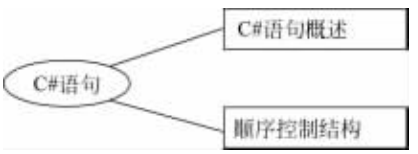
程序控制与工业生产线控制是相似的



本章指南

3.1 C#语句

知识梳理



3.1.1 C#语句概述

语句是C#程序的基本组成元素，用于定义变量和常量、创建对象、变量赋值、调用方法、控制分支和循环等。语句通常以分号终止，只有一个分号的语句称为空语句。由大括号({和})括起来的一系列语句构成语句块(block)或复合语句。

例如，以下就是一个语句块。

```
{    int n;
     n = int.Parse(Console.ReadLine());
     n += 10;
     Console.WriteLine(n);
}
```

3.1.2 顺序控制结构

C#程序中执行的基本顺序按各个语句出现位置的先后次序执行，称为“顺序控制结构”，如图3.1所示，按语句/语句块1→语句/语句块2→语句/语句块3的顺序执行。例如，上述语句块中包含的4个语句就构成一个顺序控制结构。

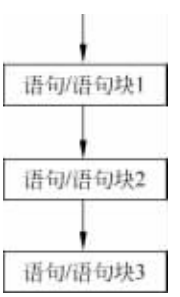
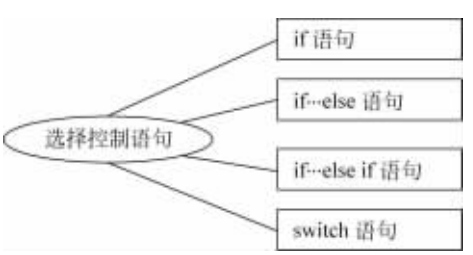


图3.1 顺序控制语句的执行流程

除了顺序控制结构的语句外，C#还包含选择控制语句和循环控制语句。为了叙述方便，后面将单个语句和语句块统一称为语句。

3.2 选择控制语句

知识梳理



3.2.1　if 语句

if 语句用于在程序中有条件地执行某一语句序列，其基本语法格式如下。

```
if (条件表达式) 语句;
```

其中，“条件表达式”是一个关系表达式或逻辑表达式，当“条件表达式”为 true 时，执行后面的“语句”。其执行流程如图 3.2 所示。

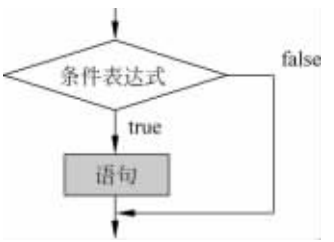


图 3.2　if 语句的执行流程

【练一练】 在“D:\C#和数据库\CH3”文件夹中创建一个 Proj1 控制台项目，说明 if 语句的应用。

其步骤如下。

(1) 创建一个控制台应用程序项目 Proj1，对应的位置为“D:\C#和数据库\CH3”文件夹。

(2) 该项目对应的 C#代码如下。

```
using System;
namespace Proj1
{    class Program
    {    static void Main(string[] args)
        {    int n;
            Console.Write("分数:");
            n = int.Parse(Console.ReadLine());
            if (n < 0 || n > 100)
                Console.WriteLine("分数输入错误");
        }
    }
}
```

上述程序用 if 语句判断用户输入的分数是否在 0～100 之间，否则输入错误。

注意：与 C/C++语言不同，C#条件表达式必须返回 bool 型值，数字在 C#中没有 bool 意义。不要将 if (x==1)错误地书写为 if (x=1)，将 if (x==1 || x==2)错误地书写为 if (x==1 || 2)，这样会出现编译错误。

3.2.2　if…else 语句

如果希望 if 语句在“条件表达式”为 true 和为 false 时分别执行不同的语句，则用 else 来引入“条件表达式”为 false 时执行的语句序列，这就是 if…else 语句，它根据不同的条件分别执行不同的语句序列，其语法形式如下。

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

其中的“条件表达式”是一个关系表达式或逻辑表达式。当“条件表达式”为 true 时执行“语句 1”；当“条件表达式”为 false 时执行“语句 2”。其执行流程如图 3.3 所示。

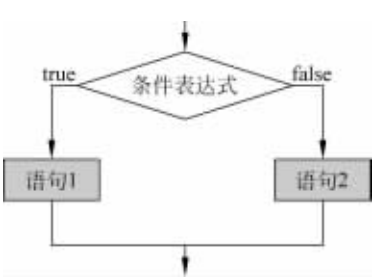


图 3.3　if…else 语句执行流程

if…else 语句可以嵌套使用。当多个 if…else 语句嵌套时，else 与哪个 if 匹配呢？为解决语义上的这种二义性，C#中规定，else 总是与最后一个出现的还没有 else 与之匹配的 if 匹配。

例如，可以将前面 Proj1 项目改为如下代码以判断用户输入正确与否的情况。

```
int n;
Console.Write("分数:");
n = int.Parse(Console.ReadLine());
if (n < 0 || n > 100)
    Console.WriteLine("分数输入错误");
else
    Console.WriteLine("分数输入正确");
```

在执行的语句十分简单的情况下，if…else 语句可以用“?:”运算符代替。例如，求 x 的绝对值可以使用以下语句。

```
x = (x<0)? -x:x;
```

3.2.3 if…else if 语句

if…else if 语句用于进行多重判断，其语法形式如下。

```
if (条件表达式 1) 语句 1;
else if (条件表达式 2) 语句 2;
⋮
else if(条件表达式 n) 语句 n;
else  语句 n + 1;
```

该语句功能是先计算“条件表达式 1”的值。如果为 true，则执行“语句 l”，执行完毕后跳出该 if…else if 语句；如果“条件表达式 1”的值为 false，则继续计算“条件表达式 2”的值。如果“条件表达式 2”的值为 true，则执行“语句 2”，执行完毕后跳出该 if…else if 语句；如果“条件表达式 2”的值为 false，则继续计算“条件表达式 3”的值，以此类推。如果所有条件中给出的表达式值都为 false，则执行 else 后面的“语句 $n+1$”。如果没有 else，则什么也不做，转到该 if…else if 语句后面的语句继续执行。其执行流程如图 3.4 所示。

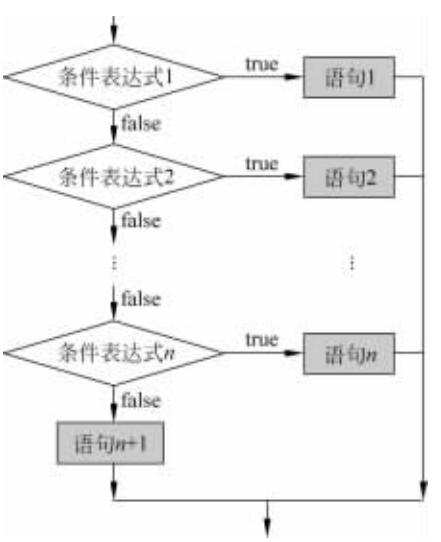


图 3.4 if…else if 语句执行流程

【练一练】 在“D:\C#和数据库\CH3”文件夹中创建一个 Proj2 控制台项目，说明 if…else if 语句的应用。

其步骤如下。

(1) 创建一个控制台应用程序项目 Proj2，对应的位置为“D:\C#和数据库\CH3”文件夹。

(2) 该项目对应的 C#代码如下。

```
using System;
namespace Proj2
{   class Program
    {   static void Main(string[] args)
        {   int a, b, c, tmp, x;
            Console.Write("a:");
            a = int.Parse(Console.ReadLine());
            Console.Write("b:");
            b = int.Parse(Console.ReadLine());
            Console.Write("c:");
            c = int.Parse(Console.ReadLine());
            if (a < b)                  //a 和 b 交换
            {
                tmp = a; a = b; b = tmp;
            }
            //以下总是 a≥b
            if (b > c)                  //b > c,则从大到小为 a,b,c
                x = b;
            else if (a > c)             //b < c < a,则从大到小为 a,c,b
                x = c;
            else                        //b < c,a < c,则从大到小为从 c,a,b
                x = a;
            Console.WriteLine("中间数是{0}",x);
        }
    }
}
```

图 3.5　Proj2 项目的执行结果

上述程序用 if…else if 语句求用户输入的三个整数的中间数。

(3) 按 Ctrl+F5 键执行本项目,其执行结果如图 3.5 所示。

3.2.4　switch 语句

switch 语句也称为开关语句,用于有多重选择的场合测试某一个变量具有多个值时所执行的动作。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 1;break;
    case 常量表达式 2: 语句 2;break;
    ⋮
    case 常量表达式 n: 语句 n;break;
    default:语句 n + 1;break;
}
```

switch 语句控制传递给与“控制表达式”值匹配的 case 块(标签)。从选定的语句开始执行,直到 break 语句将控制传递到 case 块以外。在每一个 case 块(包括 default 块)的后面,都必须有一个跳转语句(如 break 语句),因为 C#不支持从一个 case 块显式贯穿到另一个 case 块。

但有一个例外,当某个 case 语句中没有代码时,可以不包含 break 语句,这种情况通常用

于一次判断多个条件，如果满足这些条件中的任何一个，就会执行后面的代码。例如：

```
case 1:
case 2:
      Console.WriteLine("1 或 2");break;
```

如果没有任何 case 表达式与开关值匹配，则控制传递给跟在可选 default 块后的语句。如果没有 default 块，则控制传递到 switch 语句以外。

switch 语句的执行流程如图 3.6 所示。

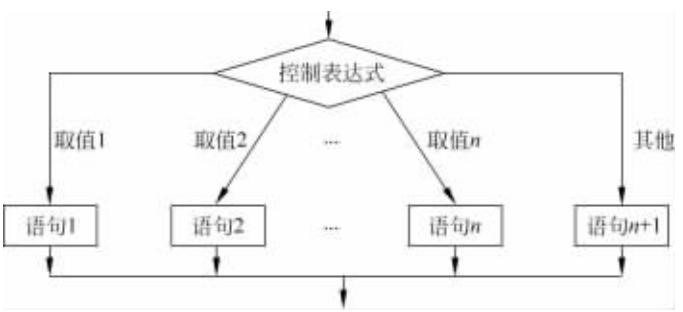


图 3.6 带 break 语句的 switch 控制流程

有关 switch 语句的要点如下。

(1) switch 中的“控制表达式”和 case 中的“常量表达式”必须是 bool、char、string、整型、枚举或相应的可以为 null 的类型。

(2) switch 语句可以包括任意数目的 case 块，但是任何两个 case 块都不能具有相同的“常量表达式”值。

(3) C/C++中 switch 执行完一个 case 的语句后，可以继续执行下一个 case 的语句，而 C#的 switch 不能这样。在 C#中只有当某个 case 语句中没有代码时，可以不包含 break 语句而贯穿到下一个 case 块。

【练一练】 在“D:\C#和数据库\CH3”文件夹中创建一个 Proj3 控制台项目，说明 switch 语句的应用。

其步骤如下。

(1) 创建一个控制台应用程序项目 Proj3，对应的位置为“D:\C#和数据库\CH3”文件夹。

(2) 该项目对应的 C#代码如下。

```
using System;
namespace Proj3
{   class Program
    {   static void Main(string[] args)
        {   DateTime d = DateTime.Now;
            int n = (int)d.DayOfWeek;
            switch(n)
            {
                case 1:
                case 2:
                case 3:
```

```
                case 4:
                case 5:
                    Console.WriteLine("{0}年{1}月{2}日我工作",
                      d.Year,d.Month,d.Day);
                    break;
                case 0:
                case 6:
                    Console.WriteLine("{0}年{1}月{2}日我休息",
                      d.Year,d.Month,d.Day);
                    break;
            }
        }
    }
}
```

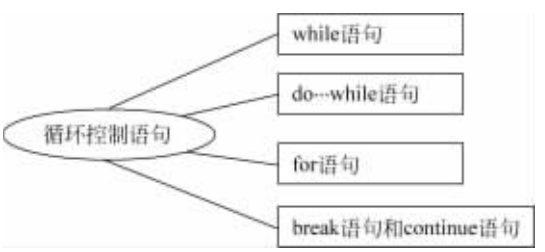
上述代码根据当日的星期号确定是工作还是休息。

(3) 按 Ctrl＋F5 键执行本项目,其一次执行结果如图 3.7 所示。

图 3.7　Proj3 项目的执行结果

3.3　循环控制语句

知识梳理



3.3.1　while 语句

循环控制语句提供重复处理的能力,当某一指定条件为 true 时,循环体内的语句就重复执行,并且每循环一次,就会测试一下循环条件,如果为 false,则循环结束,否则继续循环。C#支持三种格式的循环控制语句:while、do…while 和 for 语句。三者可以完成类似的功能,不同的是它们控制循环的方式。

while 语句的一般语法格式如下。

```
while(条件表达式) 语句;
```

当“条件表达式”的运算结果为 true 时,重复执行“语句”。每执行一次“语句”后,就会重新计算一次“条件表达式”,当该表达式的值为 false 时,while 循环结束。其执行流程如图 3.8 所示。

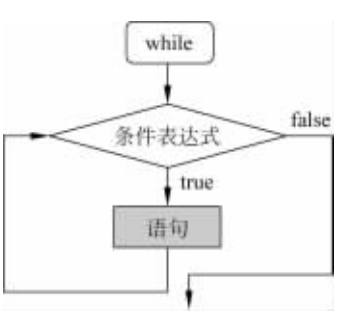


图 3.8　while 语句的执行流程

【练一练】 在“D:\C#和数据库\CH3”文件夹中创建一个 Proj4 控制台项目，说明 while 语句的应用。

其步骤如下。

(1) 创建一个控制台应用程序项目 Proj4，对应的位置为“D:\C#和数据库\CH3”文件夹。

(2) 该项目对应的 C#代码如下。

```
using System;
namespace Proj4
{   class Program
    {   static void Main(string[] args)
        {   int n;
            Console.Write("n:");
            n = int.Parse(Console.ReadLine());
            int total = 0, i = 1, j, m;
            while (i <= n)
            {   m = 1; j = 1;
                while (j <= i)
                {   m *= j;
                    j++;
                }
                total += m;
                i++;
            }
            Console.WriteLine("计算结果 = {0}",total);
        }
    }
}
```

上述代码由用户输入一个正整数 n，采用两重循环求 total=1!+2!+3!+…+n! 的值。

(3) 按 Ctrl+F5 键执行本项目，其执行结果如图 3.9 所示。实际上，如下代码是本例的优化设计，它仅使用了一重循环。

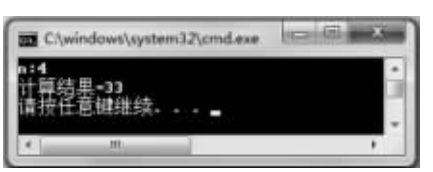


图 3.9 Proj4 项目的执行结果

```
static void Main(string[] args)
{   int n;
    Console.Write("n:");
    n = int.Parse(Console.ReadLine());
    int total = 0, i = 1, m = 1;
    while (i <= n)
    {   m *= i;
        total += m;
        i++;
    }
    Console.WriteLine("计算结果 = {0}",total);
}
```

3.3.2　do…while 语句

do…while 语句的一般语法格式如下。

```
do
    语句;
while (条件表达式);
```

do…while 语句每一次循环执行一次“语句”，就计算一次“条件表达式”是否为 true，如果是，则继续执行循环，否则结束循环。与 while 语句不同的是，do…while 循环中的“语句”至少会执行一次，而 while 语句当条件第一次就不满足时，语句一次也不会执行。其执行流程如图 3.10 所示。

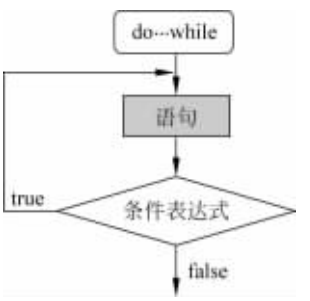


图 3.10　do…while 语句的执行流程

例如，前面的 Proj4 项目可以采用 do…while 语句编写如下。

```
static void Main(string[] args)
{   int n;
    Console.Write("n:");
    n = int.Parse(Console.ReadLine());
    int total = 0, i = 1, m = 1;
    do
    {   m * = i;
        total += m;
        i++;
    } while (i < = n);
    Console.WriteLine("计算结果 = {0}", total);
}
```

3.3.3　for 语句

for 语句通常用于预先知道循环次数的情况，其一般语法格式如下。

```
for (表达式 1;表达式 2;表达式 3)语句;
```

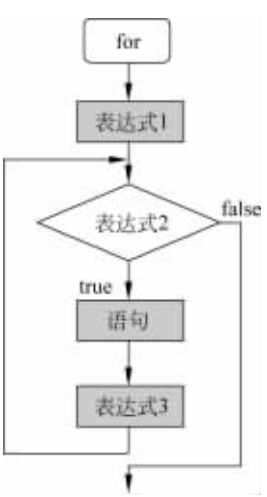


图 3.11　for 循环语句的执行流程

其中，“表达式 1”可以是一个初始化语句，一般用于对一组变量进行初始化或赋值。“表达式 2”用于循环的条件控制，它是一个条件或逻辑表达式，当其值为 true 时，继续下一次循环，当其值为 false 时，则终止循环。“表达式 3”在每次循环执行完后执行，一般用于改变控制循环的变量。“语句”在“表达式 2”为 true 时执行。具体来说，for 循环的执行过程如下。

(1) 执行“表达式 1”。

(2) 计算“表达式 2”的值。

(3) 如果“表达式 2”的值为 true，先执行后面的“语句”，再执行“表达式 3”，然后转向步骤(1)；如果“表达式 2”的值为 false，则结束整个 for 循环。

for 语句的执行流程如图 3.11 所示。需要注意的是，在

for 语句中，三个表达式均可以省略，例如，求 1＋2＋…＋n 值时可以使用如下方式。

方式 1：

```
for (int sum = 0,i = 1;i <= n;i++)
    sum += i;
```

方式 2：

```
int sum = 0,i = 1;
for (;i <= n;i++)
    sum += i;
```

方式 3：

```
int sum = 0,i = 1;
for (;i <= n;)
{   sum += i;
    i++;
}
```

另外，C#还提供了与 for 语句功能类似的 foreach 循环语句，用来循环处理一个集合中的元素，这将在第 4 章中介绍。

注意，在 for 语句内定义的变量，其作用域仅限于该 for 语句。例如：

```
for (int i = 0;i<10;i++)              //i 的作用域仅限于第一个 for 语句
    语句;
//前面 for 语句中定义的变量 i 已超出作用域
for (int i = 0;i<10;i++)              //需要定义一个新的变量 i
    语句;
```

例如，前面的 Proj4 项目可以采用 for 语句编写如下。

```
static void Main(string[] args)
{   int n;
    Console.Write("n:");
    n = int.Parse(Console.ReadLine());
    int total = 0, i, m = 1;
    for (i = 1; i <= n; i++)
    {   m *= i;
        total += m;
    }
    Console.WriteLine("计算结果 = {0}", total);
}
```

3.3.4 break 语句和 continue 语句

1. break 语句

break 语句使程序从当前的循环语句(do、while 和 for)内跳转出来，接着执行循环语句后面的语句。break 语句还可以用于从 switch 语句中跳出，接着执行 switch 语句后面的语句。

【练一练】 在“D:\C#和数据库\CH3”文件夹中创建一个 Proj5 控制台项目，说明 break 语句在 for 语句中的应用。

其步骤如下。

（1）创建一个控制台应用程序项目 Proj5，对应的位置为“D:\C＃和数据库\CH3”文件夹。

（2）该项目对应的 C＃代码如下。

```
using System;
namespace Proj5
{   class Program
    {   static void Main(string[] args)
        {   int n, i;
            bool prime = true;
            Console.Write("输入一个大于 3 的正整数:");
            n = int.Parse(Console.ReadLine());
            for (i = 2; i <= Math.Sqrt(n); i++)
                if (n % i == 0)
                {   prime = false;
                    break;          //提出 for 循环语句
                }
            if (prime) Console.WriteLine("{0}是素数", n);
            else Console.WriteLine("{0}不是素数", n);
        }
    }
}
```

上述程序用于判断从键盘输入的大于 3 的正整数 n 是否为素数。采用 for 循环语句，当 n 能被 $2\sim\sqrt{n}$ 中的任何整数整除时，用 break 语句退出循环，表示 n 不是素数，否则 n 为素数。

（3）按 Ctrl＋F5 键执行本项目，其一次执行结果如图 3.12 所示。

图 3.12　Proj5 项目的执行结果

2．continue 语句

continue 语句只能用于循环语句，它不同于 break 语句，不是直接结束整个循环，而是结束循环语句的当前一次循环，接着执行下一次循环。在 while 和 do…while 循环语句中，控制权转至对“条件表达式”的判断，若为 true，则继续执行下一次循环；在 for 循环语句中，转去执行“表达式 2”，若为 true，则继续执行下一次循环。

【练一练】　在“D:\C＃和数据库\CH3”文件夹中创建一个 Proj6 控制台项目，说明 continue 语句在 for 语句中的应用。

其步骤如下。

（1）创建一个控制台应用程序项目 Proj6，对应的位置为“D:\C＃和数据库\CH3”文件夹。

（2）该项目对应的 C＃代码如下。

```
using System;
namespace Proj6
{   class Program
    {   static void Main(string[] args)
```

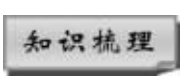
```
        {   int n,i;
            bool prime;
            for (n = 2; n <= 20; n++)
            {   prime = true;
                for (i = 2; i <= Math.Sqrt(n); i++)
                    if (n % i == 0)
                    {   prime = false;
                        break;
                    }
                if (!prime) continue;      //进入 for 的下一轮循环
                Console.Write("{0} ", n);
            }
            Console.WriteLine();
        }
    }
}
```

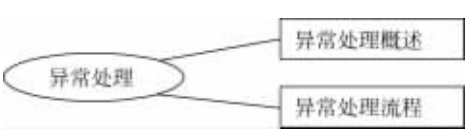
上述代码用于输出 20 以下的所有素数。用 n 进行循环，当 n 不是素数时，执行 continue 语句转入下一轮循环，执行 n++，继续对 n 进行判断。

(3) 按 Ctrl+F5 键执行本项目，其执行结果如图 3.13 所示。

图 3.13 Proj6 项目的执行结果

3.4 异常处理





3.4.1 异常处理概述

任何完美的应用程序，都不可能是绝对不出差错的。与其追求完美无错的代码，还不如在程序中对可能预知的异常进行处理。C#语言提供了异常处理机制。

try 语句块用于检查发生的异常，并帮助发送任何可能的异常，程序员将可能出现异常的语句包含在 try 语句中。其基本格式如下。

```
try
{
    //程序代码块
}
```

catch 语句块用于检测异常，并以控制权更大的方式处理错误，一个 try 语句块可以有多个 catch 子句。其基本格式如下。

```
catch(Exception e)
{
    //异常处理代码块
}
```

finally 语句块无论是否引发了异常，finally 的代码都将被执行。其基本格式如下。

```
finally
{
    //无论是否发生异常，均要执行的代码块
}
```

任何异常处理必须包含一个 try 语句块和若干 catch 语句块（至少一个），finally 语句块是可选的。

3.4.2　异常处理流程

一旦 try 语句块中的语句发生异常，控制流将跳转到第一个关联的异常处理程序，在 C# 中，用 catch 语句定义异常处理程序。如果出现的异常没有相应的异常处理程序，则程序将停止执行，并显示一条错误消息。

异常处理流程如图 3.14 所示。

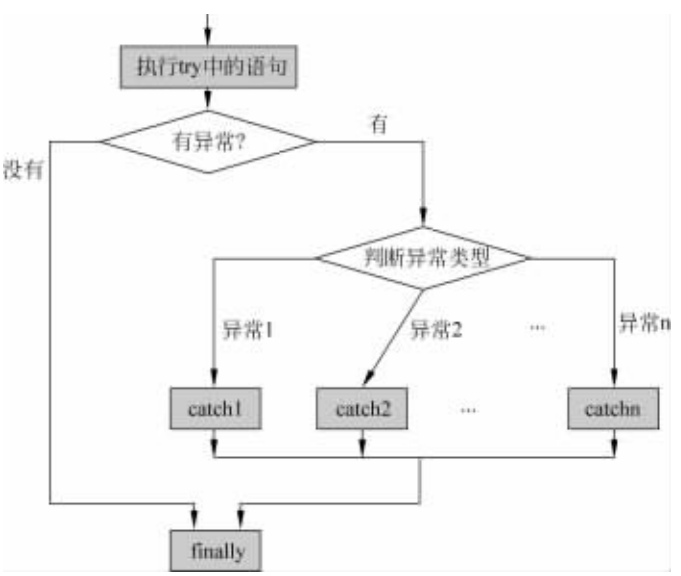


图 3.14　异常处理流程

异常类型是通过相应的异常类来捕获的，如 DivideByZeroException 异常类捕获试图用零除整数值或十进制数值时引发的异常，IndexOutOfRangeException 捕获类在一个数组的下标超出范围时引发。

所有的异常类都是由 System.Exception 异常类派生的，该类主要的属性有 Message 和 Source，前者用于获取描述当前异常的消息，后者用于表示异常发生的来源对象。

另外，程序员也可以使用 throw 关键字自己抛出一个异常。这里不再深入介绍。

【练一练】　在“D:\C#和数据库\CH3”文件夹中创建一个 Proj7 控制台项目，说明异常

处理语句的使用方法。

其步骤如下。

(1) 创建一个控制台应用程序项目 Proj7,对应的位置为“D:\C#和数据库\CH3”文件夹。

(2) 该项目对应的 C#代码如下。

```
using System;
namespace Proj7
{   class Program
    {   static void Main(string[] args)
        {   int z = 0;
            try
            {   int x, y;
                Console.Write("被除数:");
                x = int.Parse(Console.ReadLine());
                Console.Write("除数:");
                y = int.Parse(Console.ReadLine());
                z = x / y;
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
            finally
            {   if (z == 0)
                    Console.WriteLine("异常处理完毕");
                else
                    Console.WriteLine("除法结果:{0}", z);
            }
        }
    }
}
```

上述程序从键盘获取用户输入的两个整数 x 和 y,执行 x/y 运算,并处理除零处理。

(3) 按 Ctrl+F5 键执行本项目,用户分别输入 20 和 5,此时没有出现异常,z=4,不会执行 catch 语句,但一定要执行 finally 语句,显示结果为 4,如图 3.15 所示。

如果用户分别输入 20 和 0,此时出现异常,z 的值没有改变,执行 catch 语句输出“尝试除以零”的异常信息,再执行 finally 语句,输出“异常处理完毕”,如图 3.16 所示。

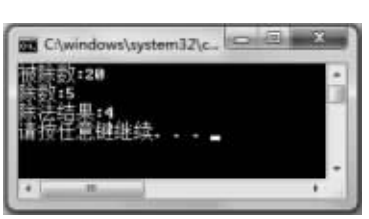


图 3.15 Proj7 项目的执行结果一

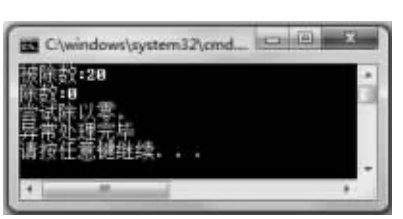
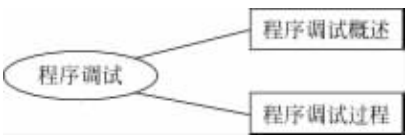


图 3.16 Proj7 项目的执行结果二

3.5　程 序 调 试

知识梳理



3.5.1　程序调试概述

在编写 C# 程序时难免有错，除了采用异常处理外，应该尽可能地发现程序中的错误。

1. 程序错误

程序错误主要分成两类，即语法错误和逻辑错误。

1）语法错误

语法错误也称为编译错误，是由于不正确地编写代码而产生的。如果错误地输入了关键字（例如，将 int 写为 Int）、遗漏了某些必需的语句成分等，那么 C# 在编译应用程序时就会检测到这些错误，并提示相应的错误信息。

例如，TMP 项目中将 int 误写为 Int，系统会在“错误列表”窗口中提示如下消息。

```
错误 1 未能找到类型或命名空间名称"Int"(是否缺少 using 指令或程序集引用?)
D:\C# 和数据库\CH3\TMP\TMP\Program.cs 13 13 TMP
```

该消息表示 TMP 项目中的第 1 个错误出现在第 13 行第 13 列，即使用了错误的类型 Int。而 Int 有可能是开发人员自定义的类型，如果是这样，需要使用 using 语句引用包含 Int 类型声明的命名空间。

在“错误列表”窗口中单击该错误提示行，光标会自动跳转到出现出错的代码位置。这类程序错误是十分明显的，并且容易改正。

2）逻辑错误

逻辑错误主要表现在程序执行后，没有提示任何错误信息且能够正常运行，但得到的结果与预期设想的不一致。这有可能是程序设计中出现了逻辑错误，这一类错误属算法设计错误，是最难纠正的，必须使用程序调试工具进行错误排查。

C# 提供了强大的程序调试功能，使用其调试环境可以有效地完成程序的调试工作，从而有助于发现执行错误。

2. 调试工具

C# 中用于查找逻辑错误的工具主要有“调试”工具栏和“调试”菜单。两种具有相同的功效。

1）“调试”工具栏

选择“视图”|“工具栏”|“调试”命令，出现“调试”工具栏，这在第 1 章中已介绍。

2）“调试”菜单

“调试”菜单提供了更完整的程序错误调试命令，如图 3.17 所示。

图 3.17 “调试”菜单

3.5.2 程序调试过程

程序调试过程是,先设置断点,再开始调试过程,在断点处查看调试信息,最后修改错误。

1. 设置断点

程序调试的基础是设置断点,断点是在程序中设置的一个位置(程序行),程序执行到该位置时中断(或暂停)。断点的作用是在调试程序时,当程序执行到断点的语句时会暂停程序的执行,供程序员检查这一位置上程序元素的执行情况,这样有助于定位产生错误输出或出错的代码段。

设置和取消断点的方法如下。

方法 1:用鼠标右键单击某代码行,从出现的快捷菜单中选择“断点”|“插入断点”命令(设置断点)或者“断点”|“删除断点”命令(取消断点)。

方法 2:将光标移至需要设置断点的语句处,然后选择“调试”菜单中的“切换断点”命令或按 F9 键。

设置了断点的代码行最左端会出现一个红色的圆点,并且该代码行也呈现红色背景。

【练一练】 在“D:\C#和数据库\CH3”文件夹中创建一个 Proj8 控制台项目,说明程序的调试过程。

其步骤如下。

(1) 创建一个控制台应用程序项目 Proj8,对应的位置为“D:\C#和数据库\CH3”文件夹。

(2) 该项目对应的 C#代码如下。

```
using System;
namespace Proj8
{   class Program
    {   static void Main(string[] args)
        {   int n = 5;
            int total = 0, i, m;
```

```
            for (i = 1; i <= n; i++)
            {   m = 0;
                m += i;
                total += m;
            }
            Console.WriteLine("计算结果:{0}",total);
        }
    }
}
```

上述代码用于计算 1＋(1＋2)＋(1＋2＋3)＋…＋(1＋2＋3＋…＋n)的值，这里 n＝5。

(3) 按 Ctrl＋F5 键执行本项目，发现计算结果为 15。

(4) 结果显然是错误的，初步判断可能 m 的计算有问题，将光标移到第 14 行，选择“调试”|“切换断点”命令(或按 F9 键)，在第 14 行处设一个断点，如图 3.18 所示。注意可以在一个程序中设置多个断点。

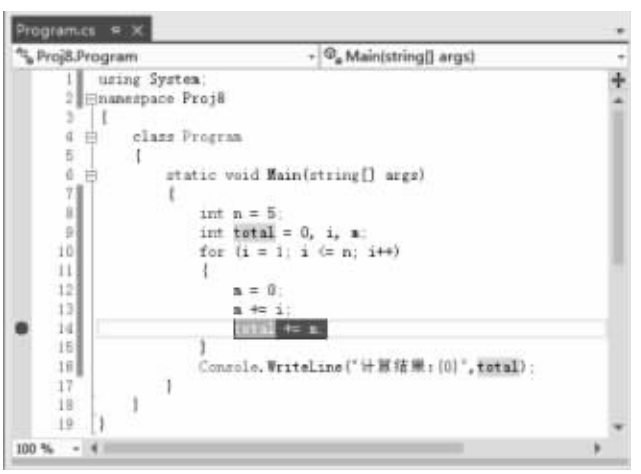


图 3.18　设置一个断点

2. 开始调试过程

在设置断点后，从“调试”菜单中选择“启动调试”、“逐语句”或“逐过程”命令，或者在代码编辑窗口中，单击鼠标右键，然后从快捷菜单中选择“运行到光标处”命令，即开始调试过程。

如果选择“启动调试”命令(或按 F5 键)，则应用程序启动并一直执行到断点。可以在任何时刻中断执行以检查值或检查程序状态。

若选择“逐语句”或“逐过程”命令，应用程序启动并执行，然后在第一行中断。

如果选择“执行到光标处”命令，则应用程序启动并一直执行到断点或光标位置，具体看是断点在前还是光标在前。可以在源窗口中设置光标位置。某些情况下，不出现中断，这意味着执行始终未到达设置光标处的代码。

3. 查看调试信息

在程序调试的中断状态下，可以通过智能感知窗口、即时窗口、局部变量窗口和快速监视窗口来观察变量的值。

接着前面的示例步骤。选择“调试”|“启动调试”命令，程序执行到断点处。

(1) 智能感知窗口。将鼠标放在希望观察的执行过语句的变量上，调试器会通过智能感

知窗口自动显示执行到断点时该变量的值，如图 3.19 所示，看到 m 的值为 1。

（2）即时窗口。此时选择“调试”|“窗口”|“即时”命令，出现“即时窗口”。可以输入“? 变量或表达式”来显示变量或表达式的值。如图 3.20 所示，看到此时 i 和 m 的值都为 1。

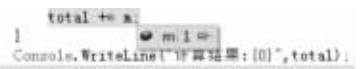


图 3.19 显示变量 m 的值

图 3.20 即时窗口显示 i 和 m 的值

（3）“局部变量”窗口。此时选择“调试”|“窗口”|“局部变量”命令，出现“局部变量”窗口，它自动显示当前过程中所有的变量值，如图 3.21 所示，查看到此时 i 和 m 的值都为 1。

（4）“快速监视”窗口。此时在某个对象上或空白处单击鼠标右键，从弹出的快捷菜单中选择“快速监视”命令，出现“快速监视”窗口，它用于显示用户在“表达式”文本框中输入的表达式的值，如图 3.22 所示，看到此时 m 的值都为 1。

图 3.21 “局部变量”窗口的结果

图 3.22 “快速监视”窗口显示变量 m 的值

在程序开始调试过程后，每次处于中断状态时，用户通过上述窗口观察变量或表达式的值，然后按 F5 键继续，从而跟踪变量或表达式的变化过程，最终找出程序出错的原因。

需要注意的是，“局部变量”窗口和“快速监视”窗口不同于即时窗口，它具有跟踪变量的功能，当按 F5 键时，程序继续执行，此时“局部变量”窗口和“快速监视”窗口的变量值会自动发生相应的改变。所以在程序调试中最常用的是这两个窗口。

接着前面的示例步骤。按 F5 键让程序执行 for 循环的第二次循环，看到“局部变量”窗口的结果如图 3.23 所示，此时 m 为 2。再按 F5 键让程序执行 for 循环的第三次循环，看到“局部变量”窗口的结果如图 3.24 所示，此时 m 为 3。

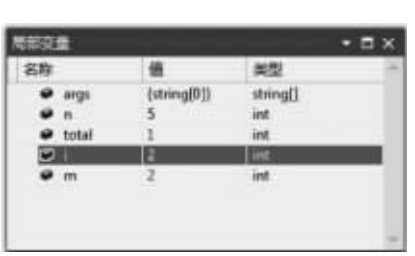


图 3.23 第二次循环时“局部变量”窗口的结果

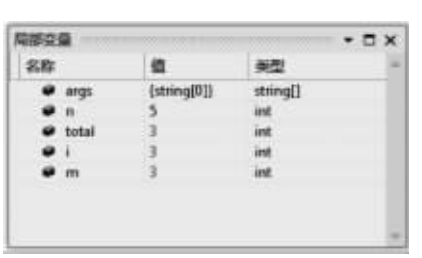


图 3.24 第三次循环时“局部变量”窗口的结果

看出错误出现在 m 的计算上，因为第二次循环 m 值应该为 3(1＋2)，第三次循环 m 值应该为 6(1＋2＋3)。

通过分析，发现 m＝0 置初值语句应该放在 for 循环外，将程序改为：

```
static void Main(string[] args)
{   int n = 5;
    int total = 0, i, m;
    m = 0;
    for (i = 1; i <= n; i++)
    {   m += i;
        total += m;
    }
    Console.WriteLine("计算结果:{0}",total);
}
```

再次执行程序，计算结果为 35，程序正确。

练　习　题

1. 单项选择题

(1) if 语句后面的表达式应该是(　　)。

A. 字符串表达式　　B. 条件表达式　　C. 算术表达式　　D. 任意表达式

(2) 执行下列语句序列后，i 和 j 的值分别是(　　)。

```
int i = 3, j = 5;
if (i-1>j)
    i--;
else
    j--;
```

A. 2，4　　B. 2，5　　C. 3，4　　D. 3，5

(3) 以下关于 if 语句和 switch 语句的说法，正确的是(　　)。

A. 如果在 if 语句和 switch 语句中嵌入 break 语句，则在程序执行过程中，一旦执行到 break 语句，就会结束相应的执行，转向执行其后面的语句

B. 凡是能够使用 if 语句的地方就可以使用 switch 语句，反之亦然

C. if 语句有三种基本形式：if…、if…else…和 if…else if…else…

D. 以上都是正确的

(4) 以下代码执行后，s 的值是(　　)。

```
int s = 0;
for (int i = 1; i<100; i++)
{   if (s>10)
        break;
    if (i % 2 == 0)
        s += i;
}
```

A. 20 B. 12 C. 10 D. 6

(5) 以下代码的输出结果是(　　)。

```
int x = 35, y = 80;
if (x < -10 || x > 30)
{   if (y >= 100)
        Console.WriteLine("危险 ");
    else
        Console.WriteLine("报警 ");
}
else
    Console.WriteLine("安全 ");
```

A. 危险 B. 报警 C. 报警安全 D. 危险安全

(6) 以下代码的输出结果是(　　)。

```
int month = 6;
int days;
switch (month)
{   case 2:days = 28; break;
    case 4:
    case 6:
    case 9:
    case 11:days = 30; break;
    default:days = 31; break;
}
Console.WriteLine(days);
```

A. 0 B. 28 C. 30 D. 31

(7) 在C#语言中,switch 语句用(　　)来处理不匹配 case 语句的值。

A. default B. anyelse C. break D. goto

(8) 以下代码的输出结果是(　　)。

```
for (int i = 0; i < 5; i++)
{
    Console.Write(++i);
}
```

A. 12345 B. 135 C. 24 D. 没有任何输出

(9) 以下代码的输出结果是(　　)。

```
for (int i = 6; i > 0; i -- )
{
    Console.Write(i -- );
}
```

A. 123456 B. 246 C. 642 D. 654321

(10) 关于如下程序结构的描述中,(　　)是正确的。

```
for(;;)
{  循环体;  }
```

A. 不执行循环体　　B. 一直执行循环体，即死循环

C. 执行循环体一次　　D. 程序不符合语法要求

(11) 以下 for 循环的执行次数是(　　)。

```
for(int i = 1;(i == 1)&(i > 2);i++);
```

A. 无限次　　B. 一次也不执行　　C. 执行一次　　D. 执行两次

(12) 以下叙述正确的是(　　)。

A. do…while 语句构成的循环不能用其他语句构成的循环来代替

B. do…while 语句构成的循环只能用 break 语句退出

C. 用 do…while 语句构成的循环，在 while 后的表达式为 true 时结束循环

D. 用 do…while 语句构成的循环，在 while 后的表达式应为关系表达式或逻辑表达式

(13) 以下由 do…while 语句构成的循环执行次数是(　　)。

```
int m = 1;
do
{
    ++m;
} while(m < 1);
```

A. 有语法错误，不能执行　　B. 一次也不执行

C. 执行一次　　D. 执行两次

(14) 以下有关 for 循环的正确描述是(　　)。

A. for 循环只能用于循环次数已经确定的情况

B. for 循环是先执行循环体语句，后判断表达式

C. 在 for 循环中，不能用 break 语句跳出循环体

D. for 循环的循环体语句中，可以包含多条语句，但必须用花括号括起来

(15) 对于 for(表达式 1；；表达式 3)可理解为(　　)。

A. for(表达式 1;false;表达式 3)　　B. for(表达式 1;表达式 3;表达式 3)

C. for(表达式 1;true;表达式 3)　　D. for(表达式 1;表达式 1;表达式 3)

(16) C#程序中，可使用 try…catch 机制来处理程序出现的(　　)错误。

A. 语法　　B. 运行　　C. 逻辑　　D. 拼写

(17) 在 C#程序中，以下用来处理异常的结构，错误的是(　　)。

A. catch{…} finally{…}　　B. try{… } finally{…}

C. try{… } catch{… }finally{… }　　D. try{… } catch{… }

(18) 以下对异常说法不正确的是(　　)。

A. try…catch 块为基本引发异常的组合

B. 在捕获异常时，可以有多个 catch 块

C. 无论异常是否发生，finally 块总会执行

D. try 块和 finally 不能连用

(19) 为了能够在程序中捕获所有的异常，在 catch 语句的括号中使用的类名为(　　)。

A. Exception　　B. DivideByZeroException

C. FormatException　　D. 以上三个均可

(20) 在调试程序时,执行到断点处,不能通过(　　)查看变量的值。

A. 即时窗口　　B. “局部变量”窗口

C. “快速监视”窗口　　D. “属性”窗口

2. 问答题

(1) 简述 if 和 switch 语句的异同。

(2) 简述 C#中 switch 语句与 C/C++中 switch 语句的不同点。

(3) 以下代码存在什么错误?

```
int i;
for (i = 1;i <= 10;i++)
{   if ((i % 2) == 0)
        continue;
    Console.WriteLine(i);
}
```

(4) 给出以下代码的输出结果。

```
int n = 9;
if (n++< 10)
    Console.WriteLine(n);
else
    Console.WriteLine("{0}",n--);
```

(5) 给出以下代码的输出结果。

```
int a = 0, i;
for (i = 1; i < 5; i++)
{   switch (i)
    {
        case 0:
        case 3: a += 2; break;
        case 1:
        case 2: a += 3; break;
        default: a += 5; break;
    }
}
Console.WriteLine(a);
```

(6) 给出以下代码的输出结果。

```
int i = 0, s = 0;
do
{   if ((i % 2) == 1)
    {
        i++; continue;
    }
    i++;
    s += i;
} while (i < 7);
Console.WriteLine(s);
```

（7）给出以下代码的输出结果。

```
int i = 0, a = 0;
while (i < 20)
{   for (; ; )
    {   if (i % 10 == 0)
            break;
        else
            i -- ;
    }
    i += 11;
    a += i;
}
Console.WriteLine(a);
```

（8）给出以下代码的输出结果。

```
int num = 14682;
int k = 1;
do
{   k * = num % 10;
    num / = 10;
} while (num!= 0);
Console.WriteLine(k);
```

（9）给出以下代码的输出结果。

```
int i = 0, s = 0;
for (; ; )
{   i += 2;
    if (i > 6)
    {   Console.WriteLine(s);
        break;
    }
    if (i == 6) continue;
    s += i;
}
```

（10）给出以下代码的输出结果。

```
int i = 20, n = 0;
do
{   n++;
    switch (i % 4)
    {
        case 0: i = i - 7;break;
        case 1:
        case 2:
        case 3: i++;break;
    }
} while (i >= 0);
Console.WriteLine(n);
```

（11）简述 try… catch…finally 的执行流程。

（12）简述 C＃程序的调试过程。

上机实验题

在 CH3 文件夹中创建一个名称为 Exp 的控制台应用程序，求解百鸡问题：鸡翁一，值钱五，鸡母一，值钱三，鸡雏三，值钱一，百钱买百鸡，问鸡翁、鸡母和鸡雏各多少？Exp 的执行结果如图 3.25 所示。

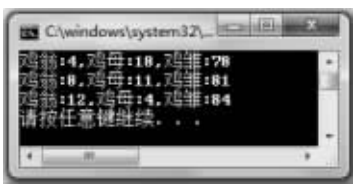


图 3.25　Exp 的执行结果