第 3 章

C# 控制语句

一个 C＃程序是由若干语句组成的，每个语句以分号作为结束符。C＃语句可以很简单，也可以很复杂，其中改变程序正常流程的语句称为控制语句。归纳起来，程序的控制语句有 3 种，即顺序控制语句、选择控制语句和循环控制语句。其中，顺序控制语句是指所有语句按顺序执行，每一条语句只执行一遍，不重复执行，也没有语句不执行，由于它十分简单，这里不做介绍，本章主要讨论选择控制语句和循环控制语句。

本章学习要点：

☑ 掌握 C＃中各种 if 语句和 switch 语句的使用方法。

☑ 掌握 C＃中 while、do…while 和 for 循环语句的使用方法。

☑ 掌握 C＃中 break、continue 语句的使用方法。

☑ 使用 C＃中的各种控制语句设计较复杂的程序。

3.1 选择控制语句

C＃中的选择控制语句有 if 语句、if…else 语句、if…else if 语句和 switch 语句，它们根据指定条件的真假值确定执行哪些简单语句，其中，简单语句既可以是单个语句，也可以是用{}括起来的复合语句。

3.1.1 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 3.1 所示。

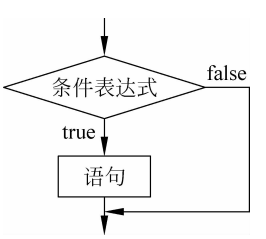

图 3.1 if 语句的执行流程

【例 3.1】 编写一个程序，用 if 语句显示用户所输入数值的绝对值。

解： 在“D:\C＃程序\ch3”文件夹中创建控制台应用程序项目 proj3-1，其代码如下。

```
using System;
namespace proj3_1
{   class Program
    {   static void Main(string[] args)
        {   int x;
```

```
            x = int.Parse(Console.ReadLine());
            if (x < 0) x = -x;
            Console.WriteLine("绝对值为{0}",x);
        }
    }
}
```

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

3.1.2 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，用 else 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 3.2 所示。

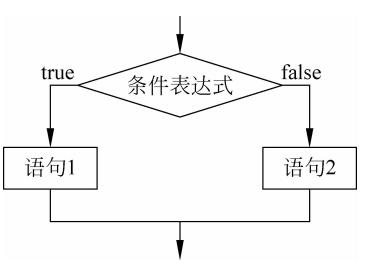

图 3.2 if…else 语句的执行流程

if…else 语句可以嵌套使用。但当多个 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，在 C# 中规定，else 总是和最后一个出现的还没有 else 与之匹配的 if 匹配。

【例 3.2】 用 if…else 语句编写一个程序，显示用户所输入数值的绝对值。

解：在“D:\C# 程序\ch3”文件夹中创建控制台应用程序项目 proj3-2，其代码如下。

```
using System;
namespace proj3_2
{   class Program
    {   static void Main(string[] args)
        {   int x;
            x = int.Parse(Console.ReadLine());
            if (x < 0) Console.WriteLine("绝对值为{0}", -x);
            else Console.WriteLine("绝对值为{0}", x);
        }
    }
}
```

在所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;
```

【例 3.3】　编写一个程序，判断输入的年份是否为闰年。

解：能被 400 整除的，或不能被 100 整除但能被 4 整除的年份为闰年。在“D:\C＃程序\ch3”文件夹中创建控制台应用程序项目 proj3-3，其代码如下。

```
using System;
namespace proj3_3
{    class Program
     {    static void Main(string[] args)
          {    int year, rem4, rem100, rem400;
               Console.Write("输入年份:");
               year = int.Parse(Console.ReadLine());
               rem400 = year % 400;
               rem100 = year % 100;
               rem4 = year % 4;
               if ((rem400 == 0) || ((rem4 == 0) && (rem100 != 0)))
                    Console.WriteLine("{0}是闰年",year);
               else
                    Console.WriteLine("{0}不是闰年", year);
          }
     }
}
```

图 3.3　例 3.3 程序的执行结果

本程序的一次执行结果如图 3.3 所示，表示 2012 年是闰年。

3.1.3　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;
```

该语句的功能是先计算“条件表达式 1”的值，如果为 true，则执行“语句 1”，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，依此类推。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。

【例 3.4】　编写一个程序，将用户输入的分数转换成等级 A(≥90)、B(80～89)、C(70～79)、D(60～69)、E(＜60)。

解：在“D:\C＃程序\ch3”文件夹中创建控制台应用程序项目 proj3-4，其代码如下。

```
using System;
namespace proj3_4
{    class Program
     {    static void Main(string[] args)
          {    float x;
               Console.Write("分数:");
               x = float.Parse(Console.ReadLine());
```

```
            if (x >= 90) Console.WriteLine("等级为 A");
            else if (x >= 80) Console.WriteLine("等级为 B");
            else if (x >= 70) Console.WriteLine("等级为 C");
            else if (x >= 60) Console.WriteLine("等级为 D");
            else Console.WriteLine("等级为 E");
        }
    }
}
```

本程序的一次执行结果如图 3.5 所示,表示 82 分的等级为 B。

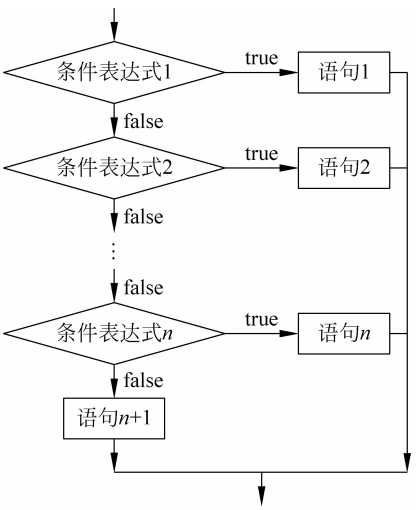

图 3.4 if…else if 语句的执行流程

图 3.5 例 3.4 程序的执行结果

3.1.4 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 (表达式)
{   case 常量表达式 1:语句 1;
    case 常量表达式 2:语句 2;
     ⋮
    case 常量表达式 n:语句 n;
    default:语句 n+1;
}
```

switch 语句将控制传递给与“表达式”值匹配的 case 块。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。语句体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 块显式地贯穿到另一个 case 块。但有一个例外,当 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。

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

switch 语句的执行流程如图 3.6 所示。

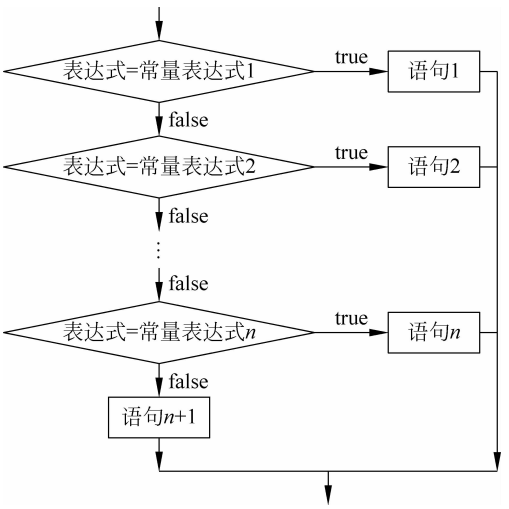

图 3.6　带 break 语句的 switch 控制流程

注意：在 C/C++ 中 switch 执行完一个 case 语句后，可以继续执行下一个 case 语句，而 C# 中的 switch 不能这样。

【例 3.5】 编写一个程序，要求输入课程后显示相应的学分：数学（代号为 m，8 学分）、物理（代号为 p，5 学分）、化学（代号为 c，5 学分）、语文（代号为 w，8 学分）、英语（代号为 e，6 学分）。

解：在“D:\C# 程序\ch3”文件夹中创建控制台应用程序项目 proj3-5，其代码如下。

```
using System;
namespace proj3_5
{    class Program
     {    static void Main(string[] args)
          {    char ch;
               Console.Write("课程代号:");
               ch = (char)Console.Read();
               switch (ch)
               {
               case 'm':case 'M':case 'w':case 'W':
                    Console.WriteLine("8 学分");
                    break;
               case 'p':case 'P':case 'c':case 'C':
                    Console.WriteLine("5 学分");
                    break;
               case 'e':case 'E':
                    Console.WriteLine("6 学分");
                    break;
               default:
                    Console.WriteLine("输入的课程代号不正确");
                    break;
               }
          }
     }
}
```

本程序的一次执行结果如图 3.7 所示，表示代号为 e 的课程的学分为 6。

图 3.7　例 3.5 程序的执行结果

3.2　循环控制语句

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,则结束循环,否则继续循环。C# 支持 3 种格式的循环控制语句,即 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。

3.2.1　while 语句

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",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。

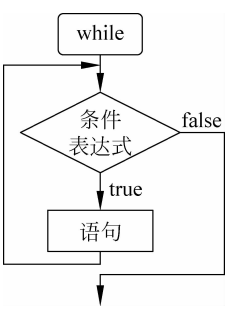

图 3.8　while 语句的执行流程

【例 3.6】　编写一个程序,将用户输入的整数反向显示出来。

解:对于用户输入的正整数 num,采用辗转相除法(while 语句实现)求出所有位对应的数字并输出。在"D:\C# 程序\ch3"文件夹中创建控制台应用程序项目 proj3-6,其代码如下。

```
using System;
namespace proj3_6
{    class Program
     {    static void Main(string[] args)
          {    int digit,num;
               Console.Write( "输入一个整数:");
               num = int.Parse(Console.ReadLine());
               Console.Write( "反向显示结果:");
               while (num!= 0)
               {    digit = num % 10;          //依次求个位、十位、…上的数字 digit
                    num = num / 10;
                    Console.Write(digit);
               }
               Console.WriteLine();
          }
     }
}
```

本程序的一次执行结果如图 3.9 所示。

图 3.9　例 3.6 程序的执行结果

3.2.2　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);
```

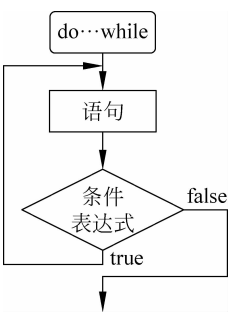

图 3.10　do…while 语句的执行流程

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 语句如果条件第一次就不满足，语句一次也不会执行。其执行流程如图 3.10 所示。

【例 3.7】 采用 do…while 语句重新编写例 3.6 的程序。

解：采用 do…while 语句实现辗转相除法求各位上的数字。在“D:\C＃程序\ch3”文件夹中创建控制台应用程序项目 proj3-7，其代码如下。

```
using System;
namespace proj3_7
{   class Program
    {   static void Main(string[] args)
        {   int digit,num;
            Console.Write("输入一个整数:");
            num = int.Parse(Console.ReadLine());
            Console.Write("反向显示结果:");
            do
            {   digit = num % 10;
                num = num/10;
                Console.Write(digit);
            } while (num!= 0);
            Console.WriteLine();
        }
    }
}
```

3.2.3　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;
```

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。

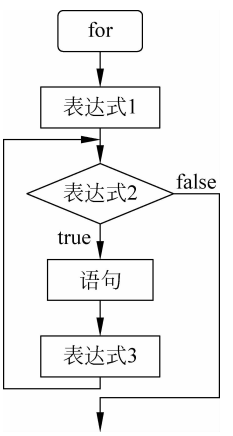

图 3.11 for 语句的执行流程

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,这将在后面介绍。

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 3.8】 编写一个程序,输出如图 3.12 所示的九九乘法表。

解:在“D:\C# 程序\ch3”文件夹中创建控制台应用程序项目 proj3-8,其代码如下。

```
using System;
namespace proj3_8
{   class Program
    {   static void Main(string[] args)
        {   int i,j;
            for (i = 1; i <= 9; i++)
            {   for (j = 1; j <= i; j++)
                    Console.Write("{0}×{1}={2} ", i, j, i * j);
                Console.WriteLine();
            }
        }
    }
}
```

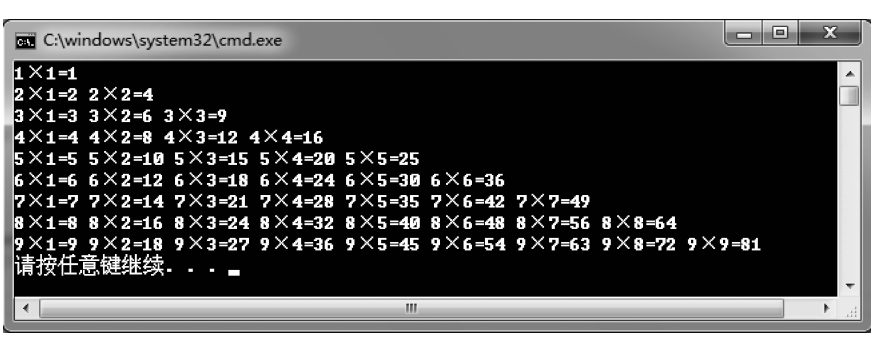

图 3.12 九九乘法表

3.3　跳 转 语 句

除了顺序执行和选择、循环控制外，有时需要中断一段程序的执行，跳转到其他地方继续执行，这时需要用到跳转语句。跳转语句包括 break、continue 和 goto 语句。

3.3.1　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。

【例 3.9】 编写一个程序，判断从键盘输入的大于 3 的正整数是否为素数。

解：采用 for 循环语句，当 n 能被 $3\sim\sqrt{n}$ 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。在“D:\C＃程序\ch3”文件夹中创建控制台应用程序项目 proj3-9，其代码如下。

```
using System;
namespace proj3_9
{   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 = 3; i <= Math.Sqrt(n); i++)
                if (n % i == 0)
                {   prime = false;
                    break;
                }
            if (prime) Console.WriteLine("{0}是素数", n);
            else Console.WriteLine("{0}不是素数", n);
        }
    }
}
```

在前面介绍的 switch 语句中也用到了 break 语句，它表示终止当前 switch 语句的执行，接着运行 switch 语句后面的语句。

3.3.2　continue 语句

continue 语句也用于循环语句，它类似于 break，但它不是结束循环，而是结束循环语句的当前循环，接着执行下一次循环。在 while 和 do…while 循环结构中，执行控制权转至对“条件表达式”的判断，在 for 结构中，转去执行“表达式 2”。

【例 3.10】 编写一个程序，对用户输入的所有正数求和，如果输入的是负数，则忽略该数。程序每读入一个数，判断它的正负，如果为负，则利用 continue 语句结束当前循环，继续下一次循环，否则将该数加到总数上。

解：使用 while 循环语句，当输入的数 n 为 0 时退出循环，当 n 小于 0 时使用 continue 语句重新开始下一轮循环，当大于 0 时将其累计加到 sum 中。在“D:\C＃程序\ch3”文件夹中创建控制台应用程序项目 proj3-10，其代码如下。

```
using System;
namespace proj3_10
{   class Program
    {   static void Main(string[] args)
        {   int sum = 0,n = 1;
            while (n!= 0)                    //循环
            {   Console.Write("输入一个整数(以 0 表示结束):");
                n = int.Parse(Console.ReadLine());
                if (n < 0) continue;         //开始下一次循环
                sum += n;
            }
            Console.WriteLine("所有正数之和 = {0}",sum);
        }
    }
}
```

3.3.3 goto 语句

使用 goto 语句也可以跳出循环和 switch 语句。goto 语句用于无条件转移程序的执行控制,它总是和一个标号相匹配,其形式如下:

```
goto 标号;
```

“标号”是一个用户自定义的标识符,它可以处于 goto 语句的前面,也可以处于其后面,但是标号必须和 goto 语句处于同一个函数中。在定义标号时,由一个标识符后面跟一个冒号组成。

【例 3.11】 编写一个程序,求满足条件 $1^2+2^2+\cdots+n^2\leqslant 1000$ 的最大的 n。

解:在“D:\C#程序\ch3”文件夹中创建控制台应用程序项目 proj3-11,其代码如下。

```
using System;
namespace proj3_11
{   class Program
    {   static void Main(string[] args)
        {   int sum = 0,n = 0;
            while (true)
            {   sum += n * n;
                if (sum > 1000) goto end;
                n++;
            }
            end:Console.WriteLine( "最大的 n 为:{0}",n - 1);
        }
    }
}
```

本程序的执行结果如图 3.13 所示,表示求出的最大的 n 为 13。

图 3.13 例 3.11 程序的执行结果

注意：由于 goto 语句会严重破坏程序的结构，完全可以将使用 goto 语句的程序修改为更加合理的结构，所以一般不推荐使用该语句。

【例 3.12】 不使用 goto 语句重新编写例 3.11 的程序。

解：在“D:\C#程序\ch3”文件夹中创建控制台应用程序项目 proj3-12，其代码如下。

```
using System;
namespace proj3_12
{   class Program
    {   static void Main(string[] args)
        {   int sum = 0,n = 0;
            do
            {   sum += n * n;
                if (sum > 1000) break;
                n++;
            } while (sum < 1000);
            Console.WriteLine("最大的 n 为:{0}",n - 1);
        }
    }
}
```

练 习 题 3

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) 有以下 C#程序：

```
using System;
namespace aaa
{   class Program
    {   static void Main()
        {   int x = 2, y = -1, z = 2;
            if (x < y)
                if (y < 0) z = 0;
                else z += 1;
            Console.WriteLine("{0}",z);
        }
    }
}
```

该程序的输出结果是_____。

A. 3　　B. 2　　C. 1　　D. 0

(3) 有以下 C#程序，在执行时从键盘输入 9，则输出结果是_____。

```
using System;
namespace aaa
{   class Program
    {   static void Main()
        {   int n;
            n = int.Parse(Console.ReadLine());
            if (n++< 10)
```

```
                Console.WriteLine("{0}", n);
            else
                Console.WriteLine("{0}",n--);
        }
    }
}
```

A. 11　　B. 10　　C. 9　　D. 8

(4) 有以下 C# 程序:

```
using System;
namespace aaa
{   class Example1
    {   static void Main(string[] args)
        {   int x = 1,a = 0,b = 0;
            switch(x)
            {   case 0:b++;break;
                case 1:a++;break;
                case 2:a++;b++;break;
            }
            Console.WriteLine("a = {0},b = {1}",a,b);
        }
    }
}
```

该程序的输出结果是_____。

A. a=2,b=1　　B. a=1,b=1　　C. a=1,b=0　　D. a=2,b=2

(5) 有以下 C# 程序:

```
using System;
namespace aaa
{   class Program
    {   static void Main()
        {   int a = 15, b = 21, m = 0;
            switch (a % 3)
            {
            case 0: m++; break;
            case 1: m++;
                    switch (b % 2)
                    {
                    case 0: m++; break;
                    default: m++; break;
                }
                break;
            }
            Console.WriteLine("{0}",m);
        }
    }
}
```

该程序的输出结果是_____。

A. 1　　B. 2　　C. 3　　D. 4

(6) 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

(7) 以下关于 for 循环的说法不正确的是_____。

A. for 循环只能用于循环次数已经确定的情况

B. for 循环是先判定表达式，后执行循环体语句

C. 在 for 循环中可以用 break 语句跳出循环体

D. 在 for 循环体语句中可以包含多条语句，但要用花括号括起来

(8) 有以下 C# 程序：

```
using System;
namespace aaa
{   class Program
    {   static void Main()
        {   int i,j,s = 0;
            for(i = 2;i < 6;i++,i++)
            {   s = 1;
                for(j = i;j < 6;j++)
                    s += j;
            }
            Console.WriteLine("{0}",s);
        }
    }
}
```

该程序的输出结果是_____。

A. 9　　B. 1　　C. 11　　D. 10

(9) 有以下 C# 程序：

```
using System;
namespace aaa
{   class Program
    {   static void Main()
        {   int i = 0, s = 0;
            do
            {   if(i % 2 == 1)
                {   i++;
                    continue;
                }
                i++;
                s += i;
            } while(i < 7);
            Console.WriteLine("{0}",s);
        }
    }
}
```

该程序的输出结果是_____。

A. 16　　B. 12　　C. 28　　D. 21

(10) 有以下 C# 程序：

```
using System;
namespace aaa
{   class Program
    {   static void Main()
        {   int i = 0, a = 0;
            while(i < 20)
            {   for (; ; )
                {   if (i % 10 == 0) break;
                    else i--;
                }
                i += 11;
                a += i;
            }
            Console.WriteLine("{0}", a);
        }
    }
}
```

该程序的输出结果是_____。

A. 21　　B. 32　　C. 33　　D. 11

2. 问答题

(1) 简述 C# 中的 3 种控制结构语句。

(2) 简述 C# 中 do…while 和 while 两种循环语句的不同之处。

(3) 简述 C# 中 continue 语句的作用。

(4) 简述 C# 中 break 语句的作用。

3. 编程题

(1) 设计控制台应用程序项目 exci3-1，读入一组整数(以输入 0 结束)，分别输出其中奇数和偶数的和。

(2) 设计控制台应用程序项目 exci3-2，输入正整数 n，计算 $s=1+(1+2)+(1+2+3)+\cdots+(1+2+3+\cdots+n)$。

(3) 设计控制台应用程序项目 exci3-3，输出如图 3.14 所示的 n 阶杨辉三角形(这里 $n=10$)，其中 n 由用户输入，该值不能大于 13。

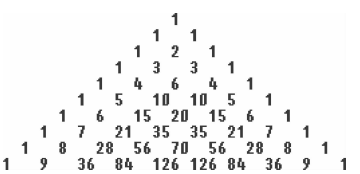

图 3.14　$n=10$ 时的杨辉三角形

(4) 设计控制台应用程序项目 exci3-4，利用下列公式编程计算 π 的值。

$$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots+\frac{1}{4n-3}-\frac{1}{4n-1}\quad(n=2000)$$

4. 上机实验题

编写控制台应用程序项目 experment3，输出所有这样的三位数：这个三位数本身恰好等于其每个数字的立方和(例如 $153=1^3+5^3+3^3$)。