第 3 章　C#程序控制

C#提供了对程序执行流程进行控制的语句，通过这些语句可以提高程序的灵活性，在不同的情况下执行不同程序。本章通过完成一个个任务来讲解 C#语言中选择、迭代和跳转等基本流程控制语句。

- if 语句和 if … else 语句
- if … else if …else 语句
- switch … case 语句
- while 语句和 do … while 语句
- for 语句
- break 语句、goto 语句、continue 语句和 return 语句

- 能够使用 if 语句编写简单的选择判断程序
- 能够使用 while、do…while 语句编写循环程序
- 能够使用 for 语句编写循环程序
- 能够使用 goto 语句、continue 语句在程序中有条件中断重复执行语句

3.1　选择语句

C#提供的选择语句主要分为两种类型，分别是 if 语句和 switch 语句，下面分别介绍选择语句的使用方法。

任务一　编写控制台应用程序判断是否应交个人所得税

编写控制台应用程序，根据输入的应发工资数和养老金等三金或四金数，判断是否应交个人所得税。要求在控制台程序中，按照要求分别输入应发工资数和养老金等三金或四金数，由应发工资数减去养老金等三金或四金数后求得全月应纳税所得额，若该数据大于 2000 元，

则显示“需要交个人所得税!”，否则不显示任何内容。

任务解决方案

（1）创建名为 Tax1 的控制台应用程序项目。

（2）在 Program.cs 中输入下列程序代码：

```
static void Main(string[] args)
{
    double pay, money, income;
    Console.WriteLine("请输入应发工资数：");
    pay = double.Parse(Console.ReadLine());
    Console.WriteLine("请输入养老金等三金或四金：");
    money = double.Parse(Console.ReadLine());
    income = pay - money;
    if (income > 2000)
    {
        Console.WriteLine("需要交个人所得税！");
    }
}
```

（3）按 F5 键运行程序，结果如图 3-1 所示。

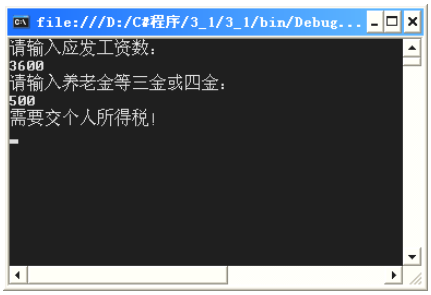

图 3-1　在控制台输入数据并显示结果

分 析 描 述

（1）运行程序后，根据提示信息，在控制台中输入应发工资数并按回车键，再输入养老金等三金或四金数并按回车键，程序首先通过 double.Parse()方法将输入的数据转换成双精度数据，存入变量 pay 和 money 中，再通过两数相减计算全月应纳税所得额，存入变量 income 中。

（2）执行 if 选择语句进行条件判断，当 if 后条件表达式“income > 2000”的值为 true 时，表示全月应纳税所得额大于 2000 元，执行语句 Console.WriteLine("需要交个人所得税！");，应用控制台输出显示结果；当“income > 2000”的值为 false 时，结束条件语句。

相 关 知 识

3.1.1　If 语句

If 语句实现单分支选择结构，语句格式如下：

```
If(表达式)
{
```

```
    语句块
}
```

执行过程说明如下：

如果表达式的逻辑值为 true，执行 If 语句控制的语句块；若表达式的值为 false，不执行 If 语句控制的语句块，执行语句块后的语句。If 语句流程图如图 3-2 所示。

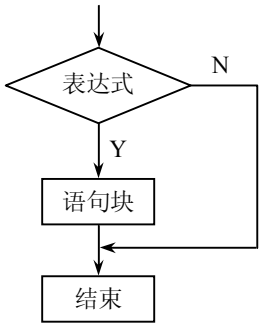

图 3-2　单分支选择结构流程图

任务二　编写 Windows 窗体应用程序判断是否应交个人所得税

问题描述

编写Windows窗体应用程序，根据用户在相应文本框中输入的应发工资数和养老金等三金或四金数，由应发工资数减去养老金等三金或四金数后求得全月应纳税所得额，并判断用户输入的应发工资和养老金等三金或四金数据是否为正数。若该数据大于2000元，则弹出消息框显示“需要交个人所得税!”，否则弹出消息框显示“不需要交个人所得税!”。

任务解决方案

（1）创建名为 Tax2 的 Windows 应用程序项目。

（2）选择 Windows 窗体设计器中的新建窗体，从“工具箱”的“公共控件”选项卡中分别选择 Label、TextBox 和 Button 等控件，按照图 3-2 所示的界面布局，用鼠标拖放到窗体的适当位置，并根据表 3.1 设置控件属性。

表 3.1　属性表

控件	属性	设置	控件	属性	设置
Label1	Name	lblPay	Button1	Name	btnJudge
	Text	应发工资：		Text	判断
Label2	Name	lblMoney	Button2	Name	btnClear
	Text	三金或四金：		Text	清除
TextBox1	Name	txtPay			
TextBox2	Name	txtMoney			

（3）双击“判断”按钮，打开代码编辑器，此时插入点已位于该按钮的 Click 事件处理

响应方法 btnJudge_Click()中，插入下列代码：

```
private void btnJudge_Click(object sender, EventArgs e)
{
    double pay, money, income;
    pay = double.Parse(txtPay.Text );
    money = double.Parse(txtMoney.Text );
    income = pay - money;
    if (income > 2000)
    {
        MessageBox.Show("需要交个人所得税！");
    }
    else
    {
        MessageBox.Show("不需要交个人所得税！");
    }
}
```

（4）按 F5 键运行程序，将显示 Windows 窗体应用程序窗口，如图 3-3 所示。在相应的文本框中分别输入应发工资和养老金等三金或四金数后，单击“判断”按钮，根据计算出的全月应纳税所得额判断是否达到交个人所得税的标准（大于 2000 元），弹出不同的消息框，如图 3-4 和图 3-5 所示。

图 3-3　在 Windows 窗体中输入数据

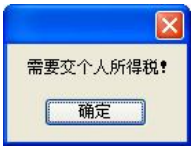

图 3-4　需要交个人所得税弹出消息框

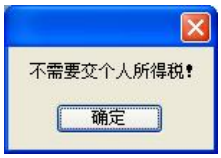

图 3-5　不需要交个人所得税弹出消息框

（5）修改 Tax2 的 Windows 应用程序项目，将“判断”命令按钮的 Click 事件程序代码修改如下：

```
private void btnJudge_Click(object sender, EventArgs e)
{
    double pay, money, income;
    pay = double.Parse(txtPay.Text );
    money = double.Parse(txtMoney.Text );
    income = pay - money;
```

```
        if (pay < 0 || money < 0)
        {
            MessageBox.Show("工资或保险金不能为负，请重新输入");
            txtPay.Text = "";
            txtMoney.Text = "";
            txtPay.Focus();
            return;
        }
        else
        {
            income = pay - money;
            if (income > 2000)
            {
                MessageBox.Show("需要交个人所得税！");
            }
            else
            {
                MessageBox.Show("不需要交个人所得税！");
            }
        }
    }
```

（6）再按 F5 键运行程序，并在“应发工资”和“三金或四金”任一文本框中输入一个负数，单击“判断”按钮，此时将弹出消息框，提示不能输入负数，如图 3-6 所示。

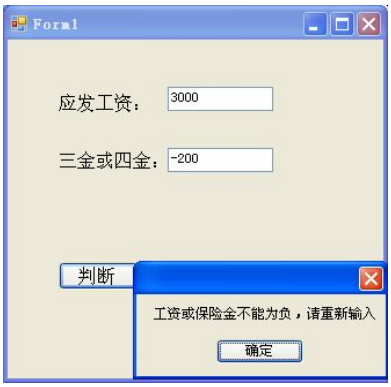

图 3-6　输入负数，弹出消息框提示

分析描述

（1）在程序中应用 if…else 语句，可以进行双选择判断，当 if 后面的条件表达式“income > 2000”值为 true 时，执行弹出消息框的语句，显示“需要交个人所得税!”；当条件表达式的值为 false 时，执行另一个语句，显示“不需要交个人所得税!”，实现了双分支选择的目的。

（2）修改程序，添加一个选择语句，判断输入的应发工资和三金或四金是否为负数，当外层 if…else 语句的逻辑表达式“pay < 0 || money < 0”的值为 false 时，再执行一个 if…else 语句，这种一层套一层的选择语句称为选择嵌套。

相 关 知 识

3.1.2　if…else 语句

if…else语句实现双分支选择结构，语句格式如下：

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

执行过程说明如下：

如果表达式的逻辑值为 True，执行 if 与 else 中间的语句块 1，若表达式的逻辑值为 false，执行 else 后面的语句块 2。if…else 语句流程图如图 3-7 所示。

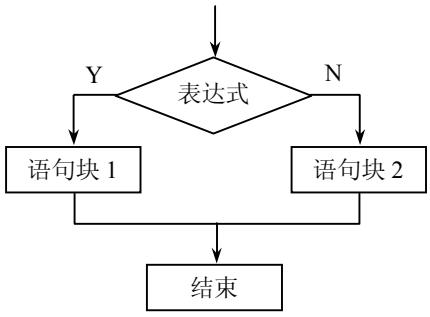

图 3-7　双分支选择结构流程图

任务三　编写 Windows 窗体应用程序计算个人所得税和实发工资

问 题 描 述

编写 Windows 窗体应用程序，根据输入的应发工资数和养老金等三金或四金数计算个人所得税和实发工资。首先要求用户输入的数据必须是正数，否则要求用户重新输入；然后再将输入的应发工资和养老金等三金或四金数按照下列公式计算个人所得税和实发工资。

计算公式：

应缴纳的个税＝[(应发工资–养老金等三金或四金) –2000]×税率–速算扣除数

其中含税级距、税率见表 3.2。

表 3.2　个人所得税税率表

含税级距	税率（%）	速算扣除数
不超过 500 元的	5	0
超过 500 元至 2,000 元的部分	10	25
超过 2,000 元至 5,000 元的部分	15	125

续表

含税级距	税率（%）	速算扣除数
超过 5,000 元至 20,000 元的部分	20	375
超过 20,000 元至 40,000 元的部分	25	1,375
超过 40,000 元至 60,000 元的部分	30	3,375
超过 60,000 元至 80,000 元的部分	35	6,375
超过 80,000 元至 100,000 元的部分	40	10,375
超过 100,000 元的部分	45	15,375

任务解决方案

（1）创建名为 Tax3 的 Windows 应用程序项目。

（2）选择新建窗体，从“工具箱”的“公共控件”选项卡中分别选择 Label、TextBox 和 Button 等控件，按照图 3-8 所示界面布局，用鼠标将控件拖放到新建窗体的适当位置，并根据表 3.3 设置控件属性。

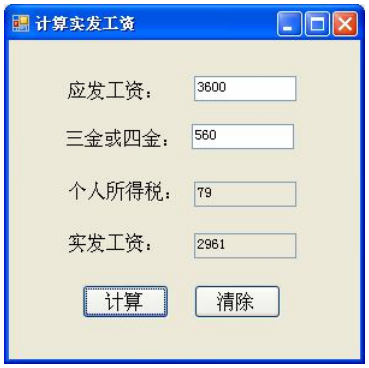

图 3-8　计算个人所得税和实发工资窗体

表 3.3　属性表

控件	属性	设置	控件	属性	设置
Form1	Name	frmCompute	TextBox1	Name	txtPay
	Text	计算实发工资	TextBox2	Name	txtMoney
Label1	Name	lblPay	TextBox3	Name	txtIncome
	Text	应发工资：		ReadOnly	true
Label2	Name	lblMoney	TextBox4	Name	txtReally
	Text	三金或四金：		ReadOnly	true
Label3	Name	lblIncome	Button1	Name	btnCompute
	Text	个人所得税：		Text	计算
Label4	Name	lblreally	Button2	Name	btnClear
	Text	实发工资：		Text	清除

（3）编写程序代码。

①双击“计算”按钮，打开代码编辑器，此时插入点已位于该按钮的 Click 事件处理响应方法 btnCompute_Click()中，插入下列代码：

```
private void btnCompute_Click(object sender, EventArgs e)
{
    double pay,money,income,incometax,really;
    pay = double.Parse(txtPay.Text );
    money = double.Parse(txtMoney.Text );
    if (pay < 0 || money <0)    //判断应发工资或养老金等是否为负
    {
        MessageBox.Show("工资或保险金不能为负，请重新输入");
        txtPay.Text = "";
        txtMoney.Text = "";
        txtPay.Focus();
        return;
    }
    income=pay-money- 2000;        //计算个人全月应纳税所得额
    if (income >0 && income <=500)
    {
        incometax =income * 0.05;          //计算个人所得税
    }
    else if (income > 500 && income <= 2000)
    {
        incometax = income * 0.1-25;
    }
    else if ( income > 2000 && income <= 5000)
    {
        incometax = income * 0.15-125;
    }
    else if (income > 5000 && income <=20000)
    {
        incometax = income * 0.2-375;
    }
    else if (income > 20000 && income <= 40000)
    {
        incometax = income * 0.25 - 1375;
    }
    else if (income > 40000 && income <=60000)
    {
        incometax = income * 0.3 - 3375;
    }
    else if (income > 60000 && income <= 80000)
    {
        incometax = income * 0.35 - 6375;
    }
    else if (income > 80000 && income <= 100000)
    {
        incometax = income * 0.4 - 10375;
    }
    else if (income > 100000)
    {
```

```
            incometax = income * 0.45 - 15375;
        }
        else
        {
            incometax = 0;
        }
        really = pay -money - incometax;     //计算实发工资
        txtIncome.Text = incometax.ToString();      //输出个人所得税
        txtReally.Text = really.ToString();      //输出实发工资
}
```

②双击“清除”按钮，在 btnClear_Click()方法中输入下列程序代码：

```
private void btnClear_Click(object sender, EventArgs e)
{
    txtPay.Text = "";
    txtMoney.Text = "";
    txtIncome.Text = "";
    txtReally.Text = "";
    txtPay.Focus();
}
```

（4）按 F5 键运行程序，显示 Windows 窗体应用程序窗口，输入应发工资和养老金等三金或四金数后，单击“计算”按钮，显示应交的个人所得税和实发工资，如图 3-8 所示。

分 析 描 述

（1）单击“计算”按钮，执行其 Click 事件处理程序，其中选择语句 if 后面的逻辑表达式“pay < 0 || money <0”表示 pay < 0 或 money <0，“||”表示逻辑或，只要该逻辑表达式中有一个表达式的值为 true，则执行 MessageBox.Show("工资或保险金不能为负，请重新输入");语句，弹出消息框要求重新输入数据。

（2）当输入的实发工资和三金或四金都为正数时，通过公式“pay-money-2000”计算个人全月应纳税所得额存入 income 变量中，再通过多选择语句 if…else if…else 判断全月应纳税所得额 income 属于哪个含税级距，再根据不同的含税级距以不同的税率和速算扣除数按照公式“income * 税率 - 速算扣除数”计算出个人所得税，并存入 incometax 变量中，最后算出实发工资。

（3）执行 if…else if…else 语句时，首先判断 if 后面的逻辑表达式“income >0 && income <=500”的值是否为 true，“&&”表示逻辑与，表示条件表达式“income >0”和条件表达式“income <=500”都为 true 时逻辑表达式值为 true，此时表示 income 值在 500≥income>0 范围内，则按照公式“income * 0.05”计算个人所得税；若逻辑表达式值为 false，则继续判断下一个 else if 后面逻辑表达式“income>500 && income <= 2000”的值，若为 true，则表示 income 值在 2000≥income>500 范围内，则按照公式“income * 0.1-25”计算个人所得税；若逻辑表达式值为 false，再继续判断下一个 else if 后面逻辑表达式，以此类推，当所有逻辑表达式都为 false，执行 else 后面的语句。

（4）单击“清除”按钮，执行该按钮的 Click 事件处理程序。执行语句 txtPay.Text = "";表示将文本框 txtPay 清空，执行语句 txtPay.Focus();表示该文本框 txtPay 得到光标。

相 关 知 识

3.1.3　if 语句的嵌套

选择结构的嵌套是一种常用的结构，它将多种选择结构嵌套成一个整体，常用格式如下：

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

执行过程说明如下：

首先判断最外层 if 选择语句中表达式 1 的逻辑值，若为 true，执行该 if 与 else 中间的语句块，该语句块中还包含另一个 if 选择语句，因此继续判断内层选择语句中表达式 2 的逻辑值，若为 true，则执行语句块 1，若为 false，执行 else 后面的语句块 2；若表达式 1 的逻辑值为 false，则执行语句块 3。if 语句的嵌套还有其他几种形式：

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

或：

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

3.1.4　if…else if…else 语句

if…else if…else 语句实现多分支选择结构，语句格式如下：

```
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
{
    语句块 n
}
```

执行过程说明如下：

如果表达式 1 的逻辑值为 true，则执行语句块 1，然后结束 if 语句；如果表达式 1 的逻辑值为 false，则判断表达式 2 的逻辑值，若为 true，则执行语句块 2，然后结束 if 语句；如果表达式 2 的逻辑值为 false，则继续判断其他表达式的逻辑值；如果所有表达式的逻辑值为 false，则执行语句块 n，然后结束 if 语句。if…else if…else 语句流程图如图 3-9 所示。

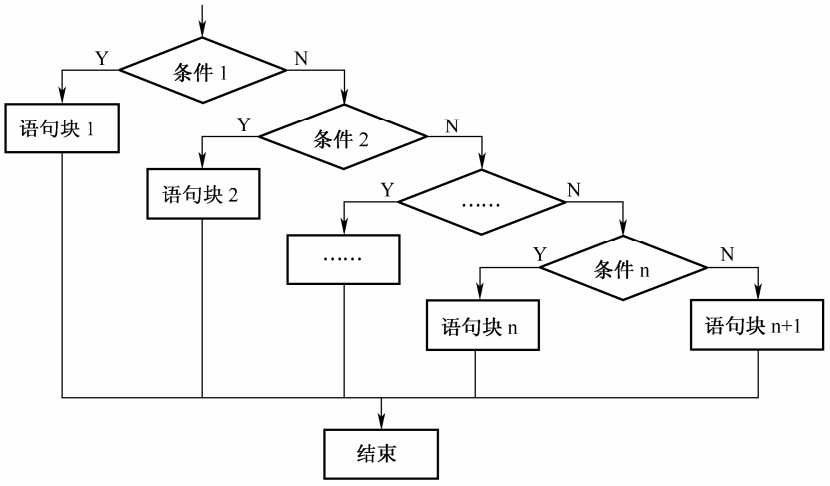

图 3-9　多分支选择结构流程图

任务四　简易计算器

问题描述

设计一个简易计算器，实现两个数的简易计算。要求在相应文本框输入两个操作数，再根据用户选择的运算符分别实现这两个操作数的加、减、乘、除运算，并在指定的文本框显示计算结果。

任务解决方案

（1）创建名为 Calculator 的 Windows 应用程序项目。

（2）添加控件并设置属性。选择新建窗体，按照图 3-10 所示界面进行布局设计，从“工

具箱”中的“Windows 窗体”选项卡中选择控件，用鼠标将控件拖放到新建窗体的适当位置，并根据表 3.4 设置控件属性。

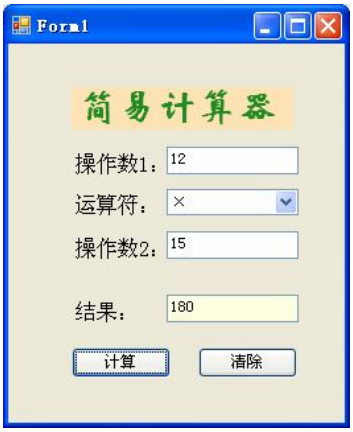

图 3-10　简易计算器

表 3.4　属性表

控件	属性	设置	控件	属性	设置
Label1	Name	lblTitle	TextBox1	Name	txtOp1
	Text	简易计算器	TextBox2	Name	txtOp2
	BackColor	255,255,192	TextBox3	Name	txtResult
	Font	华文新魏、粗体、二号		ReadOnly	true
	ForeColor	ForestGreen	ComboBox	Name	combOperation
Label2	Name	lblOp1		DropDownStyle	DropDownList
	Text	操作数 1：		Items	+、－、×、÷
Label3	Name	lblOperation			
	Text	运算符：	Button1	Name	btnCalculate
Label4	Name	lblOp2		Text	计算
	Text	操作数 2：	Button2	Name	btnClear
Label5	Name	lblResult		Text	清除
	Text	结果：			

（3）编写程序代码。

①双击“计算”按钮，打开代码编辑器，输入下列程序代码：

```
private void btnCalculate_Click(object sender, EventArgs e)
{
    double op1 = 0.0;
    double op2 = 0.0;
    double result = 0.0;
    string op;
    op1 = double.Parse(txtOp1.Text);
    op2 = double.Parse(txtOp2.Text);
```

```
        op = combOperation.SelectedItem.ToString();
        switch (op)
        {
            case "+":
                result = op1 + op2;
                break;
            case "－":
                result = op1 - op2;
                break;
            case "×":
                result = op1 * op2;
                break;
            default :
                result = op1 / op2;
                break;
        }
        txtResult.Text = result.ToString();
    }
```

②双击“清除”按钮，输入下列程序代码：

```
private void btnClear_Click(object sender, EventArgs e)
{
    txtOp1.Text = "";
    txtOp2.Text = "";
    combOperation.Text = "";
    txtResult.Text = "";
}
```

（4）测试程序。按 F5 键运行程序，显示 Windows 应用程序窗口，输入操作数 1 和操作数 2，选择相应的运算符，单击“计算”按钮，显示运算结果，如图 3-10 所示。

分析描述

（1）单击“计算”按钮，执行按钮的 Click 事件处理程序。首先将文本框 txtOp1 和文本框 txtOp2 中输入的数据通过 double.Parse 转换成双精度数据后赋值给双精度类型变量 op1 和 op2，将组合框中选择的运算符赋值给字符串类型变量 op。

（2）执行 switch 语句，将 switch 后括号中 op 的值依次与 case 语句中的常数值进行比较，若满足两值相等的条件，则执行该 case 后相应的语句块，然后执行跳转语句 break，结束 switch 语句；若两值不相等，则继续寻找下一个满足相等条件的 case 语句，当所有 case 语句都不满足条件时，执行 default 语句后的语句块。

因此执行 switch 语句时首先将 op 中的值与“+”比较，若 op 中存放的是“+”，则执行加法操作，然后执行跳转语句结束 switch 语句。

```
result = op1 + op2;
break;
```

若op中存放的不是“+”，则继续用op中的值与“－”比较，若op中存放的是“－”，则执行减法操作，然后执行跳转语句结束switch语句。

```
result = op1 - op2;
break;
```

若op中存放的不是“－”，则继续用op中的值与“×”比较，若op中存放的是“×”，则执

行乘法操作，然后执行跳转语句结束switch语句。

```
result = op1 * op2;
break;
```

若op中存放的不是“×”，而是“÷”，则执行default语句后的除法操作，然后执行跳转语句结束switch语句。

```
result = op1 / op2;
break;
```

（3）使用switch语句进行多项选择判断时，将switch后的表达式值计算出来，其值应该是下列几种类型的数据：char、字符串型或整型，当其值与case语句中的值相匹配时，执行case语句中的语句块，而且语句块中都要有跳转语句break；default语句是在所有case指定条件都不为true的情况下执行的操作。

相 关 知 识

3.1.5　switch 语句

switch 语句实现多分支选择结构，语句格式如下：

```
switch(表达式)
{
  case 常数表达式 1:
    {语句块 1}
     跳转语句
  case 常数表达式 2:
    {语句块 2}
     跳转语句
  ……
  defalut:
     {语句块 n+1}
     跳转语句
}
```

执行过程说明如下：

如果 switch 语句中表达式的值与常数表达式 1 的值相等，执行语句块 1，再执行跳转语句；若不相等，再判断表达式的值与常数表达式 2 的值是否相等，若相等，执行语句块 2，若不相等继续判断与下一个常数表达式是否相等，当与所有常数表达式都不相等时，则执行语句块 n+1。

3.1.6　条件运算符

?: 运算符为条件运算符，用于条件表达式中，根据布尔型表达式的值返回两个值中的一个，其为三元运算符，采用的是右结合的形式。

条件表达式形式如下：

表达式 1 ？ 表达式 2 : 表达式 3

执行过程说明如下：

首先计算表达式 1，如果表达式 1 的值为 true，则计算表达式 2，并将表达式 2 的值作为条件表达式的值；如果表达式 1 的值为 false，则计算表达式 3，并将表达式 3 的值作为条件表达式的值。

例如：

```
employ=workyears>1 ? "可应聘" : "不可应聘";
```

该语句表示当 workyears 中存放的值大于 1，条件表达式的值为“可应聘”，并将该字符串赋值给 employ 变量，否则将“不可应聘”字符串赋值给 employ 变量。

3.1.7 关系运算符

关系运算符是二元操作符，通过关系运算符把操作数连接起来，构成关系表达式，关系表达式的运算结果是布尔值。关系运算符见表 3.5。

表 3.5 关系运算符

关系运算符	功能	关系运算符	功能
==	等于	>=	大于或等于
!=	不等于	<	小于
>	大于	<=	小于或等于

例如：

```
int a,b;
a=5;
b=10
c=a>b;          //结果为 false
```

3.1.8 逻辑运算符

C#提供了几种逻辑运算符，通过逻辑运算符把操作对象连接起来的式子称为逻辑表达式，逻辑表达式的结果为布尔值。逻辑运算符见表 3.6。

表 3.6 逻辑运算符

运算符	功能	作用
&&	逻辑与	x && y，如果 x 和 y 的值均为 true，则 x && y 值为 true，否则为 false
||	逻辑或	x ||y，如果 x 或 y 的值为 false，则 x || y 值为 false，否则为 true
&	位与	x & y，如果 x 和 y 的值均为 true，则 x & y 值为 true，否则为 false。 如果 x 和 y 是整数，x & y 还可以执行位与
|	位或	x | y，如果 x 或 y 的值为 false，则 x | y 值为 false，否则为 true。 如果 x 和 y 是整数，x | y 还可以执行位或
!	逻辑非	!x，如果 x 的值为 true，则!x 的值为 false；如果 x 的值为 false，则!x 的值为 true
^	逻辑异或	x^y，如果 x 的值为 true 而 y 的值为 false，或者 x 的值为 false 而 y 的值为 true，则 x^y 的值为 true，否则为 false。 如果 x 和 y 是整数，x^y 还可以执行位异或

3.2 迭代语句

C#提供的迭代语句有 while 语句、do while 语句、for 语句、foreach 语句，这些语句可以实现重复循环执行程序代码，下面分别介绍前三种语句的使用方法，foreach 语句的使用方法在数组章节中介绍。

任务五　计算某人一年公积金账户余额

问题描述

编写 Windows 窗体应用程序，根据输入的某人工资、公积金交存比例和住房贷款月还款金额，计算出他一年公积金账户余额。

任务解决方案

（1）创建名为 Fund 的 Windows 应用程序项目。

（2）添加控件并设置属性。选择新建窗体，按照图 3-11 所示的界面进行布局，向新建的窗体添加控件，并根据表 3.7 设置控件属性。

图 3-11　计算公积金账户余额窗体

表 3.7　属性表

控件	属性	设置	控件	属性	设置
Form1	Name	frmBalance	TextBox1	Name	txtPay
	Text	计算公积金账户余额	TextBox2	Name	txtScale
Label1	Name	lblPay	TextBox3	Name	txtLoan
	Text	月工资：	TextBox4	Name	txtSum
Label2	Name	lblScale		ReadOnly	true
	Text	公积金比例：	Button1	Name	btnCal
Label3	Name	lblLoan		Text	计算
	Text	月还贷金额：	Button2	Name	btnClear
Label4	Name	lblSum		Text	清除
	Text	一年公积金余额：			

（3）编写程序代码。

①双击“计算”按钮，打开代码编辑器，输入下列代码：

```
private void btnCal_Click(object sender, EventArgs e)
{
```

```
        double sum,pay,scale,loan;
        pay=double.Parse (txtPay.Text );                    //将工资存入变量
        scale =double.Parse (txtScale.Text );               //将公积金交存比例存入变量
        loan = double.Parse(txtLoan.Text );                 //将每月公积金还贷金额存入变量
        sum = 0;
        int i = 1;
        while (i <=12)                                      //循环 12 次表示 12 个月
        {
        sum +=2*pay*scale/100-loan;                         //计算每月公积金余额并累加
        i=i +1;
        }
        txtSum.Text = sum.ToString();                       //输出一年公积金余额
    }
```

②双击“清除”按钮，输入下列代码：

```
    private void btnClear_Click(object sender, EventArgs e)
    {
        txtPay.Text = "";
        txtScale.Text = "";
        txtLoan.Text = "";
        txtSum.Text = "";
    }
```

（4）测试程序。按 F5 键运行程序，显示 Windows 窗体应用程序窗口。输入月工资数、公积金交存比例和每月的还贷金额，单击“计算”按钮，显示一年公积金账户余额，如图 3-11 所示。

（5）对“计算”按钮单击事件处理程序进行修改。双击“计算”按钮，打开代码编辑器，将程序代码修改为下列代码：

```
    private void btnCal_Click(object sender, EventArgs e)
    {
        double sum,pay,scale,loan;
        pay=double.Parse (txtPay.Text );                    //将工资存入变量
        scale = double.Parse(txtScale.Text);                //将公积金交存比例存入变量
        loan = double.Parse(txtLoan.Text);                  //将每月公积金还贷金额存入变量
        sum = 0;
        int i = 1;
        do
        {
            sum += 2 * pay * scale / 100 - loan;            //计算每月公积金余额并累加
            i = i + 1;
        }
        while (i <= 12);                                    //循环 12 次表示 12 个月
        txtSum.Text = sum.ToString();                       //输出一年公积金余额
    }
```

（6）测试程序。按 F5 键运行程序，按图 3-11 所示输入相应数据。

（7）再次对“计算”按钮单击事件处理程序进行修改。双击“计算”按钮，打开代码编辑器，将程序代码修改为下列代码：

```
    private void btnCal_Click(object sender, EventArgs e)
    {
        double sum,pay,scale,loan;
        pay=double.Parse (txtPay.Text );                    //将工资存入变量
```

```
        scale = double.Parse(txtScale.Text);          //将公积金交存比例存入变量
        loan = double.Parse(txtLoan.Text);            //将每月公积金还贷金额存入变量
        sum = 0;
        for (int i = 1; i <= 12;i ++ )                //循环 12 次表示 12 个月
        {
            sum += 2 * pay * scale / 100 - loan;      //计算每月公积金余额并累加
        }
        txtSum.Text = sum.ToString();                 //输出一年公积金余额
    }
```

（8）再测试程序。按 F5 键运行程序，按图 3-11 所示输入相应数据。

分析描述

（1）在第一段“计算”按钮的单击事件处理程序中，在执行循环语句 while 前要给循环变量 i 赋初值，语句为 int i = 1;，执行循环语句时首先判断关系表达式“i <=12”的值是否为 true，为 true 则执行循环体中语句 sum+=2*pay*scale/100-loan;计算每月公积金余额并进行累加，然后执行语句 i=i +1;使循环变量递增，再转到循环起始语句 while 继续执行，直到关系表达式的值为 false，停止执行循环。循环语句共执行循环体 12 次，表示 12 个月，把每个月的公积金余额计算出来后进行累加，得到一年的公积金余额。

（2）由于在窗体中输入的公积金交存比例是正整数而不是百分比，因此在计算每月公积金时，要将输入的公积金交存比例数除以 100，如：scale/100，将其转换成百分数参加计算；公积金分为两部分，自己交一份，工作单位按同等金额交一份，因此公积金存款为交存的公积金金额的 2 倍，公式为：2*pay*scale/100。

（3）将“计算”按钮单击事件处理程序中的循环语句改为 do…while 后，执行循环时首先执行循环体，再判断关系表达式的值“i <= 12”是否为 true，当为 true 时，再转到循环的起始语句 do 语句继续执行循环体，当关系表达式的值为 false 时，结束循环。

（4）while 循环与 do…while 循环的区别在于前一种循环结构是先判断后执行循环体，后一种循环结构是先执行循环体后判断。一般情况下这两种循环结构没多大区别，只有在一种情况下是有区别的，就是在一开始关系表达式的值为 false 时，前一种结构是一次循环体都不执行，而后一种结构是执行了一次循环体才结束。

（5）再次修改“计算”按钮的单击事件处理程序，将其中的循环语句改为 for 循环结构后，执行循环时首先执行 for 语句中表达式 int i = 1;，再判断关系表达式 i <= 12;的值，为 true，则执行循环体，执行完循环体，再执行 for 语句中的 i++，再次判断关系表达式 i<=12;的值是否为 true，当为 true 时，继续执行循环体，为 false 时，结束循环。

相关知识

3.2.1　while 语句

while 语句格式如下：

```
while (表达式)
{
    循环体
}
```

执行过程说明如下：

如果 while 语句中表达式的值为 true，则执行循环体中的程序代码，执行完循环体，控制将转到 while 语句的开头，再次执行 while 语句；如果 while 语句中表达式的值为 false，则结束 while 语句的执行。while 语句的流程图如图 3-12 所示。

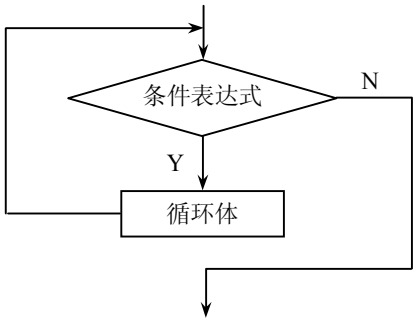

图 3-12　while 循环结构

任务六　设计一个计算阶乘和 e 的指数幂的计算器

问题描述

使用 Windows 窗体应用程序设计一个计算器，要求通过按钮输入数据，单击“n!”按钮计算出阶乘值，单击“e^”按钮则计算出 e 的指数幂，如图 3-13 所示。

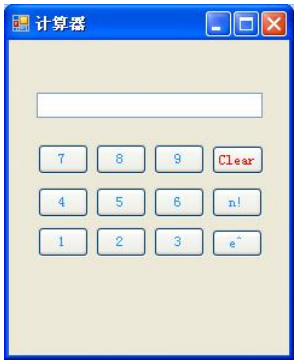

图 3-13　计算阶乘和 e 的指数幂的计算器

任务解决方案

（1）创建名为 Eindex 的 Windows 应用程序项目。

（2）添加控件并设置属性。选择新建窗体，按照图 3-13 所示的界面进行布局，向新建的窗体添加 TextBox 和 Button 控件，并根据表 3.8 设置控件属性。

（3）编写程序代码。

① 双击“1”按钮，打开代码编辑器，在其 Click 事件处理程序中输入下列代码：

```
private void btn1_Click(object sender, EventArgs e)
{
```

```
        this.txtResult.Text += this.btn1.Text;
    }
```

表 3.8　属性表

控件	属性	设置	控件	属性	设置
Form1	Name	frmCounter	Button6	Name	btn6
	Text	计算器		Text	6
TextBox1	Name	txtResult	Button7	Name	btn7
	TextAlign	Right		Text	7
Button1	Name	btn1	Button8	Name	btn8
	Text	1		Text	8
Button2	Name	btn2	Button9	Name	Btn9
	Text	2		Text	9
Button3	Name	btn3	Button10	Name	btnN
	Text	3		Text	n!
Button4	Name	btn4	Button11	Name	btnE
	Text	4		Text	e^
Button5	Name	btn5	Button12	Name	btnClear
	Text	5		Text	Clear

② 分别双击“2”、“3”、“4”、“5”、“6”、“7”、“8”、“9”按钮，打开代码编辑器，在其 Click 事件处理程序中输入和步骤①类似的代码。

例如：在按钮“2”的 Click 事件处理程序 btn2_Click()中输入代码如下：

```
this.txtResult.Text += this.btn2.Text;
```

③ 双击“Clear”按钮，打开代码编辑器，在其 Click 事件处理程序中输入下列代码：

```
private void btnClear_Click(object sender, EventArgs e)
{
    this.txtResult.Text = string.Empty;
    this.txtResult.Focus();
}
```

④ 双击“n!”按钮，打开代码编辑器，在其 Click 事件处理程序中输入下列程序代码：

```
/// <summary>
/// 求阶乘
/// </summary>
private void btnN_Click(object sender, EventArgs e)
{
    int n;
    try
    {
        n = int.Parse(txtResult.Text);
    }
    catch (System.FormatException)   //阶乘输入为小数时的错误异常处理
    {
        this.txtResult.Text = "阶乘必须是整数!";
        return;
```

```
        }
        if (n <= 0)     //输入阶乘为负数时的处理
        {
            //MessageBox.Show("阶乘不能为负数！");
            this.txtResult.Text = "阶乘不能为负数!";
        }
        else
        {
            int m = 1;
            try
            {
                checked
                {
                    for (int t = 1; t <= n; t++)
                    {
                        m = m * t;
                    }
                }
            }
            catch (System.OverflowException)     //结果超出范围时的处理
            {
                MessageBox.Show("计算结果超出范围!", "计算错误", MessageBoxButtons.OK,
                MessageBoxIcon.Warning);
                return;
            }
            this.txtResult.Text = m.ToString();
        }
}
```

⑤ 双击“e^”按钮，打开代码编辑器，在其 Click 事件处理程序中输入下列程序代码：

```
/// <summary>
///  求 e 的指数幂
/// </summary>
const double E = 2.72;   //定义常量 E
private void btnE_Click(object sender, EventArgs e)
{
    string param1 = this.txtResult.Text;
    int n;
    int t = 1;
    double m = E;
    try
    {
        n =int.Parse(param1);
    }
    catch (System.FormatException) //指数输入为小数时的错误异常处理
    {
        this.txtResult.Text = "指数必须是整数!";
        return;
    }
    if (n < 0)     //指数为负数时的处理
    {
        try
```

```
            {
                checked
                {
                    for (; t < -n; t++)
                    {
                        m = m * E;
                    }
                }
            }
            catch (System.OverflowException)     //结果超出范围时的处理
            {
                MessageBox.Show("计算结果超出范围!", "计算错误", MessageBoxButtons.OK,
                MessageBoxIcon.Warning);
                return;
            }
            m = 1.0 / m;
            this.txtResult.Text = m.ToString();
        }
        else if (n > 0)
        {
            try
            {
                checked
                {
                    for (; t < n; t++)
                    {
                        m = m * E;
                    }
                }
            }
            catch (System.OverflowException)
            {
                MessageBox.Show("计算结果超出范围!", "计算错误", MessageBoxButtons.OK,
                MessageBoxIcon.Warning);
                return;
            }
            this.txtResult.Text = m.ToString();
        }
        else
        {
            this.txtResult.Text = "1";
        }
    }
```

（4）测试程序。按 F5 键运行程序，通过按钮输入整数数据，按“n!”按钮或“e^”按钮观察结果。

分 析 描 述

（1）通过语句 this.txtResult.Text += this.btn1.Text;可以把用按钮输入的数字串连接起来，实现输入。

（2）语句 this.txtResult.Text = string.Empty;是将文本框清空，相当于 this.txtResult.Text = "";。

（3）在循环语句中 for 后面的表达式可以省略，但分隔符不能省略，如：

```
for (; t < n; t++)
```

上述语句中省略了表达式 int t = 1，这句在循环语句开始前已执行。

（4）由于阶乘和指数都不能为小数，因此将 n 定义为 int 型，如果输入的数据不为 int 类型，将这种可能发生异常情况的程序放到 try 语句中，当出现异常时，由 catch 语句根据格式异常情况作出相应的处理，在文本框中显示提示信息；如果没有异常情况出现，则执行 try…catch 语句后面的代码。如下列程序段：

```
try
{
    n =int.Parse(param1);
}
catch (System.FormatException) //指数输入为小数时的错误异常处理
{
    this.txtResult.Text = "指数必须是整数!";
    return;
}
```

同样的，如果在进行算术运算时可能有溢出情况发生，可以使用 checked 语句进行检查，并将检查可能出现的溢出情况放在 try 语句中，一旦有溢出引发异常时由 catch 语句作出处理，如下列程序段：

```
try
{
    checked
    {
        for (int t = 1; t <= n; t++)
        {
            m = m * t;
        }
    }
}
catch (System.OverflowException)    //结果超出范围时的处理
{
    MessageBox.Show("计算结果超出范围!", "计算错误", MessageBoxButtons.OK,
    MessageBoxIcon.Warning);
    return;
}
```

相 关 知 识

3.2.2 do while 语句

do while 语句格式如下：

```
do
{
    循环体
}
while (表达式)
```

执行过程说明如下：

首先执行循环体中的程序代码，执行完循环体，判断 while 语句中表达式的值，若为 true，

则控制将转到 do 语句的开头，再次执行循环体；如果 while 语句中表达式的值为 false，则结束 do while 语句的执行。do while 语句的流程图如图 3-14 所示。

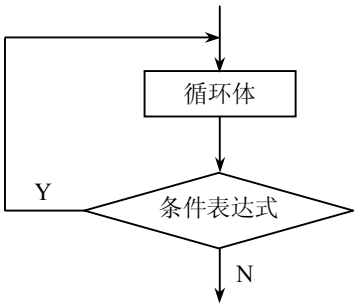

图 3-14　do while 循环结构

3.2.3　for 语句

for 语句格式如下：

```
for (表达式 1；表达式 2；表达式 3)
{
    循环体
}
```

执行过程说明如下：

首先计算表达式 1 的值，接着判断表达式 2 的逻辑值，若为 true，则执行循环体，然后计算表达式 3 的值，再次判断表达式 2 的逻辑值，若逻辑值为 true，则继续执行循环体，若逻辑值为 false，则结束循环语句。for 语句的流程图如图 3-15 所示。

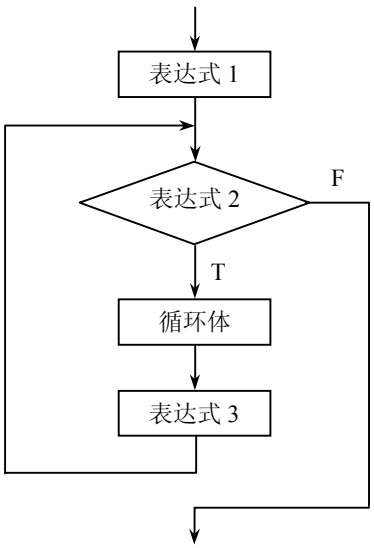

图 3-15　for 循环结构

任务七　使用控制台应用程序计算班级的平均成绩

问题描述

使用控制台应用程序，计算班级的平均成绩。要求可以输入三个班每个班四名学生的学生分数，再计算每个班级的平均分。

任务解决方案

（1）创建名为 Aver 的 Windows 控制台应用程序项目。

（2）编写程序代码。在 Program.cs 中输入下列程序代码：

```
static void Main(string[] args)
{
    int i, j;          //循环变量
    double sum = 0;    //存放总分
    int average;       //存放平均分
    int score;         //存放学生的分数
    for (i = 0; i < 3; i++)   //外层循环控制班级数
    {
        sum = 0;       //总分清零
        Console.WriteLine("\n 请输入第{0}个班的成绩",i+1);
        for (j = 0; j < 4; j++)     //内层循环控制学生数
        {
            Console.Write("第{0}个学生的成绩：",j+1);
            score= int.Parse(Console.ReadLine());   //输入学生成绩
            sum = sum + score;      //累加学生成绩，计算班级总分
        }
        average =(int )Math.Round(sum / 4);        //计算班级平均分
        Console.WriteLine("第{0}个班的平均分为：{1}分",i+1,average);
    }
    Console.ReadLine();
}
```

（3）测试程序。按 F5 键运行程序，并输入数据，结果如图 3-16 所示。

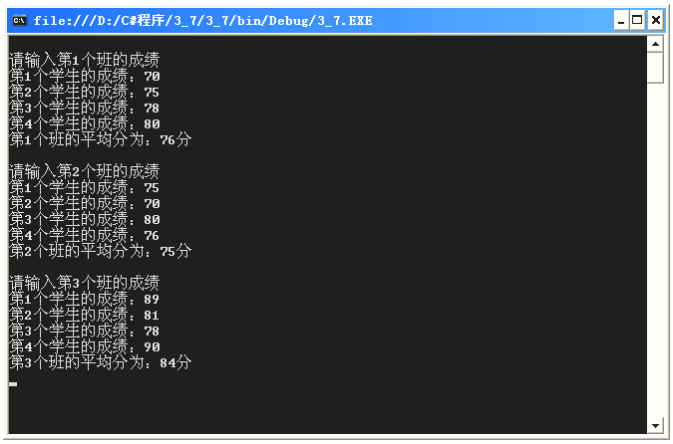

图 3-16　应用控制台输入和输出数据

分 析 描 述

（1）循环嵌套只能一层循环套一层循环，不能交叉。本任务中内循环是用来控制输入某班每个学生的成绩，并通过累加计算该班的总成绩；外循环是用来控制不同的班级，计算班级的平均分数并输出。

（2）将 sum 定义成 double 的目的是用方法 Math.Round(sum/4)计算出来的平均分可以包含小数，再对小数部分四舍五入。

相 关 知 识

3.2.4　循环嵌套

一个循环体内可以包含另一个完整的循环，这种结构为循环嵌套。

while 循环嵌套格式如下：

```
while (表达式 1)
{
    while (表达式 2)
    {
        循环体
    }
}
```

执行过程说明如下：

如果 while 语句中表达式 1 的值为 true，则执行外循环的循环体，接着再判断内循环 while 语句表达式 2 的值，若为 true，则执行内循环的循环体，执行完一遍内循环体，将再次判断内循环表达式 2 的值，如果表达式的值仍然为 true，则再次执行内循环体，直到表达式 2 的值为 false，则结束内循环；控制将转到外循环的 while 语句开头，再次判断外循环表达式 1 的值，若为 true，则再判断内循环 while 语句表达式 2 的值，决定是否执行内循环的循环体，其过程和上面的步骤完全一致，直到外循环表达式 1 的值为 false，结束外循环语句。

对于两层循环嵌套称为双层嵌套，三层以上的循环嵌套，则称为多层嵌套。循环嵌套还包括 for 循环嵌套，do while 循环嵌套，for 循环再套一层 while 循环等。

3.2.5　Math 类

System.Math 类可以用来完成一些常用的数学运算，它提供了一些实现常用数学函数的方法，调用方法时直接用类名.方法名。表 3.9 给出了 Math 类的常用成员。

表 3.9　Math 类的常用成员

方法	描述
Abs()	返回指定数的绝对值
Asin()，ACos()，Atan()	返回反三角函数值
Ceilling()，Floor()	取整函数
Exp()	返回 e 为底的指数幂
Lon()，Lon10()	返回自然对数值，返回以 10 为底的对数值

续表

方法	描述
Max()，Min()	返回两个数中的最大值，两个数中的最小值
Pow()	返回指定数的乘方
Round()	四舍五入
Rint()	返回最近的整数值
Sin()，Cos()，Tan()	返回三角函数值
Sign()	返回指定数的符号
Sqrt()	返回指定数的平方根

例如：

```
int x,y,z,a;
Double b,c;
y=0.025;
x=5;
a=(1+y)math.Pow(x);
b=math.Sqrt(x);
c=math.Exp(x);
console.writeline(a.tostring());
console.writeline(b.tostring());
console.writeline(c.tostring());
```

3.3 跳转语句

C#主要提供了四种不同的跳转语句，分别是 break 语句、goto 语句、return 语句、continue 语句，下面分别介绍这几种语句。

3.3.1 Break 语句

Break 语句可以终止一条多选择语句或迭代语句，使控制流程转到该语句的下一条语句执行。若在多选择 switch 语句中执行 break 语句，则退出 switch 语句，执行该语句的下一句，见任务四中“计算”按钮的事件处理程序；若在循环语句中执行 break 语句，则退出该层循环，执行该循环语句的下一句。

例如：

```
static void Main(string[] args)
{
    int a;
    int s = 1;
    for (int i = 1; i<= 3;i ++ )
    {
        Console.WriteLine("请输入一个正整数：");
        a=int.Parse (Console.ReadLine());
        int j = 1;
        s = 1;
        while (j <= a)
        {
```

```
            s = s * j;
            if (s > 100)
            {
                break ;
             }
             j += 1;
        }
        Console.WriteLine(s );
    }
}
```

按 Ctrl+F5 组合键，程序运行后分别输入 4、5、6，观察结果。当 s 值大于 100 时，执行 break 语句，跳转到控制台输出语句 Console.WriteLine(s); 执行。结果如图 3-17 所示。

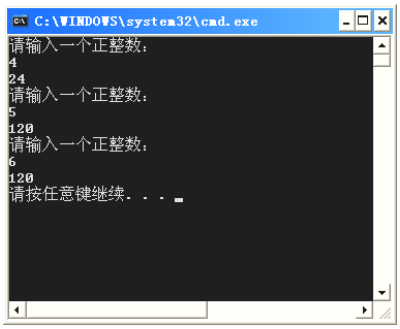

图 3-17　用控制台输出结果

3.3.2　goto 语句

goto 语句可以实现无条件跳转，跳转到标签所指定的代码行执行。该语句可以方便地从某一个地方转到另一个语句去执行，该语句要谨慎使用。

例如：将上面一段程序进行修改，把 break 语句改为 goto 语句。

```
static void Main(string[] args)
{
    int a;
    int s = 1;
    for (int i = 1; i<= 3;i ++ )
    {
        Console.WriteLine("请输入一个正整数：");
        a=int.Parse (Console.ReadLine());
        int j = 1;
        s = 1;
        while (j <= a)
        {
            s = s * j;
            if (s > 100)
            {
                goto end ;
            }
            j += 1;
         }
         Console.WriteLine (s);
```

```
        }
        end: Console.WriteLine ("s 值大于 100 结束" );
    }
```

按 Ctrl+F5 组合键，程序运行后分别输入 4、5，观察结果。当 s 值大于 100 时，执行 goto 语句，跳转到标签为 end 的控制台输出语句 Console.WriteLine ("s 值大于 100 结束");。结果如图 3-18 所示。

图 3-18　用控制台输出结果

3.3.3　continue 语句

continue 语句可以在执行循环体时使程序流程跳过循环体中的剩余语句，继续执行下一轮循环。

例如：

```
static void Main(string[] args)
{
    Console.WriteLine("请输入一个正整数：");
    int a = int.Parse(Console.ReadLine());
    int s = 1;
    for (int j = 1;j <= a;j++)
    {
        s = s * j;
        if (j <a)
        {
           continue;
         }
         Console.WriteLine (s);
    }
}
```

按 Ctrl+F5 组合键，程序运行后输入 6，观察结果。当 j 的值小于 a 的值时，执行 continue 语句，跳转到 for 语句执行 j++，而不执行 Console.WriteLine (s); 语句，直到 j 的值等于 a 的值时，才执行该语句，用控制台输出结果。结果如图 3-19 所示。

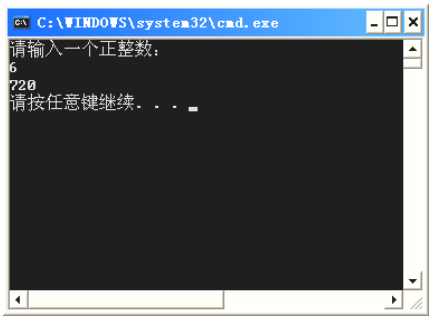

图 3-19　用控制台输出结果

3.3.4　return 语句

return 语句终止方法的执行，并将控制权返回给调用方法。

例如：

```
static void Main(string[] args)
{
    int a;
    int s = 1;
    for (int i = 1; i<= 3;i ++ )
    {
        Console.WriteLine("请输入一个正整数：");
        a=int.Parse (Console.ReadLine());
        int j = 1;
        s = 1;
        while (j <= a)
        {
            s = s * j;
            if (s > 100)
            {
               return;
            }
            j += 1;
        }
        Console.WriteLine (s);
    }
}
```

按 Ctrl+F5 组合键，程序运行后分别输入 4、5，观察结果。当输入 5 时，s 值大于 100，执行 return 语句，退出 Main 方法，将不执行控制台输出语句。结果如图 3-20 所示。

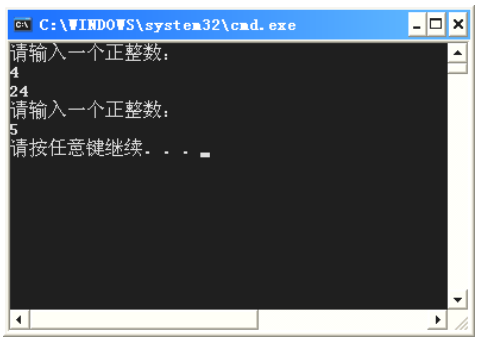

图 3-20　用控制台输出结果

习题三

一、选择题

1．已知 x、y、z 的值分别是 5、10、15，执行下列程序段后，判断 a 中存放的值是多少？（　）

```
if (x+y<z)
a=5;
else if(y<x)
a=y;
else if(z<y)
```

```
a=z;
else
a=x+y+z;
```

A．5　　B．10　　C．15　　D．30

2．下列运算符中具有三元运算符的是（　）。

A．!=　　B．&&　　C．‖　　D．?:

3．循环语句在程序设计中常用的结构是（　）。

A．for 结构、while 结构、if 结构、do…while 结构

B．for 结构、while 结构、switch…case 结构、do…while 结构

C．for 结构、while 结构、do…while 结构、foreach 结构

D．for 结构、while 结构、if 结构、switch…case 结构

4．在循环语句中执行下列哪一语句可以退出当前循环并执行该循环语句的下一语句？（　）

A．continue　　B．break　　C．goto　　D．return

5．try…catch语句中，下列哪种情形是正确的？（　）

A．可能出现异常情况的语句放在 try 中，出现异常情况后的处理程序放在 catch 中

B．可能出现异常情况的语句放在 catch 中，出现异常情况后的处理程序放在 try 中

C．一个 try 可以对多条 catch

D．一个 catch 可以对多条 try

6．下列Math类中的哪个方法可以对x变量中的小数部分四舍五入后取整？（　）

A．Math. Ceilling(x)　　B．Math. Floor(x)

C．Math. Round(x)　　D．Math. Rint(x)

二、应用题

1．输入任意三角形的三条边，计算三角形面积。

要求编写一个程序，输入三角形的三条边，判断能否构成三角形，若不能构成三角形，弹出消息框，要求重新输入；若能构成三角形，则计算三角形的面积并显示输出。界面如图 3-21 和图 3-22 所示。

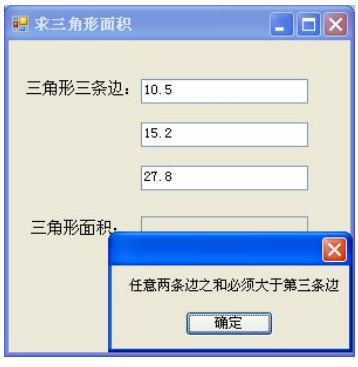

图 3-21　两边之和小于第三条边时弹出消息框

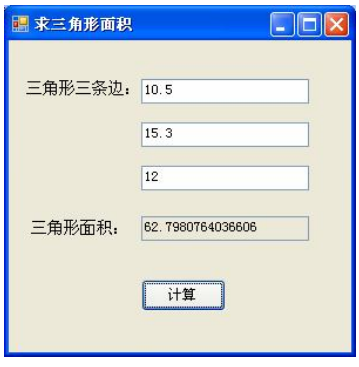

图 3-22　计算三角形面积

三角形面积公式：

$$面积=\sqrt[2]{(P*(P-A)*(P-B)*(P-C))}$$

其中 P = (A + B + C) / 2。

2．输入一个班级学生考试分数，根据考试分数判断等级，若分数≥90，等级为“优”；若 90>分数≥80，等级为“良”；若 80>分数≥70，等级为“中”；若 70>分数≥60，等级为“及格”；分数<60，等级为“不及格”；并统计各等级的人数。

3．购买家用电器，一次购买金额在 10000 元以上，优惠 800 元；购买金额在 8000 元以上，优惠 600 元；购买金额在 5000 元以上，优惠 300 元；若有金卡，还可以在购买总额上优惠 1%；若有银卡，还可以在购买总额上优惠 0.5%；计算实际优惠金额和实付金额。界面如图 3-23 所示。

图 3-23　计算优惠金额和实付金额界面

4．设计一个计算器，能够进行加、减、乘、除、求阶乘等运算，界面如图 3-24 所示。

图 3-24　计算器