第3章 流程控制与算法

实验 3.1 顺序结构程序设计

实验目的

- 了解结构化程序设计的思想。
- 掌握顺序结构程序设计的方法。
- 熟练掌握 C# 中控制台输入输出的方法。
- 掌握赋值语句的使用方法。

实验内容

【案例 3-1】 输入一个小写字母,转换为对应的大写字母输出。

1. 问题分析

小写字母 a 的 ASCII 的值为 97,大写字母 A 的 ASCII 值为 65,所以大小写字母 ASCII 值的差值为 32。输入一个小写字母,将其转换为大写字母,只需将其转换为整型后减去 32 即可。

进行顺序结构程序设计时,读者要明确以下 3 点。

首先,输入是什么?

其次,中间的数据处理过程是怎样的?

最后,输出结果是什么?

在本案例中,输入的是一个小写字母 c1;中间处理过程为 c2=c1-32(c2 为大写字母);最后输出 c2。

2. 操作步骤

(1) 新建一个控制台应用程序,并在 Main()方法中输入如下代码。

```
char c1,c2;              //c1 为小写字母,c2 为转换后的大写字母
Console.WriteLine("请输入一个小写字母: ");
```

```
c1=Convert.ToChar(Console.ReadLine());          //输入一个小写字母
c2=Convert.ToChar(Convert.ToInt32(c1)-32);      //小写字母转换为大写字母
Console.WriteLine("您输入的小写字母为：{0},其对应的大写字母为：{1}", c1, c2);
Console.ReadLine();
```

(2) 执行程序,运行结果如图 3-1 所示。

图 3-1　案例 3-1 的运行结果

说明

虽然顺序结构程序的设计比较简单,但读者设计时需要注意步骤的前后顺序不能随意打乱。在上例中,如果将语句:

```
c2=Convert.ToChar(Convert.ToInt32(c1)-32);      //小写字母转换为大写字母
```

放在

```
c1=Convert.ToChar (Console.ReadLine());         //输入一个小写字母
```

的前面,编译时将会报错,因为此时变量 c1 还未被赋值。

【案例 3-2】 输入两个整型变量 a 和 b 的值,输出 a 除以 b 的商及余数。

1. 问题分析

求两个整型变量的商和余数,分别采用运算符/和%。

2. 操作步骤

(1) 新建一个控制台应用程序,并在 Main()方法中添加如下代码。

```
int a, b, c, d;             //c 代表商,d 代表余数
Console.WriteLine("请输入两个正整数：");
a=Convert.ToInt32(Console.ReadLine());
b=Convert.ToInt32(Console.ReadLine());
c=a / b;                    //求两个整数的商
d=a % b;                    //求两个整数的余数
Console.WriteLine("{0}和{1}的商为{2},余数为{3}。", a, b, c, d);
Console.ReadLine();
```

(2) 执行程序,运行结果如图 3-2 所示。

图 3-2　案例 3-2 的运行结果

【案例 3-3】 求一元二次方程 $ax^2+bx+c=0$ 的两个根,其中 a、b、c 由键盘输入,设 $b^2-4ac\geqslant 0$。

1. 问题分析

求一元二次方程根的公式如下。

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

从键盘输入 a、b、c 的值(输入时要保证 $b^2-4ac\geqslant 0$),之后分别计算 x_1 和 x_2 的值,并输出即可。

2. 操作步骤

(1) 新建一个控制台应用程序,在 Main()方法中添加如下代码。

```
int a, b, c;
double x1, x2;
Console.WriteLine("请输入 a、b、c 的值: ");
a=Convert.ToInt32(Console.ReadLine());
b=Convert.ToInt32(Console.ReadLine());
c=Convert.ToInt32(Console.ReadLine());
x1=(-b+Math.Sqrt(b*b+4*a*c)) / (2*a);
x2=(-b-Math.Sqrt(b*b+4*a*c)) / (2*a);
Console.WriteLine("一元二次方程的两个根分别为: {0},{1}", x1, x2);
Console.ReadLine();
```

(2) 执行程序,程序的运行结果如图 3-3 所示。

图 3-3　案例 3-3 的运行结果

实践提高

(1) 设计一个程序,输入两个整数,分别输出两个数相加、相减、相乘、相除的结果,运行结果如图 3-4 所示。

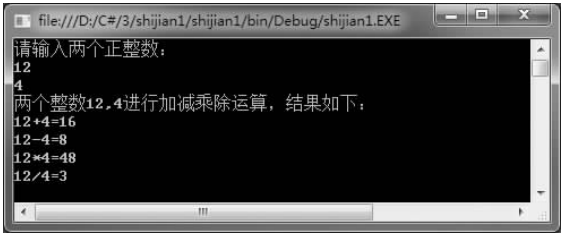

图 3-4　运行结果

(2) 输入两个整数变量,交换两个变量的值后输出。

提示　要交换两个变量 a、b 的值,需引入第 3 个变量 c 作为中转变量,交换过程分为

如下 3 步。

① a→c(将变量 a 的值赋给 c)。

② b→a(将变量 b 的值赋给 a)。

③ c→b(将变量 c 的值赋给 b)。

实验 3.2　选择结构程序设计

实验目的

- 掌握逻辑表达式的含义和作用。
- 熟练掌握各种 if 语句的使用方法。
- 熟练掌握 switch 语句的使用方法。
- 能够使用选择结构解决常见问题。

实验内容

【案例 3-4】 输入三角形的三个边长,计算三角形的面积。

1. 问题分析

首先要判断输入的三个边长是否构成三角形,如果构成三角形,再利用第 2 章实验中讲到的海伦公式计算三角形的面积。否则输出一个提示信息:不能构成三角形。算法描述如下。

① 输入三角形的三个边长。

② 判断能否构成三角形,如果能构成三角形,转换到第③步,否则转到第④步。

③ 计算三角形的面积。

④ 输出提示信息:不能构成三角形。

⑤ 程序结束。

2. 操作步骤

(1) 新建一个控制台应用程序,并在 Main()方法中输入如下代码。

```
int a, b, c;
Console.WriteLine("请输入三角形的三个边长: ");
a=Convert.ToInt32(Console.ReadLine());
b=Convert.ToInt32(Console.ReadLine());
c=Convert.ToInt32(Console.ReadLine());
//判断能否构成三角形,如能构成,计算其面积
if (a+b >c && a+c >b && b+c >a)
{
    double s, area;
    s=(a+b+c) / 2.0;
```

```
    area=Math.Sqrt(s*(s-a)*(s-b)*(s-c));
    Console.WriteLine("边长为{0},{1},{2}的三角形面积为:{3}", a, b, c, area);
}
else
{
    Console.WriteLine("边长分别为:{0},{1},{2}不能组成一个三角形!", a, b, c);
}
Console.ReadLine();
```

(2) 执行程序,按照提示输入三角形的边长,两种不同的输出结果如图 3-5 所示。

(a) 能构成三角形的输出结果　　(b) 不能构成三角形的输出结果

图 3-5　案例 3-4 的运行结果

【案例 3-5】 输入一个字符,如果该字符是小写字母,将其转换为对应的大写字母输出;如果是大写字母,将其转换为对应的小写字母输出;如果是其他字符,则原样输出。

1. 问题分析

本案例涉及 3 种情况,所以可以使用多分支 if 语句实现:第一个分支用来判断输入的字符 c1 是否是小写字母,其逻辑表达式如下。

```
c1 >='a' && c1 <='z'
```

如果不符合该条件,可进入第二个分支的判断,判断它是否是大写字母,逻辑表达式如下。

```
c1>='A' && c1<='Z'
```

如果以上两个条件都不符合,那么就是其他字符,直接执行 else 部分。

2. 操作步骤

(1) 新建一个控制台应用程序,并在 Main()方法中添加如下代码。

```
char c1, c2;
Console.WriteLine("请输入一个字符:");
c1=Convert.ToChar (Console.ReadLine());
if (c1>='a' && c1<='z')
{
    //将小写字母转换为大写字母
    c2=Convert.ToChar(Convert.ToInt32(c1)-32);
}
else if (c1>='A' && c1<='Z')
{
```

```
    //将大写字母转换为小写字母
    c2=Convert.ToChar(Convert.ToInt32(c1)+32);
}
else
{
    //其他字符不做转换
    c2=c1;
}
Console.WriteLine("输入的原始字符为{0},转换后为{1}", c1, c2);
Console.ReadLine();
```

(2) 执行程序,运行结果如图 3-6 所示。

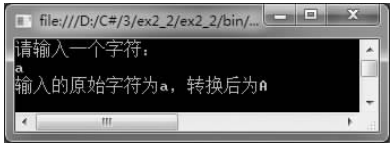

图 3-6　案例 3-5 的运行结果

【案例 3-6】 利用计算机随机出一道 100 以内的加、减法题目,并根据用户的答案给出"答对了"或"答错了"的提示信息。

1. 问题分析

本案例涉及随机数,C#中的随机数采用 Random 类来实现,具体实现过程如下。

```
Random rd=new Random();     //定义一个随机数对象 rd
```

之后采用 Random 类的 Next()方法生成位于一定范围内的随机数,举例如下。

```
int x=rd.Next(0, 100);
```

这样就生成了一个范围在 0～100 之间的随机数。

考虑到本案例中所有的操作数(假设为 a 和 b)是随机生成的,而且是做加法还是做减法也不固定,所以操作符(加减法)也由随机数来设定,当取得的随机数可以被 2 整除时,将计算 a+b,否则计算 a－b。

本案例将用到选择结构的嵌套。首先外层选择结构判断是加法还是减法操作,之后在选择结构的内部判断用户给出的答案是否正确。算法的流程如图 3-7 所示。

2. 操作步骤

(1) 新建一个控制台应用程序,并在 Main()方法中添加如下代码。

```
int a, b, c;                //a 和 b 代表两个操作数,c 代表操作符
int result;                 //result 代表用户输入的结果
Random rd=new Random();     //定义一个随机数对象 rd
a=rd.Next(0, 100);          //生成 0~100 之间的随机数
b=rd.Next(0, 100);
c=rd.Next(0, 50);           //生成 0~50 之间的随机数
```

```
//首先判定做加法还是减法,如能被 2 整除,则做加法,否则做减法
if (c % 2==0)
{
    Console.WriteLine("请计算: {0}+{1}=?",a,b);
    result=Convert.ToInt32(Console.ReadLine());
    if(result ==a+b)
        Console.WriteLine("哈哈,答对了,恭喜你!");
    else
        Console.WriteLine("答错了,再努力呀!");
}
else
{
    Console.WriteLine("请计算: {0}-{1}=?",a,b);
    result=Convert.ToInt32(Console.ReadLine());
    if(result ==a-b)
        Console.WriteLine("哈哈,答对了,恭喜你!");
    else
        Console.WriteLine("答错了,再努力呀!");
}
Console.ReadLine();
```

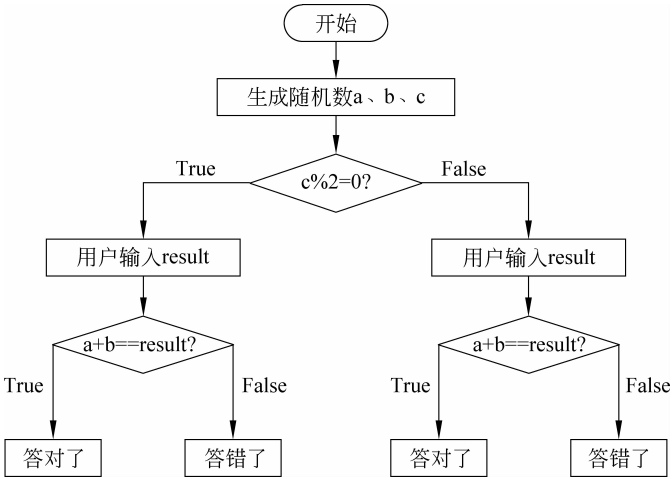

图 3-7　案例 3-6 的流程图

(2) 执行程序,程序运行结果如图 3-8 所示。

【案例 3-7】 输入一个年份和月份,打印出该月份有多少天。

1. 问题分析

一年 12 个月中的天数共有如下 4 种情况。

月份 1、3、5、7、8、10、12: 31 天;

月份 4、6、9、11: 30 天;

月份 2: 又分为两种情况,一是闰年的 2 月,为 29 天,二是平年的 2 月为 28 天。

图 3-8　案例 3-6 的运行结果

本案例可以使用 switch 结构实现，switch 结构内部分为 12 种情况。其中测试表达式分别为 1、3、5、7、8、10、12 的分支对应同一个操作，即天数为 31 天；测试表达式分别为 4、6、9、11 的分支也对应同一个操作，即天数为 30 天。而测试表达式为 2 的情况则复杂一些，要根据给定的年份判断天数是 28 天还是 29 天，所以其内部需要嵌入一个双分支 if 结构。

2. 操作步骤

(1) 新建一个控制台应用程序，并在 Main()方法中添加如下代码。

```
int y,m,n=0;     //y 代表年份,m 代表月份,n 代表天数
Console.WriteLine("请输入一个年份: ");
y=Convert.ToInt32(Console.ReadLine());
Console.WriteLine("请输入一个月份: ");
m=Convert.ToInt32(Console.ReadLine());
switch (m)
{
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:   n=31; break;
    case 2:
        if (y % 4==0 && y % 100 !=0 || y % 400==0)     //判断是否是闰年
        {
            n=29;
        }
        else
        {
            n=28;
        }
        break ;
    case 4:
    case 6:
    case 9:
    case 11: n=30; break;
    default :
```

```
        Console.WriteLine("您输入的月份有误!");break ;
}
Console.WriteLine("{0}年{1}月的天数为{2}天!", y, m, n);
Console.ReadLine();
```

(2) 执行程序,运行结果如图 3-9 所示。

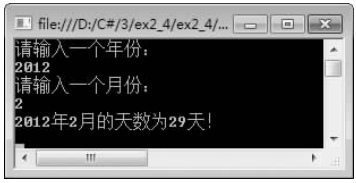

图 3-9　案例 3-7 的运行结果

实践提高

(1) 猜随机数游戏。产生一个 0～9 之间的随机数,请用户猜,如果成功,则显示“猜对了”,否则显示“猜错了”,运行界面如图 3-10 所示。

提示　定义一个随机数对象,并使用其 Next()方法产生 0～9 之间的随机数。

(2) 商店开展促销活动,如果顾客购买的商品金额超过 5000,折扣为 8.5 折;超过 3500,折扣为 9 折;超过 2000,折扣为 9.5 折;超过 1000,折扣为 9.8 折,其他没有折扣。试输入一个应缴金额,计算其实缴金额和优惠的金额数,运行结果如图 3-11 所示。

图 3-10　运行结果

图 3-11　运行结果

提示　采用多分支 if 语句实现。

实验 3.3　循环结构程序设计

实验目的

- 熟练掌握 3 种循环语句的使用。

- 掌握 foreach 语句的使用。
- 加深理解循环的概念,掌握循环的规则及其执行过程。
- 掌握 break 和 continue 语句的使用。
- 掌握常用的循环算法。

实验内容

【案例 3-8】 分别用 3 种循环语句(for、while、do-while)设计程序,计算并输出 1～10 的平方值,并对 3 种循环语句进行比较。

1. 问题分析

设计循环程序的算法,重要的是考虑以下 3 点。

第一,何时进入循环。

第二,循环中做什么。

第三,何时退出循环。

其中,第一个问题是考虑循环变量的初始值是多少,本例中要计算 1～10 的平方,所以循环将在 1～10 之间进行,循环变量的初值为 1。

第二个问题解决循环体中都实现哪些功能,也就是循环体的内容。本例中将计算循环变量的平方,并输出。

第三个问题解决的是循环条件的设置问题。本例中,当循环变量的值大于 10 时,跳出循环,所以循环退出的条件是关系表达式 i<=10 不再成立。

根据以上分析,分别写出采用 3 种循环语句实现本案例的程序。

2. 操作步骤

(1) 用 for 语句实现的程序代码:

```
int i;
for (i=1; i<=10; i++)
{
    Console.WriteLine("{0,2}的平方为: {1,3}", i, i * i);
}
Console.ReadLine();
```

(2) 用 while 语句实现的程序代码:

```
int i=1;
while(i<=10)
{
    Console.WriteLine("{0,2}的平方为: {1,3}", i, i * i);
    i++;
}
Console.ReadLine();
```

(3) 用 do-while 语句实现的程序代码：

```
int i=1;
do
{
    Console.WriteLine("{0,2}的平方为: {1,3}", i, i * i);
    i++;
} while (i<=10);
Console.ReadLine();
```

(4) 执行程序，程序运行结果如图 3-12 所示。

图 3-12　案例 3-8 的运行结果

说明

任何一个循环结构的程序既可以使用 for 语句实现，也可以使用 while 语句和 do-while 语句实现。区别在于使用哪种语句更合适。for 语句适合实现循环次数已知的循环；而 while 和 do-while 则适合次数未知，但循环条件很明确的循环。

【案例 3-9】 输出 100～200 之间能同时被 3 和 5 整除的数。

1. 问题分析

这是一个穷举算法的题目，逐一测试 100～200 之间的所有数字，看其是否能同时被 3 和 5 整除，如果符合条件，输出即可。

可以使用 for 语句实现以上算法。循环变量的初值为 100，循环条件为 i＜＝200，步长值为 1，即 i＋＋；。

2. 操作步骤

(1) 新建一个控制台应用程序，并在 Main()方法中添加如下代码。

```
int n;
for (n=100; n<=200; n++)
{
    if (n % 3==0 && n % 5==0)
    {
        Console.WriteLine("{0} 能同时被 3 和 5 整除。",n);
    }
}
Console.ReadLine ();
```

图 3-13　案例 3-9 的运行结果

(2) 执行程序，输出结果如图 3-13 所示。

说明

穷举法是计算机编程中的常用算法，一般需要程序设计人员确定以下两个问题。

① 穷举范围；

② 需要筛选出符合什么条件的值。

只要确定这两个问题，穷举法的程序是很容易编写的。

【案例 3-10】 输入一个任意的自然数，把它反序输出，例如输入 12345，输出为 54321。

1. 问题分析

因为输入的是任意自然数，所以不能确定其位数，也就不能确定循环执行的次数，可以考虑使用 while 循环语句实现。因无法确定位数，所以只能将其最低位取出来，取出最低位的表达式如下。

```
t=s%10;     //s 代表这个自然数,t 代表最低位
```

例如，s 为 123，则 t 为 3。

之后，用 s 除以 10，即：

```
s=s/10;
```

这样，s 的值就变为 12，再重复前面取余数(最低位)的操作，直到 s＞0 条件不再成立为止。

具体算法描述如下。

① 定义整型变量 s，用于存储输入的自然数；定义整型变量 n，存放 a 的反序数，n 的初值为 0；定义 t，表示每次从 s 中取出的余数。

② 输入一个自然数，赋值给变量 s。

③ 若 s＞0，执行第④步，否则执行第⑦步。

④ t＝a％10。

⑤ n＝n＊10＋t。

⑥ s＝s/10，并返回第③步。

⑦ 输出 n。

2. 操作步骤

(1) 新建一个控制台应用程序，并在 Main()方法中输入如下代码。

```
int s, n=0, t,s1;
Console.WriteLine("请输入一个自然数: ");
s=Convert.ToInt32(Console.ReadLine());
s1=s;          //将 s 的原始值用 s1 保存
while (s >0)
{
    t=s % 10;
    n=n * 10+t;
    s=s / 10;
}
Console.WriteLine("您输入的自然数{0},其反序数字为: {1}", s1, n);
Console.ReadLine();
```

(2) 执行程序，运行结果如图 3-14 所示。

【案例 3-11】 计算 $s=1+x-\frac{x^2}{2!}+\frac{x^3}{3!}-\frac{x^4}{4!}+\cdots+(-1)^{n-1}\frac{x^n}{n!}$，直到 $\left|\frac{x^n}{n!}\right|<10^{-6}$

图 3-14　案例 3-10 的运行结果

为止。若 $x=2.5$，则结果为 1.917915。

1. 问题分析

对于本案例，需要解决以下两个问题才能编写正确的程序。

① 累加项的通用表达式

这是一个累加问题，累加问题的通用表达式如下。

```
s=s+t;
```

其中，s 代表累加和，t 为累加项。在累加算法中，累加项 t 有时是一个简单变量，有时是一个复杂表达式。当 t 为一个复杂表达式时，首先要考虑的就是找出规律，写出 t 的通用表达式。

从给定题目中可以看到，从第 3 项开始，t 就以某种规律出现，规律如下。

$$t=\frac{x^n}{n!}$$

考虑将前两项排除在循环体外，t 的初值为 x，从第 3 项开始，t 的表达式可以写成如下形式。

$$t=t*x/2$$

即

$$t=\frac{x*x}{2}$$

则第 4 项中 t 的表达式如下。

$$t=t*x/3$$

即

$$t=\frac{x*x}{2}*\frac{x}{3}$$

所以，t 的通用表达式应如下所示。

```
t= t * x/n     //n 是不断变化的
```

但是，还需注意另一个问题：每一个累加项的正负号是不断变化的，你能找出规律，并写成通用表达式吗？试一试！

② 采用什么循环及循环条件的设置。

因为不知道循环多少次，所以采用 for 循环语句不合适，可以考虑采用 do-while 循环，循环条件如下。

```
Math.Abs(t)>=1e-6
```

其中，Abs()为 Math 类中求绝对值的方法。

2. 操作步骤

(1) 新建一个控制台应用程序，并在 Main()方法中输入如下代码。

```
double s,x,t;       //s 代表累加和,t 代表累加项
int n=1;
Console.WriteLine("请输入 x 的值: ");
x=Convert.ToDouble(Console.ReadLine());
s=1+x;
t=x;
do
{
    n++;
    t=t * (-1) * x / n;
    s=s+t;
} while (Math.Abs(t)>=1e-6);
Console.WriteLine("当 x 的值为{0}时,累加和 s 的值为{1}", x, s);
Console.ReadLine();
```

(2) 执行程序，运行结果如图 3-15 所示。

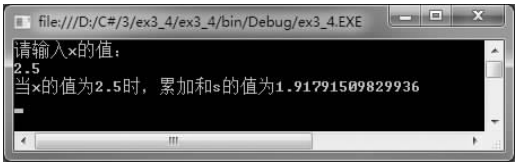

图 3-15　案例 3-11 的运行结果

【案例 3-12】 求两个正整数的最大公约数和最小公倍数。

1. 问题分析

求两个数最大公约数的方法有很多种，下面介绍以下两种算法。

(1) 算法一：最简单也最容易理解——穷举法。例如，有两个正整数 m 和 n，保证 m＞n，设定一个变量 t，t＝n。计算表达式：m％t 和 n％t，如果都为 0，则 t 就是两者的最大公约数；如不能整除，将 t 的值－1，重复 m％t 和 n％t 操作，直到找到一个数字，使 m％t 和 n％t 的结果都为零为止。

(2) 算法二：辗转相除法。辗转相除法的思想是：假设两个正整数 m 和 n，保证 m＞n，将计算 m％n 的结果放在变量 t 中，如果 t＝0，则 n 就是两者的最大公约数；如果 t 不为零，则令：

```
m=n
n=t
```

之后再次执行

```
t=m%n
```

并判断 t 是否为零。如不为 0,重复以上过程。

最小公倍数的公式如下。

```
m*n                    //最大公约数
```

2. 操作步骤

(1) 用穷举法实现求最大公约数和最小公倍数的代码如下。

```
int m, n, t;
Console.WriteLine("请输入两个正整数: ");
m=Convert.ToInt32(Console.ReadLine());
n=Convert.ToInt32(Console.ReadLine());
if (m<n)               //保证 m>n
{
    int k=m;
    m=n;
    n=k;
}
t=n;
//只要用 m 和 n 去除 t,只要有一个不能被整除,就将 t 减 1,直到都能整除,t 就是最大公约数
while (m % t !=0 || n % t !=0)
{
    t--;
}
Console.WriteLine("{0}和{1}的最大公约数为{2}", m, n, t);
Console.WriteLine("{0}和{1}的最小公倍数为{2}", m, n, m*n / t);
Console.ReadLine();
```

(2) 辗转相除法实现求最大公约数和最小公倍数的代码下。

```
int m, n, t;
int a, b;              //a 和 b 将用来保留 m 和 n 的原始值,以计算最小公倍数
Console.WriteLine("请输入两个正整数: ");
m=Convert.ToInt32(Console.ReadLine());
n=Convert.ToInt32(Console.ReadLine());
if (m<n)               //保证 m>n
{
    int k=m;
    m=n;
    n=k;
}
a=m; b=n;
t=m%n;
//只要用 m 和 n 去除 t,只要有一个不能被整除,就将 t 减 1,直到都能整除,t 就是最大公约数
```

```
while (t!=0)
{
    m=n;
    n=t;
    t=m % n;
}
Console.WriteLine("{0}和{1}的最大公约数为{2}", a, b, n);
Console.WriteLine("{0}和{1}的最小公倍数为{2}", a, b, a * b / n);
Console.ReadLine();
```

(3) 执行程序,运行结果如图 3-16 所示。

图 3-16　案例 3-12 的运行结果

【案例 3-13】 对一个班 15 名学生的成绩进行处理,要求如下。

① 输出所有学生的成绩。

② 计算最高分、最低分和平均分。

1. 问题分析

为了便于操作,考虑将这 15 个成绩放在数组中,之后利用 foreach 语句对数组中的元素进行遍历。但需要注意的是:foreach 语句只可以用于对数组元素进行遍历,并不能修改数组元素的值,所以输入 15 个元素的操作仍由 for 语句控制进行。

2. 操作步骤

(1) 新建控制台应用程序,并在 Main()方法中输入如下代码。

```
int[] a=new int[15];
int max, min;
float avg=0;            //分别代表最高分,最低分和平均分
int i;
//利用 for 循环输入学生成绩
for (i=0; i<15; i++)
{
    Console.WriteLine("请输入第{0}个学生的成绩", i+1);
    a[i]=Convert.ToInt32(Console.ReadLine());
}
//利用 foreach 求最高分、最低分和平均分
max=min=a[0];           //假设第 0 个元素既是最高分也是最低分
foreach(int x in a)
{
    if (max<x) max=x;
    if (min>x) min=x;
    avg+=x;
}
avg=avg / 15.0f;
Console.WriteLine ("这 15 个学生的成绩为: ");
foreach(int x in a)
```

```
{
    Console.Write("{0}   ",x);
}
Console.Write("\n");
Console.WriteLine("这 15 个学生的平均成绩为：{0}", avg);
Console.WriteLine("这 15 个学生中的最高分为：{0}", max);
Console.WriteLine("这 15 个学生中的最低分为：{0}", min);
Console.ReadLine();
}
```

(2) 执行程序，运行结果如图 3-17 所示。

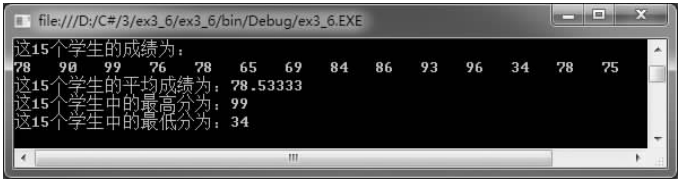

图 3-17　案例 3-13 的运行结果

【案例 3-14】 一个球从 100 米高空自由落下，每次落地后反弹回原来高度的一半，再落下，求它从开始落地直到 10 次反弹后落地时共经过了多少米，每一次反弹多高？

1. 问题分析

这是一个递推问题，已知条件为高度 $h=100$，每次反弹回原来的一半，即 $h/2$，也就是说 $h/2$ 将是下一次的高度，所以每一次的高度表达式如下。

```
h=h/2;
```

要想计算从最开始一直到第 10 次反弹经历了多少米，所用的表达式如下。

```
s= s+ 2 * h
```

之所以是 $2*h$，是因为球反弹上去后还要下落。

2. 操作步骤

(1) 新建控制台应用程序，并在 Main()方法中输入如下代码。

```
float h=100f;
float s=h;
int i;    //下落的次数
for (i=1; i<=10; i++)
{
    Console.WriteLine("第{0}次反弹的高度为：{1}米", i, h / 2);
    h=h / 2;
    s=s+2 * h ;
}
Console.WriteLine("第 10 次反弹后落地，共走过了{0}米", s);
Console.ReadLine();
```

(2) 执行程序,运行结果如图 3-18 所示。

图 3-18　案例 3-14 的运行结果

实践提高

(1) 计算 1!+2!+3!+…+7!。

提示　这是一个典型的累加算法,其核心代码如下。

```
s=s+t
```

本题中,第一次累加时 t 的值为 1;第二次累加时 t 的值为 2!,即 1 * 2;第三次累加时 t 的值为 3!,即 1 * 2 * 3,所以,可以给出 t 的通用表达式如下。

```
t=t * i
```

其中 i 为循环变量(1～7)。

(2) 编程计算：aaaaa…a-…-aaaa-aaa-aa-a,其中第一个为 n 个 a,a 的值和 n 的值由键盘输入。例如,n 的值为 5,a 的值为 2,则表达式为：

```
22222-2222-222-22-2
```

提示　可以使用两个 for 语句嵌套实现,第一个 for 语句用来构造 n 个 a 形成的数字,例如 5 个 2：22222;第二个 for 语句实现 22222-2222-222-22-2 的操作,其本质还是一个累加算法。

(3) 编程计算 $1-\frac{1}{2}-\frac{1}{3}-\cdots-\frac{1}{m}$ 的值。

提示　用 for 语句实现,累加算法。

实验 3.4　多重循环程序设计

实验目的

- 掌握多重循环的使用。

- 掌握多重循环中程序的执行过程。
- 掌握多重循环的常用算法。

实验内容

【案例 3-15】 有 3 个正整数 a、b、c，已知 $a>b>c>0$，且 $a+b+c<100$。求满足 $\frac{1}{a^2}+\frac{1}{b^2}=\frac{1}{c^2}$ 的 a、b、c 共有多少组？

1. 问题分析

题目中已知的条件有 3 个：

① $a>b>c>0$

② $a+b+c<100$

③ $\frac{1}{a^2}+\frac{1}{b^2}=\frac{1}{c^2}$

从这 3 个条件中并不能推断出 a、b、c 的具体值，可以考虑使用穷举法，对 a、b、c 逐一找出所有的组合，并判断是否符合以上 3 个条件。

程序结构采用三重循环，三层循环的循环范围均在 1～99 之间。

需要注意：整数相除的结果仍是整数，所以 $1/a^2$ 等表达式不能简单地写成 $1/(a*a)$。

2. 操作步骤

(1) 新建控制台应用程序，并在 Main()方法中输入如下代码。

```
int a, b, c;
for(a=1;a<=99;a++)
    for(b=1;b<=99;b++)
        for (c=1;c<=99; c++)
        {
            if (a >b && b >c & a+b+c<100 && 1.0 / (a * a)+1.0 / (b * b)
                ==1.0 / (c * c))
            Console.WriteLine("符合条件的 a、b、c 的值有{0},{1},{2}", a, b, c);
        }
Console.ReadLine();
```

(2) 执行程序，运行结果如图 3-19 所示。

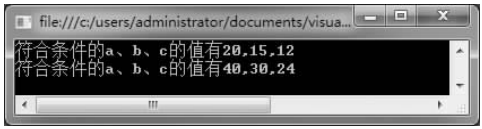

图 3-19　案例 3-15 的运行结果

说明

对于多重循环，循环的次数一般是每重循环循环次数的乘积，所以可以视情况优化

(减少循环次数),并将循环次数最多的循环尽量放在内层。

【案例 3-16】 求 100 以内所有素数之和。

1. 问题分析

本案例要用双重循环实现,外循环实现累加所有素数,其本质是一个累加算法,即:

```
s=s+t;
```

其中 t 就是 100 以内的素数,所以外层循环可以使用一个 for 语句实现,循环范围为 2～100。但是在计算累加之前,必须判断这些数字中哪些是素数。所以累加之前需要再使用一个循环语句来判断该数字是否是素数。

程序的结构如下。

```
外层循环(循环变量 x,范围为 2~100)
{
    内层循环(循环变量 i,范围为 2~x-1)
    {

    }
    如果 x 是素数,则计算
    s=s+x
}
```

2. 操作步骤

(1) 新建一个控制台应用程序,并在 Main()方法中输入如下代码。

```
int x;
int i;
int s=0;            //s 是所有素数的累加和
for (x=2; x<=100; x++)
{
    for (i=2; i<x; i++)
    {
        if (x %i==0) break;
    }
    if(x==i)        //如果是素数,则累加
        s+=x;
}
Console.WriteLine ("100 以内所有素数的和为{0}",s);
Console.ReadLine ();
```

(2) 执行程序,运行结果如图 3-20 所示。

说明

如何知道程序运行结果是否正确呢? 本例中,要想知道结果是否正确,就要把 100 以内所有素数找出来,逐一累加,这是一个非常烦琐的工作。其实,在调试过程中,对于一些大数的计算,如果不能明确知道结果,可以试着将较大的数改为较小的数,以验证程序的