项目3 设计流程控制程序

扫码答题

项目情境

小张到一家公司面试销售代表，该公司员工的基本工资根据学历层次确定。本科生4500元，硕士研究生6500元，博士生8500元。奖金是月销售额的1%。如何编程计算员工的工资总额呢？

学习重点与难点

- 掌握if选择结构的使用
- 掌握条件运算符的使用
- 掌握switch多分支结构的使用
- 掌握while()、do…while()、for()循环结构的使用
- 理解多重循环结构
- 掌握Foreach()循环的使用

学习目标

- 学会使用if和switch分支结构编写程序
- 学会使用循环结构编写程序
- 学会多重循环结构程序的设计
- 学会Visual C#中的异常处理机制

任务描述

任务1　输入两个数a和b，编写程序使a的值大于b的值

任务2　判断一个数是不是3的倍数

任务3　成绩转换

任务4　采用switch语句实现任务3

任务5　计算景点门票的优惠率

任务6　简单计算器

任务7　输出100以内的所有奇数和、偶数和

任务8　用do…while语句改写任务7

任务9　用for循环改写任务7

任务 10　利用 foreach 统计字符串中各种字符的个数
任务 11　石头、剪刀、布猜拳游戏
任务 12　输出图形
任务 13　输出斐波那契数列的前 20 项
任务 14　输出 1000 以内的完数
任务 15　百钱买百鸡问题的求解

相关知识

知识要点：

- if 选择结构程序设计
- switch 多分支结构程序设计
- while 循环控制语句
- do…while 循环控制语句
- for 循环控制语句
- foreach 循环语句
- 循环的嵌套
- 循环的中断
- 异常处理

在前面的学习中，各语句是按自上而下的顺序执行的，这是顺序结构的程序。而在实际应用中，有时根据是否满足某个条件来决定是否执行指定的操作任务，有时需要从给定的两种或多种操作中选择一种，有时需要重复执行相同的操作。这就是本章要学习的内容——流程控制结构，常用的流程控制结构有三种：顺序结构、选择结构和循环结构。

选择结构是程序设计中常见的基本结构，根据给定的条件进行不同分支的选择。常用的选择结构有 if 条件语句和 switch 分支结构。

知识点 1　if 选择结构程序设计


观看视频

if 条件语句包含多种形式：单分支、双分支和多分支。

1. 单分支结构

```
if(表达式)
  {
      语句块;
  }
```

执行过程：

先计算表达式的值，如果“表达式”的值为真(true)，则执行其后的“语句块”；否则不执行该语句块。

说明：

(1) 条件表达式可以是关系表达式或逻辑表达式，也可以是其他类型的表达式(赋值表达式等)，甚至可以是一个变量或常量。

(2) 语句块可以是单个语句，也可以是多个语句。

单分支结构的流程如图 3-1 所示。

2. 双分支结构

if 语句更常用的形式是双分支语句，一般形式为：

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

执行过程：

如果表达式的值为真(true)，即条件成立，则执行语句块 1；若表达式的值为假(false)，即条件不成立，则执行语句块 2。

说明：

if 语句无论几行，都是一个整体，属于同一个语句。else 子句不能作为语句单独使用，它必须是 if 语句的一部分，与 if 配对使用。

双分支结构的流程如图 3-2 所示。

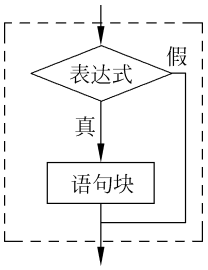

图 3-1　单分支结构

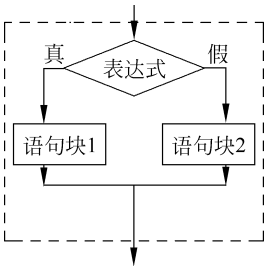

图 3-2　双分支结构

3. 多分支结构

前两种形式的 if 语句一般都用于两个分支的情况。当有多个分支可供选择时，可采用 if-else-if 语句，其一般形式为：

```
if(表达式 1)
   语句块 1;
else if(表达式 2)
   语句块 2;
else if(表达式 3)
   语句块 3;
    …
else if(表达式 n－1)
   语句块 n－1;
else
   语句块 n;
```

多分支语句的执行流程如图 3-3 所示。

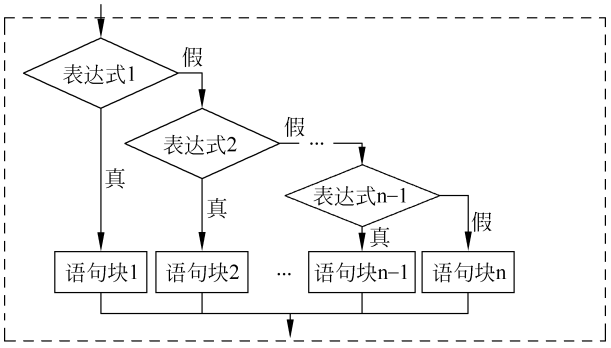

图 3-3　多分支结构

执行过程：

当表达式 1 的值为真(true)时，执行语句块 1，然后跳过整个结构执行下一条语句；当表达式 1 的值为假(false)时，跳过语句块 1 来判断表达式 2。若表达式 2 的值为真(true)，则执行语句块 2，以此类推，当表达式 1、表达式 2、…、表达式 n－1 全为假(false)时，将执行 else 后面的语句块 n，然后继续执行后续程序。

知识点 2　switch 多分支结构程序设计


观看视频

当判断的条件有多个时，如果使用 else if 语句，则会使程序变得难以阅读。而且有的时候比较的是具体的离散值，而不是像“大于 x”这样的语句。例如等级转换程序：成绩在 90 分以上为“优秀”，80～89 分为“良好”，70～79 分为“中等”，60～69 分为“及格”，60 分以下为“不及格”。

switch 语句是多分支语句。switch 语句的一般格式如下：

```
switch(表达式)
{
 case 常量表达式 1:
      语句块 1;
      break;
 case 常量表达式 2:
      语句块 2;
      break;
 …
 case 常量表达式 n:
      语句块 n;
      break;
 default:
      语句块 n+1;
      break;
}
```

执行过程：

先计算表达式的值，并逐个与其后常量表达式的值相比较，当表达式的值与某个常量表达式的值相等时，即执行其后的语句，然后通过 break 语句退出 switch 结构，执行位于整个 switch 结构后面的语句。如果没有 break 语句，流程控制转移到下一个 case 语句继续执

行。如果表达式的值与 case 语句中的常量表达式的值均不相同，则执行 default 后面的语句。

说明：

（1）switch 下面的花括号是不能省略的，花括号下面的语句是 switch 语句的语句体。语句体内包含多个以关键字 case 开头的语句行和最多一个以 default 开头的行。case 后面跟一个常量（或常量表达式）。

（2）switch 后面的表达式所允许的数据类型包括整数类型、字符类型、字符串类型和枚举类型。各个 case 分支后的常量表达式的数据类型必须与表达式的类型相同或者能够隐式地转换为所允许的数据类型。

（3）各个 case 出现的次序不影响执行结果。

（4）每一个 case 常量必须互不相同。

（5）多个 case 可以共用一组执行语句。

switch 多分支结构的流程图如图 3-4 所示。

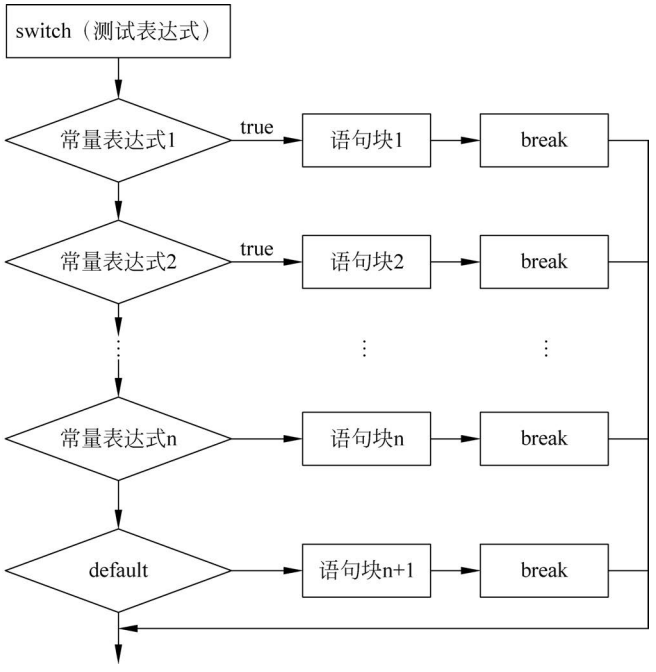

图 3-4　switch 多分支结构的流程图

注意：在 C# 中，switch 语句的一个有趣的地方是 case 子句的顺序无关紧要，甚至可以把 default 子句放在最前面。因此，任意两个 case 都不能相同，包括值相同的不同常量，例如：

```
const string eng = "en";
const string br = "en";
string country;
string language;
country = Console.ReadLine();
switch(country)
{
```

```
            case eng:
                break;
            case br:
                language = "english";
                break;
            }
```

以上代码存在如图 3-5 所示的错误。

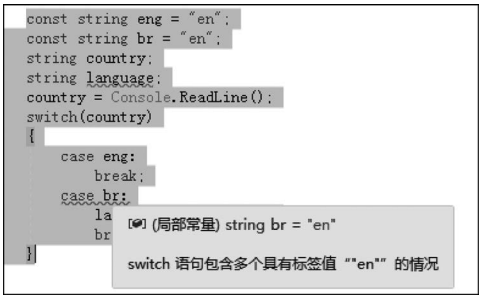

图 3-5　值相同时示例

在程序处理中常常遇到需要重复处理的问题。例如，要向计算机输入全班 50 名学生的成绩，需要重复 50 次相同的输入操作；检查 30 名学生的成绩是否及格，需要重复 30 次相同的判断操作。

要处理以上问题，最原始的方法是分别编写若干相同或相似的语句或程序段进行处理。

例如，我们要求全班 30 名学生的平均成绩，可以先编写求一名学生的平均成绩的程序段：

```
float score1, score2, score3, score4, score5,aver;
score1 = float .Parse (Console.ReadLine());   //以下输入一名学生的 5 门课的成绩
score2 = float.Parse(Console.ReadLine());
score3 = float.Parse(Console.ReadLine());
score4 = float.Parse(Console.ReadLine());
score5 = float.Parse(Console.ReadLine());
aver = (score1 + score2 + score3 + score4 + score5) / 5;       //求平均成绩
 Console.WriteLine("aver = {0}", aver);                        //输出平均成绩
```

然后重复写 29 个同样的程序段。这种方法虽然可以实现要求，但是存在工作量大、程序冗长、重复、难以阅读和维护等问题。实际上，每一种计算机语言都提供了用来处理重复操作的语句，这就是循环控制。

在 C＃中，可以用循环语句来处理上面的问题：

```
float score1, score2, score3, score4, score5,aver;
int i = 1;
while(i <= 30)
    {
       score1 = float .Parse (Console.ReadLine());
       score2 = float.Parse(Console.ReadLine());
       score3 = float.Parse(Console.ReadLine());
       score4 = float.Parse(Console.ReadLine());
       score5 = float.Parse(Console.ReadLine());
       aver = (score1 + score2 + score3 + score4 + score5) / 5;
       Console.WriteLine("第{0}个人的平均成绩:aver = {1}",i, aver);
        i++;
    }
```

可以看到，通过一个循环语句，成功解决了需要重复执行 30 次的程序段的问题。

C＃中的循环结构主要有三种：while()、do…while()和 for()。

知识点 3　while 循环控制语句

while 循环也称当型循环，一般形式如下：

```
while(表达式)
{
语句;
}
```

观看视频

其中，“表达式”是循环条件，“语句”为循环体。循环体只能是一个语句，可以是一个简单的语句，也可以是复合语句（用花括号包起来的若干语句）。

执行过程：计算表达式的值，当值为真（非 0）时，执行循环体语句，当值为“假”（0）时，不执行循环体语句。其流程图如图 3-6 所示。

知识点 4　do…while 循环控制语句

除 while 语句外，C 语言还提供了 do…while 语句来实现循环结构。

do…while 语句的一般形式为：

```
do
语句
while(表达式);
```

其中的“语句”就是循环体。它的执行过程可以用图 3-7 表示。

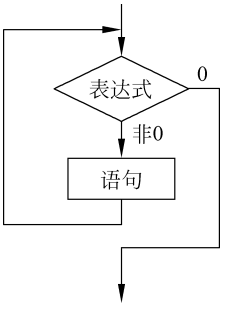

图 3-6　while 循环流程图

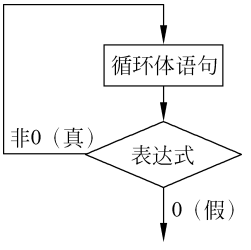

图 3-7　do…while 循环流程图

执行过程：先执行循环体，再检查条件是否成立，若成立，则执行循环体。这和 while 语句是不同的。

注意：do…while 语句的特点是，先无条件地执行循环体，再判断循环条件是否成立。

知识点 5　for 循环控制语句

for 循环是一种计数型循环语句，语句更加灵活，不仅可以用于循环次数已经确定的情况，还可以用于循环次数不确定而只给出循环结束条件的情况。它完全可以代替 while 语句，其一般形式如下：

```
for(表达式 1;表达式 2;表达式 3)
    {
```

```
        循环体;
    }
```

这 3 个表达式的主要作用说明如下。

- 表达式 1：为零个、一个或多个变量设置初始值，此语句只执行一次。
- 表达式 2：是循环条件表达式，用来判定是否继续循环。在每次执行循环体前先执行此表达式，若条件成立，则执行循环体。
- 表达式 3：迭代表达式，通常是递增或递减循环变量表达式，或者是用逗号分隔的表达式列表。

执行过程：

(1) 计算表达式 1 的值。

(2) 判断表达式 2 的值，若其值为真(非 0)，则执行 for 语句的循环体中的语句，然后执行下面的(3)；若其值为假(0)，则结束循环，转到(5)。

(3) 求解表达式 3 的值。

(4) 返回(2)继续执行。

(5) 循环结束，执行程序的下一条语句，即 for 循环语句体后面的语句。

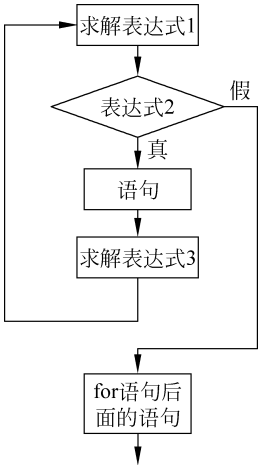

图 3-8　for 循环语句的执行流程

for 循环语句的执行流程如图 3-8 所示。

注意：

(1) for 循环中的“表达式 1(循环变量赋初值)”“表达式 2(循环条件)”和“表达式 3(循环变量增量)”都是选择项，可以省略，但“；”不能省略。

(2) 将表达式 1 放在 for 循环语句之前，例如：

```
i = 1;
for(;i <= 100;i++) sum = sum + i;
```

(3) 省略了“表达式 2(循环条件)”，则不进行其他处理便会成为死循环。

(4) 表达式 3 可以放在循环体中，例如：

```
for(i = 1;i <= 100;)
{
    sum = sum + i;
    i++;
}
```

for 语句最简单、最容易理解的形式如下：

```
for(循环变量赋初值;循环条件;循环变量增值)
{
   循环体
}
```

其中，循环变量赋初值是一个赋值语句；循环条件是一个关系表达式，决定什么时候退出循环；循环变量增值定义循环控制变量每循环一次按什么方式变化。

例如，求 1～100 的奇数和。

```
for(i = 1;i < = 100;i = i + 2)
  sum = sum + i;
```

它相当于语句：

```
    i = 1;
while(i < = 100)
{
    sum = sum + i;
    i = i + 2;
}
```

知识点 6　foreach 循环语句

foreach 是在 C＃中新引入的循环，除使用 for、while 和 do…while 语句外，还可使用 foreach 循环语句来依次遍历数组或集合中的每一个元素。

所谓遍历，就是从头到尾走一趟。遍历数组，即从头到尾逐个读出数组的每个元素。其语法格式如下：

```
foreach(<类型名称> <元素变量> in <数组或集合对象>)
{
   循环体
}
```

说明：

（1）“类型名称”是循环变量的类型，必须与数组或集合的类型一致。例如，如果要遍历一个字符串数组中的每一项，那么此处变量的类型应该是 string 类型。

（2）“元素变量”是一个循环变量，在循环中，该变量依次获取数组或集合中各元素的值。

（3）在 foreach 循环体的语句序列中，数组或集合的元素是只读的，其值不能改变。如果需要迭代数组或集合中的各元素并改变其值，应该使用 for 循环。


观看视频

知识点 7　循环的嵌套

一个循环体内又包含另一个完整的循环结构，称为循环的嵌套。内嵌的循环中还可以嵌套循环，这就是多重循环。while 循环、do…while 循环和 for 循环可以互相嵌套。

知识点 8　循环的中断

为了对循环语句进行精确控制以实现一些特殊的应用，C＃中提供了 break 和 continue。break 语句用于立即终止循环，继续执行循环后面的第一条语句。continue 语句用于终止当前的循环，继续执行下一次循环。下面举例说明 break 和 continue 的用法。该程序用于输入若干非负整数，输出奇数的平均数，当输入－1 时，程序结束，代码如下：

```
static void Main(string[] args)
    {
```

```
        int count = 0;
        float sum = 0;
        int number;
        while(true)
            {
            Console.Write("请输入整数:");
            number = Convert.ToInt32(Console.ReadLine());
            if (number == -1)
                break;
            else if (number % 2 == 0)
                continue;
            count++;
            sum += number;
           }
        Console.WriteLine("一共输入了{0}个奇数,平均值为{1}", count, sum / count);
    }
```

运行结果如图 3-9 所示。

图 3-9　运行结果

知识点 9　异常处理


观看视频

程序出错并不总是编程人员的原因,有时程序会因为用户的行为或运行环境变化而发生错误。因此,在程序中应该预测可能出现的错误,并进行相应的处理。这些在程序执行期间出现的错误或问题就称为异常。而异常处理是对程序中出现的问题的一种响应机制。C#语言的异常处理功能通过使用 try、catch、finally 语句来定义代码块,执行某些操作,以处理可能出现的异常情况。

try 语句提供一种机制,用于捕捉在块执行期间发生的各种异常。此外,try 语句可以指定一个代码块,并保证当控制离开 try 语句时总是先执行该代码块。

C#的异常处理语句 try…catch…finally 的基本语法格式如下:

```
try
{
<可能出现异常的代码块>
}
catch(ExceptionName e)
{
```

```
<捕获异常并进行处理>
}
finally
{
<负责最终的操作>
}
```

执行过程：

(1) 如果 try 下面的语句块中的每一条语句均正常执行，则跳转到(3)；如果 try 下面的语句块中的某条语句出现异常，则跳转到(2)。

(2) 执行与发生此错误匹配的 catch 语句块中的语句(catch 块可以指定要捕捉的异常类型，这个异常类型必须是 Exception 类型，或者必须是此类型的派生类型。catch 语句可以有多个，多个 catch 块的计算顺序是从顶部到底部，但是对于引发的每个异常，都只执行第一个引发的异常类型最匹配的 catch 块。如果没有任何 catch 块相匹配，则执行没有异常类型的 catch 块)，然后跳转到(3)。

(3) 执行 finally 语句块。只要存在 finally 块，它都将在执行完 try 和 catch 之后执行。finally 块始终会执行，与是否发生异常或者是否找到异常类型没有关系。因此，可以使用它来释放资源(如文件流、数据库连接等)。但是如果不需要，可以省略。

【例 3-1】 实现除法运算。

```
static void Main(string[] args)
      {
          int a, b, c;
          a = 20;
          b = 0;
          c = a / b;
          Console.WriteLine("{0}除以{1}0,得:{2}" ,a,b, c);
      }
```

以上代码没有语法错误，按 Ctrl＋F5 组合键开始执行。由于 0 作为除数没有意义，而代码中又没有处理异常，故运行后出现如图 3-10 所示的未处理异常的运行结果及如图 3-11 所示的 calculator 已停止工作窗口。在该窗口中单击“关闭程序”按钮即可退出程序运行。

图 3-10 被 0 除异常

如果在 calculator 已停止工作窗口中单击“调试程序”选项，则打开如图 3-12 所示的“选择实时调试程序”对话框，单击“确定”按钮，将出现如图 3-13 所示的“未经处理的异常”对话框。

注意：在这里可以看到异常的类型为 DivideByZeroException。

图 3-11　停止工作对话框

图 3-12　“选择实时调试程序”窗口

图 3-13　未经处理的异常

【例 3-2】　改进【例 3-1】,在代码中加入异常处理代码块 try…catch。

```
static void Main(string[ ] args)
    {
```

```
            int a, b, c;
            a = 20;
            b = 0;
            try
            {
                c = a / b;
                Console.WriteLine("{0}除以{1}0,得:{2}", a, b, c);
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine(" 除数为零!");
            }
        }
```

执行结果如图 3-14 所示。

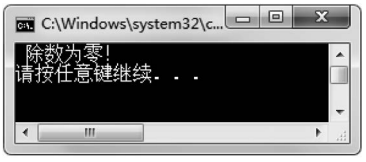

图 3-14　异常处理结果

【例 3-3】 建立控制台应用程序项目，编写整数除法运算代码，使之形成 try…catch…finally 结构，无论是否存在异常都能正确地执行其中的语句。

```
static void Main(string[] args)
        {
           try
           {
               int x, y, z;
               Console.Write("请输入整数的被除数:");
               x = int.Parse(Console.ReadLine());
               Console.Write("请输入整数的除数:");
               y = int.Parse(Console.ReadLine());
               z = x / y;
               Console.WriteLine("整除结果:" + z);
           }
           catch (FormatException e)
           {
               Console.WriteLine("格式异常:" + e.Message);
           }
           catch (DivideByZeroException e)
           {
               Console.WriteLine("除数为零异常:" + e.Message);
           }
           finally
           {
               Console.WriteLine(" ----- 程序结束");
           }
      }
```

执行程序，输入正确的除数和被除数，运行结果如图 3-15 所示。

执行程序，除数输入 0，运行结果如图 3-16 所示。

图 3-15　输入正确的除数和被除数的运行结果

图 3-16　除数为 0 的运行结果

执行程序，输入非数字时的运行结果如图 3-17 所示。

图 3-17　输入非数字的运行结果

任务 1　输入两个数 a 和 b，编写程序使 a 的值大于 b 的值

■ 任务分析

这个问题的算法很简单，只要做一次比较，然后进行一次交换即可。程序流程如图 3-18 所示。

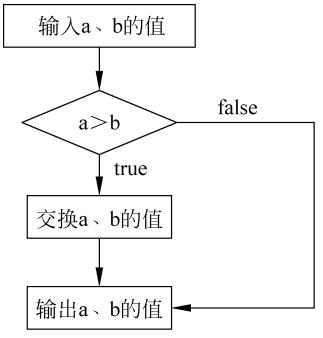

图 3-18　程序流程图

根据流程图，只要使用 if 语句进行条件判断即可。本程序的难点在于如何实现 a 和 b 两个变量值的交换。要实现 a 和 b 两个变量值的交换必须借助第 3 个变量，而不能直接使用下面的语句：

```
a = b;        //把变量 b 的值赋给变量 a,a 中原来的值就会被替换成 b 的值
b = a;        //再把变量 a 的值赋给变量 b,变量 b 的值没有改变
```

就像我们要把 a 和 b 两个水杯中的水互换，不能直接将 b 水杯中的水倒入 a 水杯中，要拿来第三个容器 c，首先将 a 中的水倒入 c 中，这样 a 就空出来了，再将 b 中的水倒入 a 中，

b就空出来了，再将c中的水倒入b中，完成a、b两个杯子中的水的互换。因此，本题可以使用下面的语句来实现交换：

```
c = a;        //把变量a的值保存在变量c中
a = b;        //把变量b的值赋给变量a，a中原来的值就会被替换成b的值
b = c;        //再把变量c的值赋给变量b，c的值等于原来a的值
```

◆ 任务实施

【步骤1】启动Visual Studio 2017，单击“文件”菜单，选择“新建”→“项目”选项，打开“新建项目”对话框，选择“控制台应用(.NET Framework)”选项，输入名称swap，选择位置为F:\C#工作目录\，如图3-19所示。

图3-19 “新建项目”对话框

【步骤2】在“新建项目”对话框中，单击“确定”按钮，打开“代码编辑器”窗口，编写代码，如图3-20所示。

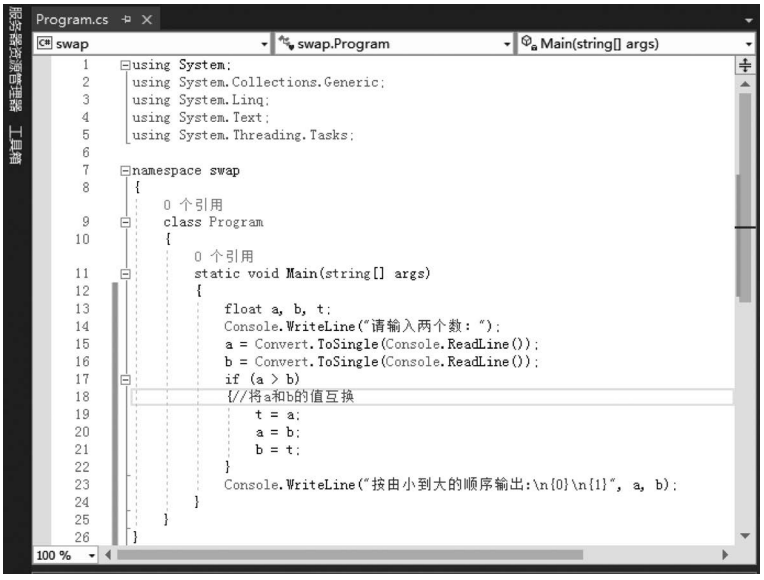

图3-20 “代码编辑器”窗口

【步骤 3】按 Ctrl＋F5 组合键或单击工具栏上的“启动”按钮，程序开始运行，依次输入两个数，运行结果如图 3-21 所示。

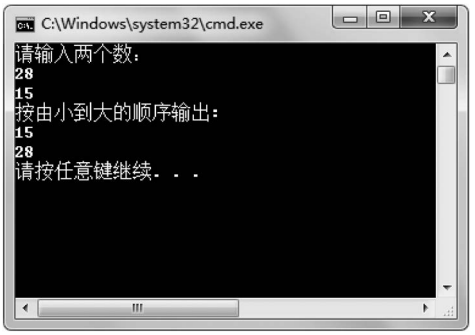

图 3-21　运行结果

注意：程序代码中的\n 为换行符。

任务 2　判断一个数是不是 3 的倍数

■ 任务分析

编写控制台应用程序，任意输入一个整数 n，判断其是不是 3 的倍数，如果是，则输出“n 是 3 的倍数”，否则输出“n 不是 3 的倍数”。该程序根据条件是否成立需要两个分支，所以使用 if 的双分支结构可以实现。程序流程图如图 3-22 所示。

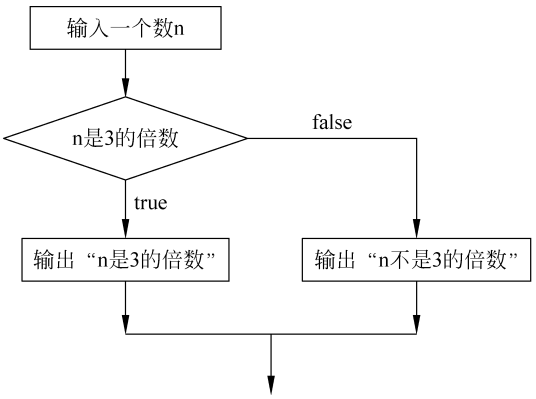

图 3-22　程序流程图

◆ 任务实施

【步骤 1】启动 Visual Studio 2017，单击“文件”菜单，选择“新建”→“项目”选项，打开“新建项目”对话框，选择“控制台应用(.NET Framework)”选项，输入名称 ThreeMultiples，选择位置为 F:\C＃工作目录\，单击“确定”按钮，打开“代码编辑器”窗口。

【步骤 2】编写代码，如图 3-23 所示。

【步骤 3】按 Ctrl＋F5 组合键或单击工具栏上的“启动”按钮，程序开始运行，输入一个整数，运行结果如图 3-24 所示。

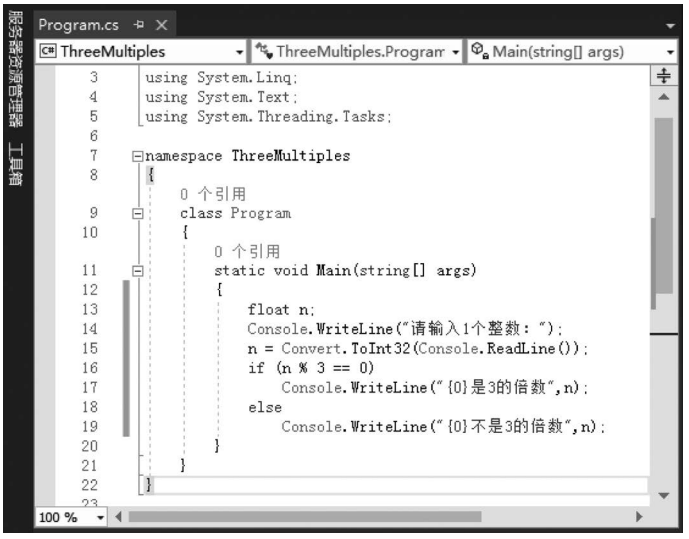

图 3-23 “代码编辑器”窗口

图 3-24 运行结果

任务 3 成绩转换

■ 任务分析

创建一个控制台应用程序，实现成绩由百分制到五级制的转换。评定标准如下：成绩大于或等于 90 分为优，成绩在 80～89 分为良，成绩在 70～79 分为中，成绩在 60～69 分为及格，成绩在 60 分以下为不及格。由于条件超过两个，因此需要使用 if 多分支结构来实现。

程序流程图如图 3-25 所示。

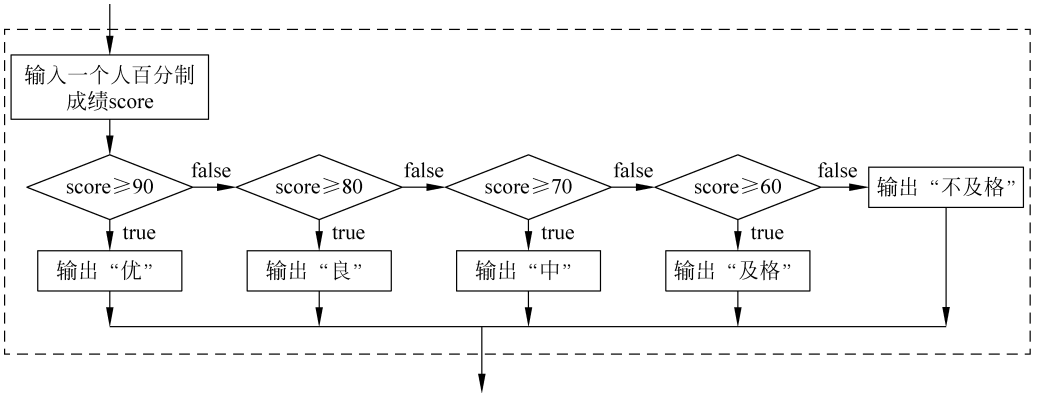

图 3-25 程序流程图

◆ **任务实施**

【步骤 1】启动 Visual Studio 2017，单击“文件”菜单，选择“新建”→“项目”选项，打开“新建项目”对话框，选择“控制台应用(.NET Framework)”选项，输入名称 ScoreConvert，选择位置为 F:\C# 工作目录\，单击“确定”按钮，打开“代码编辑器”窗口。

【步骤 2】编写代码，如图 3-26 所示。

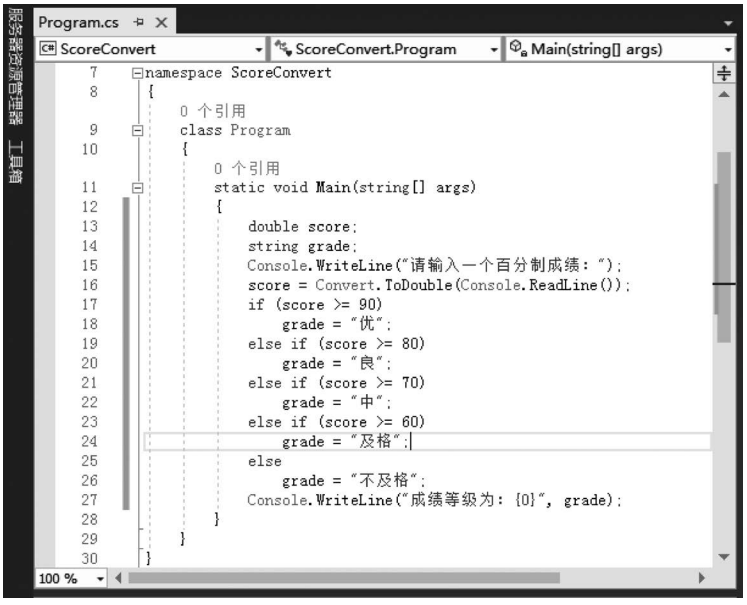

图 3-26　程序代码

【步骤 3】执行程序。

【步骤 4】按 Ctrl＋F5 组合键或单击工具栏上的“启动”按钮，程序开始运行，输入百分制成绩，运行结果如图 3-27 所示。

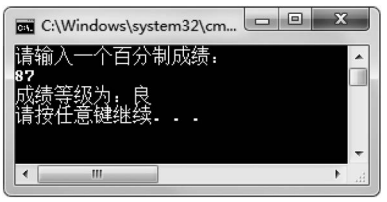

图 3-27　运行结果

说明：

(1) 本例采用 if…else if 结构实现多分支选择结构，也可以采用多个单分支的 if 语句实现。

(2) 多分支选择结构的另一种控制方法是通过 switch 语句实现。

任务 4　采用 switch 语句实现任务 3

■ **任务分析**

本例需要根据输入数据的不同范围执行不同的操作，由于分数范围超过两个，因此需要使用多分支选择结构来实现。switch 语句是将表达式的值与 case 后的常量进行比较，因此直接取输入的分数需要设计很多个 case 分支，程序较为烦琐。仔细观察，发现同一分数段的分数是有共同点的，即每个分数等级的十位上的数值是相同的。因此，控制表达式可以采用“分数整除 10”。

◆ **任务实施**

【步骤 1】启动 Visual Studio 2017，单击“文件”菜单，选择“新建”→“项目”选项，打开“新建项目”对话框，选择“控制台应用(.NET Framework)”选项，输入名称 ScoreConvertNew，选择位置为 F:\C#工作目录\，单击“确定”按钮，打开“代码编辑器”窗口。

【步骤 2】编写代码，如图 3-28 所示。

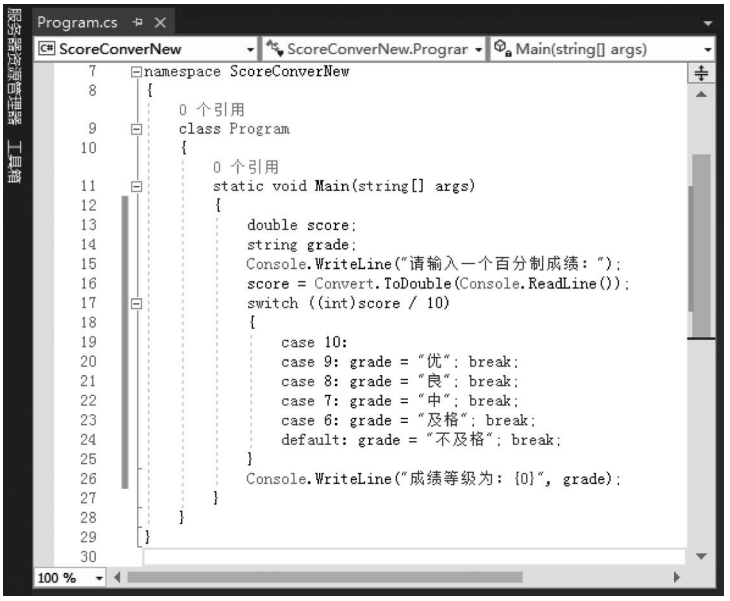

图 3-28 “代码编辑器”窗口

【步骤 3】按 Ctrl+F5 组合键或单击工具栏上的“启动”按钮，程序开始运行，输入百分制成绩，运行结果如图 3-27 所示。

说明：

(1) 本例从逻辑结构上较任务 3 更加清晰明了，建议使用多分支选择结构时优先选用 switch 语句。

(2) switch 后面的表达式不能是小数，所以需要使用(int)score/10，即将 score 强制转换为整型，注意 score 为 double 类型。

(3) case 10 和 case 9 共用同一条执行语句。

任务 5　计算景点门票优惠率

某旅游景点规定：根据月份和订门票的张数来决定门票的优惠率：在旅游旺季(7～10 月)，如果订票数超过 10 张，则票价优惠 10%，10 张以下优惠 5%；在旅游淡季(1～5 月、11 月)，如果订票数超过 10 张，则票价优惠 15%，10 张以下优惠 10%；其他情况一律优惠 5%。设计一个控制台应用程序，依据上述规则，输入订门票的月份、订门票的张数以及每张门票的原价，输出需要支付的费用。

■ **任务分析**

由于一年中一共有12个月，因此可以考虑采用12个case分支的switch语句作为主流程控制结构。对于同一个月份而言，不同的订票数量对应的优惠率是不同的，因此需要采用双分支的if…else结构进行处理。综合分析，程序可采用switch语句中嵌套if…else语句来实现。

◆ **任务实施**

【步骤1】启动Visual Studio 2017，单击“文件”菜单，选择“新建”→“项目”选项，打开“新建项目”对话框，选择“控制台应用(.NET Framework)”选项，输入名称TickePreference，选择位置为F:\C#工作目录\，单击“确定”按钮，打开“代码编辑器”窗口。

【步骤2】编写如下代码：

```
static void Main(string[] args)
{
            int month;
            int ticketcount;
            double Preference;
            double price;
            double pricesum;
            Console.WriteLine("请输入月份:");
            month = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("请输入订票数:");
            ticketcount = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("请输入每张门票的原价:");
            price = Convert.ToDouble(Console.ReadLine());
            switch (month)
            {
                case 1:
                case 2:
                case 3:
                case 4:
                case 5:
                case 11:
                    if (ticketcount >= 10)
                        Preference = 0.1;
                    else
                        Preference = 0.05;
                    break;
                case 7:
                case 8:
                case 9:
                case 10:
                    if (ticketcount >= 10)
                        Preference = 0.15;
                    else
                        Preference = 0.1;
                    break;
                default:
                    Preference = 0.05; break;
            }
```

```
        pricesum = price * (1 - Preference) * ticketcount;
        Console.WriteLine("{0}月份订购{1}张门票,总价为{2}元.",month,ticketcount,
    pricesum);
    }
```

【步骤 3】执行程序，输入月份、订票数及票价，输出结果如图 3-29 所示。

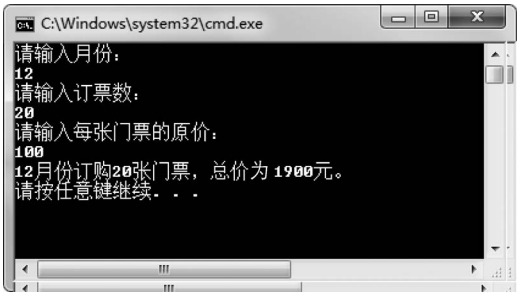

图 3-29　运行结果

任务 6　简单计算器

■ 任务分析

编写控制台应用程序，实现简单的计算器功能，输入要运算的数据以及运算符，进行相应的计算，程序运行结果如图 3-30 所示。

◆ 任务实施

【步骤 1】启动 Visual Studio 2017，单击“文件”菜单，选择“新建”→“项目”选项，打开“新建项目”对话框，选择“控制台应用(.NET Framework)”选项，输入名称 calculator，选择位置为 F:\C#工作目录\，单击“确定”按钮，打开“代码编辑器”窗口。

【步骤 2】编写如下代码：

```
static void Main(string[] args)
{
    float a, b;
    string c;
    Console.WriteLine("请输入要运算的数据:");
    a = Convert.ToSingle(Console.ReadLine());
    b = Convert.ToSingle(Console.ReadLine());
    Console.WriteLine("请输入运算符:");
    c = Convert.ToString(Console.ReadLine());
    Console.Write("运算结果:");
    switch (c)
    {
        case "+":
            Console.WriteLine("{0}+{1}={2}", a, b, a + b); break;
        case "-":
            Console.WriteLine("{0}-{1}={2}", a, b, a - b); break;
        case "*":
            Console.WriteLine("{0}*{1}={2}", a, b, a * b); break;
        case "/":
```

```
            if (b == 0)
                Console.WriteLine("错误!被除数不能为零.\n");
            else
                Console.WriteLine("{0}/{1} = {2}", a, b, a / b); break;
        default:
            Console.WriteLine("不合法的运算符!\n"); break;
    }
}
```

【步骤 3】执行程序，输入要运算的数据及运算符，运行结果如图 3-30 所示。

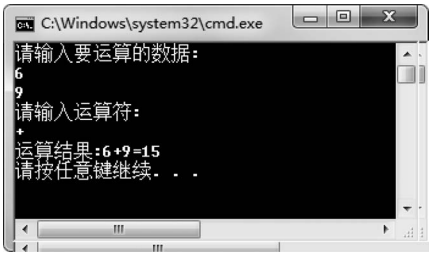

图 3-30　运行结果

任务 7　输出 100 以内的所有奇数和、偶数和

■ 任务分析

要求 100 以内的奇数和，即求 1＋3＋5＋…＋99 的值，这是一个典型的累加操作，需要使用循环语句来完成。相邻的两个操作数相差 2，所以循环体内改变循环变量取值的语句为 i＝i＋2；要求 100 以内偶数的和，即求 2＋4＋6＋…＋100，方法同求奇数的和。奇数和与偶数和的累加可以放在一个循环中实现，即在循环体中通过 if…else 语句分别进行处理。

◆ 任务实施

【步骤 1】启动 Visual Studio 2017，单击“文件”菜单，选择“新建”→“项目”选项，打开“新建项目”对话框，选择“控制台应用(.NET Framework)”选项，输入名称 aggregate，选择位置为 F:\C#工作目录\，单击“确定”按钮，打开“代码编辑器”窗口。

【步骤 2】编写如下代码：

```
static void Main(string[] args)
    {
        int oddsum = 0, evensum = 0, i = 1;
        while (i <= 100)
        {
            if (i % 2 != 0)
                oddsum = oddsum + i;
            else
                evensum = evensum + i;
            i = i + 1;
        }
        Console.WriteLine("100 以内奇数的和为:{0}", oddsum);
        Console.WriteLine("100 以内偶数的和为:{0}", evensum);
    }
```

【步骤 3】执行程序，运行结果如图 3-31 所示。

图 3-31　运行结果

任务 8　用 do…while 语句改写任务 7

◆ 任务实施

【步骤 1】创建控制台应用程序。

【步骤 2】输入如下代码：

```
static void Main(string[] args)
    {

        int oddsum = 0, evensum = 0, i = 1;
        do
        {
            if (i % 2 != 0)
                oddsum = oddsum + i;
            else
                evensum = evensum + i;
            i = i + 1;
        } while (i <= 100);
        Console.WriteLine("100 以内奇数的和为:{0}", oddsum);
        Console.WriteLine("100 以内偶数的和为:{0}", evensum);

    }
```

【步骤 3】执行程序，结果如图 3-31 所示。

说明：从任务 7 和任务 8 可以看出，do…while 语句结构和 while 语句结构可以相互转换。在一般情况下，用 while 语句和用 do…while 语句处理同一问题时，若二者的循环体部分是一样的，那么结果也一样。例如任务 7 和任务 8 中的循环体是相同的，得到的结果也相同。但是如果 while 后面的表达式一开始就为假(0 值)，则两种循环的结果是不同的。例如下面两段程序。

1. while 循环

```
static void Main(string[] args)
    {
        int i, sum = 0;
        Console.WriteLine("请输入 i 的值");
        i = int.Parse(Console.ReadLine());
        while (i <= 10)
```

```
        {
            sum = sum + i;
            i++;
        }
        Console.WriteLine("sum = {0}", sum);
    }
```

2. do…while 循环

```
static void Main(string[] args)
    {
        int i, sum = 0;
        Console.WriteLine("请输入 i 的值");
        i = int.Parse(Console.ReadLine());
        do
        {
            sum = sum + i;
            i++;
        } while (i <= 10);

        Console.WriteLine("sum = {0}\n", sum);
    }
```

分别运行两段代码，当输入 i 的值为 1 时，两段程序的运行结果分别如图 3-32 和图 3-33 所示。

图 3-32　while 循环运行结果

图 3-33　do…while 循环运行结果

当输入 i 的值为 11 时，两段程序的运行结果分别如图 3-34 和图 3-35 所示。

图 3-34　while 循环运行结果

图 3-35　do…while 循环运行结果

可以看到，当输入 i 的值小于或等于 10 时，二者得到的结果相同。而当 i＞10 时，二者得到的结果就不同了。这是因为此时对 while 循环来说，一次也不执行循环体（表达式 i＜＝10 的值为假），而对 do…while 循环来说，至少要执行一次循环体。因此，可以得出结论：如果两者的循环体相同，当初始条件为真时，两种循环语句得到的结果是相同的；否则，二者得出的结果是不同的。

任务 9　用 for 循环改写任务 7

■ 任务分析

任务 7 中循环执行的起始值、终止值以及循环变量每次变化的量都是已知的，所以可以使用 for 循环来完成。

◆ 任务实施

【步骤 1】创建控制台应用程序。

【步骤 2】输入如下代码：

```
static void Main(string[] args)
    {
        int oddsum = 0, evensum = 0;
        int i;
        for (i = 1; i <= 100; i++)
        {
            if (i % 2 != 0)
                oddsum = oddsum + i;
            else
                evensum = evensum + i;
        }
        Console.WriteLine("100 以内奇数的和为:{0}", oddsum);
        Console.WriteLine("100 以内偶数的和为:{0}", evensum);
    }
```

【步骤 3】执行程序，运行结果如图 3-31 所示。

任务 10　利用 foreach 统计字符串中各种字符的个数

■ 任务分析

本任务要统计用户输入的字符串中大写字母的个数、小写字母的个数、数字的个数以及其他字符的个数。任务实现过程：本任务需要 4 个计数变量，分别表示大写字母个数、小写字母个数、数字的个数和其他字符的个数，4 个变量的初始值都为 0。输入字符串 s，对 s 中的每一个字符进行遍历，并判断字符属于大写字母、小写字母、数字还是其他字符，并将所属类的计数变量的值增加 1。程序流程图如图 3-36 所示。

◆ 任务实施

【步骤 1】创建控制台应用程序。

【步骤 2】编写如下代码：

```
static void Main(string[] args)
    {
        int upsum = 0, lowsum = 0,numsum = 0,othersum = 0;
```

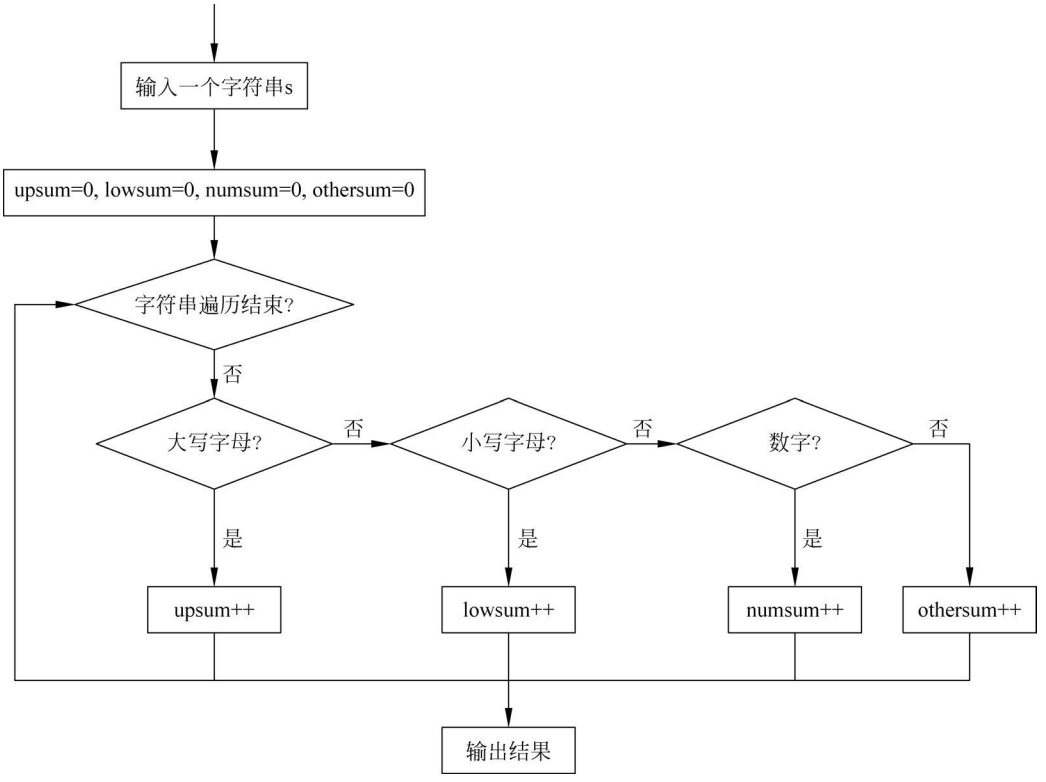

图 3-36　字符统计流程图

```
        string s;
        Console.WriteLine("请输入字符串:");
        s = Console.ReadLine();
        foreach(char c in s)
        {
            if (c >= 'A' && c <= 'Z')
                upsum++;
            else if (c >= 'a' && c <= 'z')
                lowsum++;
            else if (c >= '0' && c <= '9')
                numsum++;
            else
                othersum++;
        }
        Console.WriteLine("大写字母的个数为:{1}", upsum);
        Console.WriteLine("小写字母的个数为:{1}", lowsum);
        Console.WriteLine("数字的个数为:{1}", numsum);
        Console.WriteLine("其他字符的个数为:{1}", othersum);
    }
```

【步骤 3】执行程序,输入字符串,程序运行结果如图 3-37 所示。

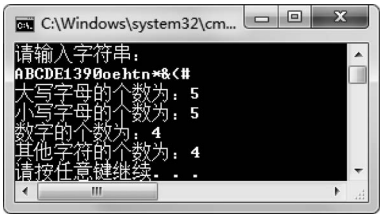

图 3-37　运行结果

任务 11　石头、剪刀、布猜拳游戏

模拟“石头、剪刀、布”五局三胜猜拳游戏：选手和计算机轮流猜拳 5 次，3 次胜利者赢。

■ 任务分析

选手输入选项(“剪刀”“石头”“布”)，计算机随机给出选项，按照游戏规则：“布”>“石头”、“石头”>“剪刀”、“剪刀”>“布”进行评判和计数，一旦一方满足五局三胜，则游戏结束。流程图如图 3-38 所示。

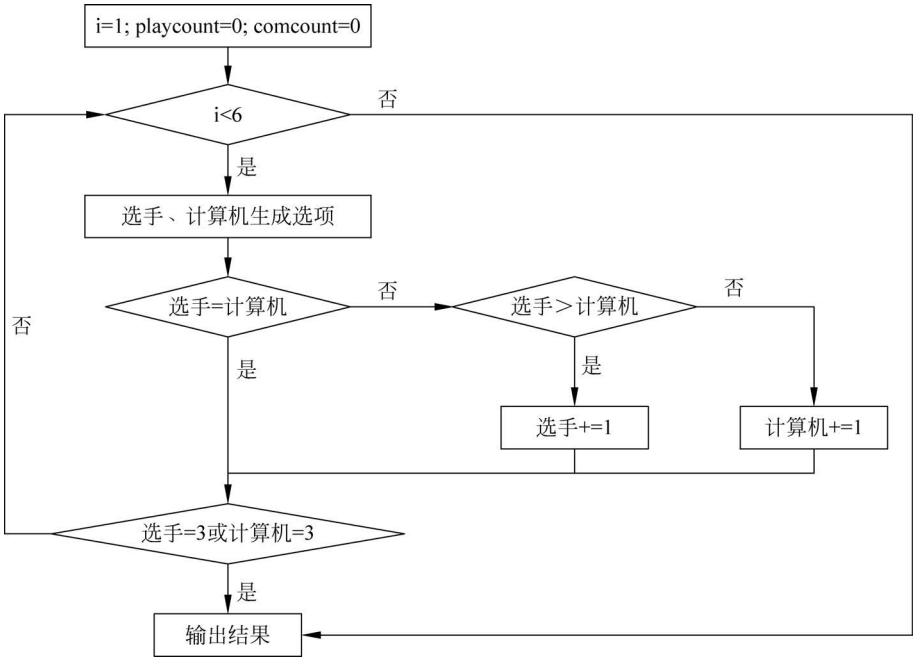

图 3-38　流程图

◆ 任务实施

【步骤 1】创建控制台应用程序。

【步骤 2】输入如下代码：

```
static void Main(string[] args)
    {
        int playcount = 0, computercount = 0;
```

```
        int i,computer,play;
        Random rd = new Random();
        string s;
        for(i = 1;i < 6;i++)
        {
            Console.WriteLine("第{0}局", i);
            Console.WriteLine("选手(石头、剪刀、布):");
            s = Convert.ToString(Console.ReadLine());
            computer = rd.Next(1, 3);
            if (computer == 1)
                Console.WriteLine("计算机:剪刀");
            else if (computer == 2)
                Console.WriteLine("计算机:石头");
            else
                Console.WriteLine("计算机:布");
            if (s == "剪刀")
                play = 1;
            else if (s == "石头")
                play = 2;
            else
                play = 3;

            if (play == computer)
                Console.WriteLine("选项一样!");
            else if ((play == 1 && computer == 2) || (play == 2 && computer == 3) || (play == 3
&& computer == 1))
            {
                Console.WriteLine("计算机赢!");
                computercount++;
            }
            else
            {
                Console.WriteLine("选手赢!");
                playcount++;
            }
            if (computercount == 3 || playcount == 3)
                break;
            Console.WriteLine("_____________________________");
        }
        if (computercount > playcount)
            Console.WriteLine("最终计算机胜出!");
        else if (computercount < playcount)
            Console.WriteLine("最终选手胜出!");
        else
            Console.WriteLine("平局!");
    }
```

【步骤 3】执行程序，输入“石头”或“剪刀”或“布”，程序运行结果如图 3-39 所示。

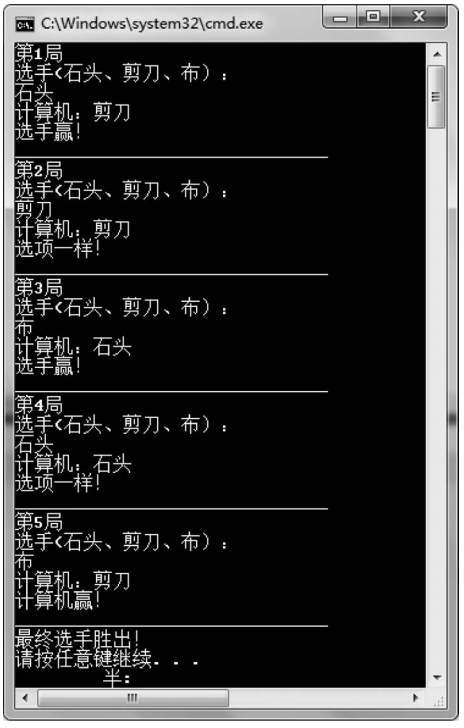

图 3-39　运行结果

任务 12　输出图形

设计一个 Windows 窗体应用程序，根据用户的要求，输出三角形和平行四边形图案。

■ 任务分析

输出的图形由行和列构成，所以用双重循环来实现。

◆ 任务实施

【步骤 1】程序设计界面。

（1）创建一个 Windows 应用（.NET Framework），添加一个标签控件（label1）和两个命令按钮控件（button1 和 button2），适当调整控件的大小和布局。

（2）设置窗体及控件属性。

设置窗体与控件的 Text 属性。设置 label1 的 AutoSize 属性为 False，BorderStyle 属性为 Fixed3D，Text 属性为空。

【步骤 2】输入如下代码：

```
private void button1_Click(object sender, EventArgs e)
    {
        int i, j, k;
        label1.Text = "三角形:\n\n";
        for (i = 1; i <= 9; i++)
          {
```

```
            for (j = 1; j <= 20 - i; j++)
                label1.Text += " ";
            for (k = 1; k <= 2 * i - 1; k++)
                label1.Text += "*";
            label1.Text += "\n";
        }

    }
    private void button2_Click(object sender, EventArgs e)
    {
        int i, j, k;
        label1.Text = "平行四边形:\n\n";
        for (i = 1; i <= 9; i++)
        {
            for (j = 1; j <= 15 - i; j++)
                label1.Text += " ";
            for (k = 1; k <= 17; k++)
                label1.Text += "*";
            label1.Text += "\n";
        }
    }
```

【步骤 3】执行程序，分别单击“三角形”按钮和“平行四边形”按钮，结果分别如图 3-40 和图 3-41 所示。

图 3-40　输出三角形

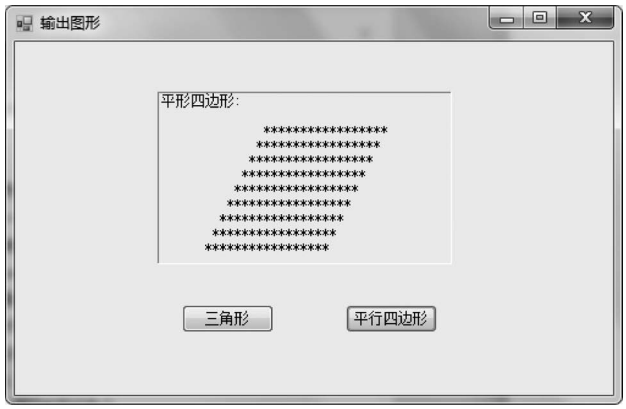

图 3-41　输出平行四边形

任务 13　输出斐波那契数列的前 20 项

意大利著名的数学家斐波那契在《计算之书》中指出了一个有趣的兔子问题：一对成年兔子每个月恰好生下一对小兔子(一雌一雄)。在年初时，只有一对小兔子。在第一个月结束时，它们成长为成年兔子，并且在第二个月结束时，这对成年兔子再生下一对小兔子。这种成长与繁殖的过程会一直持续下去，假设生下的小兔子都不会死，那么一年之后可有多少对兔子？

■ 任务分析

斐波那契数列又称黄金分割数列、兔子数列。分析题目可以发现，年初时只有一对小兔子，第二个月这一对小兔子长成中兔子(兔子总数为 1 对)；第三个月中兔子长成大兔子并生下一对小兔子(兔子总数为 2 对)；第四个月小兔子长成中兔子，大兔子又生下一对小兔子(兔子总数为 1＋1＋1＝3 对)。因此，得出斐波那契数列为 1,1,2,3,5,8,…，从第三个数开始，后一个数是前两个数的和。表达式如下：

$$F(n)=\begin{cases}1 & n=1 \text{ 和 } n=2\\ F(n-1)+F(n-2) & n>2\end{cases}$$

◆ 任务实施

【步骤 1】创建控制台应用程序。

【步骤 2】输入如下代码：

```
static void Main(string[] args)
  {
      int f1 = 1;
      int f2 = 1;
      int i;
      for(i = 1;i <= 20;i++)
      {
          f1 = f1 + f2;
          f2 = f2 + f1;

          Console.Write(f1.ToString() + "  " + f2.ToString() +  "  ");
      }
  }
```

【步骤 3】执行程序，运行结果如图 3-42 所示。

图 3-42　运行结果

任务 14　输出 1000 以内的完数

■ 任务分析

一个数如果恰好等于它的因子之和，则这个数称为“完数”。例如，6 的因子为 1，2，3，而 6＝1＋2＋3。

◆ 任务实施

【步骤 1】创建控制台应用程序。

【步骤 2】输入如下代码：

```
static void Main(string[] args)
    {
        for (int n = 1; n <= 1000; n++)
        {
            int sum = 0;
            for (int i = 1; i < n; i++)
            {
                if (n % i == 0)
                    sum += i;
            }
            if (n == sum)
                Console.WriteLine(n.ToString());
         }
    }
```

【步骤 3】执行程序，输出 1000 以内的完数为：6，28，496。

任务 15　百钱买百鸡问题的求解

我国古代数学家张丘建在《算经》一书中提出了一个数学问题：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、鸡雏各几何？

■ 任务分析

假设有 x 只鸡翁（公鸡），y 只鸡母（母鸡），则鸡雏（小鸡）有（100－x－y）只。而每只公鸡值 5 元，每只母鸡值 3 元，3 只小鸡值 1 元，所以 5x＋3y＋1/3（100－x－y）＝100，这个公式未知数有两个，所以无法用普通的方法求解，可以使用穷举法。

穷举法是计算机求解复杂问题时常用的算法，用来解决那些通过公式推导、规则演绎等方法不能解决的问题。穷举法就是将可能的情况一一列举出来。本题中 100 元全部买公鸡可以买 20 只，全部买母鸡可以买 33 只，所以公鸡数量的取值范围为 0～20，母鸡数量的取值范围为 0～33，使用双重循环来完成。

◆ 任务实施

【步骤 1】创建控制台应用程序。

【步骤 2】输入如下代码：

```
static void Main(string[] args)
    {
        {
            int i, j;
            for (i = 0; i <= 33; i++)        //外循环控制输出的行数
              {
                for (j = 0; j <= 20; j++)   //内循环 1,用于输出空格
                    if (i * 3 + j * 5 + (100 - i - j) / 3 == 100 && (100 - i - j) % 3 == 0)
                        Console.WriteLine("母鸡为:{0}只,公鸡为:{1}只,小鸡为:{2}只", i,
                        j, 100 - i - j);
              }
        }

    }
```

【步骤 3】执行程序，运行结果如图 3-43 所示。

图 3-43　运行结果

项 目 小 结

结构化程序设计有 3 种基本结构：顺序结构、选择结构和循环结构。顺序结构最简单，就是按照语句出现的先后次序执行。选择结构也叫分支结构，分为单分支选择结构、双分支选择结构和多分支选择结构。C＃中实现分支结构的语句有两个：if 语句和 switch 语句。循环结构是只要满足循环执行的条件就重复地执行循环体内的语句。C＃提供了 4 种不同的循环机制：for、while、do…while 和 foreach。

跳转语句可实现程序的无条件转移。C＃的跳转语句包括 break、continue、goto 和 return 语句。

C＃通过使用 try…catch…finally 语句来处理系统级和应用程序级的错误。这些语句允许开发者尝试执行可能不会成功的操作，处理失败情况，并在操作完成后进行资源清理等。

拓 展 实 训

一、实训的目的和要求

1. 熟练掌握分支结构程序设计的基本方法。
2. 掌握 if…else 语句和 switch 语句的语法格式和使用技巧。
3. 熟练掌握循环结构程序设计的方法。
4. 掌握 while 语句、do…while 语句和 for 语句的语法格式和使用技巧。

5. 综合运用选择结构和循环结构解决问题。

二、实训内容

1. 创建 Windows 窗体应用程序，程序运行时，在“年份”文本框中输入一个正整数，单击“判断”按钮，就能在标签控件中显示判断结果，运行结果如图 3-44 所示。

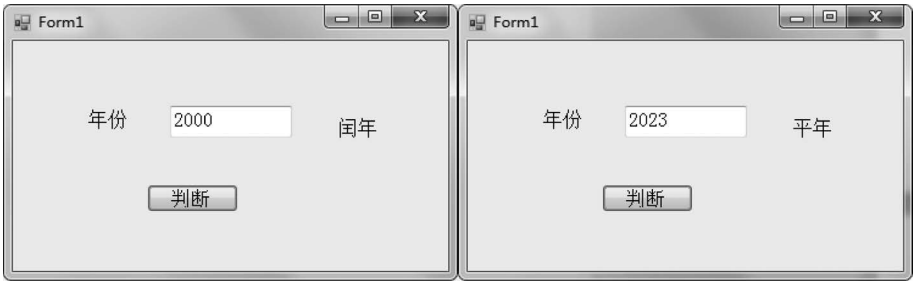

图 3-44　运行结果

2. 创建控制台应用程序，求解猴子吃桃问题：猴子第一天摘下若干桃子，当即吃了一半后又多吃了一个；第二天将剩下的桃子吃掉一半后再多吃一个；以后每天都吃掉前一天剩下的一半零一个。到第 10 天想再吃时，只剩下一个桃子。求猴子第一天共摘下多少个桃子？

3. 所谓“水仙花数”，指的就是这样一个三位数：其各位数的立方和等于该数。例如 $153=1^3+5^3+3^3$。编写控制台应用程序，计算输出所有的水仙花数。

4. 创建一个控制台应用程序，输入月份，输出对应的季节。

5. 编写程序，求解以下问题：

（1）创建 Windows 窗体应用程序，在窗体上输出 100～500 的所有奇数，并计算它们的和。

（2）创建 Windows 窗体应用程序，统计 1～1000 中既能被 5 整除，又能被 7 整除的数的个数，并输出在窗体上。

（3）创建控制台应用程序，从 300 开始，找出连续 100 个既能被 3 整除又能被 5 整除的数。

（4）创建控制台应用程序，计算 s=1！+2！+3！+…+n！（其中 n 是用户输入的正整数）。

（5）创建控制台应用程序，计算不大于 1000 的 10 个最大的素数。

习　　题

一、选择题

1. 先判断条件的循环语句是(　　)。

A. do…while　　B. while　　C. while…do　　D. do…loop

2. 下列语句中，控制台上输出的是(　　)。

```
if(true)
   System.Console.WriteLine("FirstMessage");
   System.Console.WriteLine("SecondMessage");
```

A. FirstMessage
SecondMessage

B. SecondMessage

C. 无输出

D. FirstMessage

3. 执行C#语句序列：int i; for(i=0;i++<4;);后，变量的值是(　　)。

A. 5　　B. 4　　C. 1　　D. 0

4. 若有如下程序，则语句a+1执行的次数是(　　)。

```
static void Main(string[] args)
    {
        int x = 1, a = 1;
        do
        {
            a = a + 1;
        }while(x!= 0)
    }
```

A. 0　　B. 1　　C. 无限次　　D. 有限次

5. 有如下程序，该程序的输出结果是(　　)。

```
static void Main(string[] args)
    {
        int n = 9;
        while (n > 6)
        {
            n--;
            Console.WriteLine(n);
        }

    }
```

A. 987　　B. 876　　C. 8765　　D. 9876

二、填空题

1. 当程序中执行到_____语句时，将结束在循环语句中循环体的一次执行。
2. 在switch语句中，每个语句标号所含关键字case后面的表达式必须是_____。
3. 在while循环语句中，一定要有修改循环条件的语句，否则可能造成_____。
4. 在C#语言中，实现循环的语句主要有while、do-while、for和_____。
5. Visual Studio提供的_____方法就是专门用于人为引发异常的。

三、写出程序的运行结果

1. 有以下程序段：

```
class Program
{
    static void Main(string[] args)
    {
        int s = 0,i;
        for(i = 1;;i++)
        {if(s > 50) break ;
```

```
            if(i % 2 == 0)s += i;
        }
        Console.WriteLine("i,s = " + i + "," + s);
    }
}
```

运行结果为：_____。

2. 有以下程序段：

```
static void Main(string[] args)
    {
        int x = 15;
        while (x > 10 && x < 50)
        {
            x++;
            if (x / 3!= 0) { x++; break; }
            else continue;
        }
        Console.WriteLine(x);
    }
```

运行结果为：_____。

3. 有以下程序段：

```
static void Main(string[] args)
    {
        int k = 4, n = 0;
        for (; n < k; )
        {
            n++;
            if (n % 3 != 0)
                continue;
            k--;
        }
        Console.WriteLine("{0},{1}", k, n);
    }
```

运行结果为：_____。

4. 有以下程序段：

```
static void Main(string[] args)
    {
        int k = 5, n = 0;
        while (k > 0)
        {
            switch (k)
            {
                case 1: n += k; break;
                case 2: break;
                case 3: n += k; break;
                default: break;
            }
```

```
            k = k - 1;
        }
    Console.WriteLine(n);
    }
```

运行结果为：_____。

5. 有以下程序段：

```
static void Main(string[] args)
    {
        int i = 0, s = 0;
        do
        {
            if (i % 2 != 0)
            {
                i++;
                continue;
            }
            i++;
            s += i;
        } while (i < 7);
        Console.Write(s);
    }
```

运行结果为：_____。

四、编程题

1. 猜数字。先由计算机“想”一个 1～100 的数请人猜，如果猜对了，则结束游戏，并在屏幕上输出猜了多少次才猜对此数，以此来反映猜数者“猜”的水平；否则计算机给出提示，告诉人们所猜的数是太大还是太小，直到猜对为止。

2. 有规律数列求和。有一个分数序列：2/1，3/2，5/3，8/5，13/8，21/13，找出数列的规律并求出前 30 项的和。

3. 输入一个数，判断它是不是素数。什么样的数是素数？如果一个数除了 1 和它本身之外不能被任何数整除，这个数就是素数。所以对于输入的这个数 n，要分别用这个数除以 2～n－1(n/2～n/n－1)，共需执行 n－2 次。

4. 某公司为了防止数据信息泄露，想对数据进行加密，数据是小于 8 位的整数。加密规则如下：将每位数字都加上 5，再用和除以 10 的余数代替该数字，最后将第一位数字和最后一位数字交换，如输入 1234，则应输出 9786，请实现加密程序。

5. 输出九九乘法表(上三角、下三角)。