实验 3　程序流程控制(2)

实 验 目 的

- 掌握 for 循环语句的使用；
- 掌握 while 循环语句的使用；
- 掌握 do…while 循环语句的使用；
- 掌握多重循环结构程序流程；
- 了解跳转语句的使用；
- 了解程序异常处理机制。

实 验 内 容

实验 3-1　求 $n!$

实验要求：输入整数 $n(n\geqslant 0)$，分别利用 for 循环、while 循环、do…while 循环求 $n!$。运行效果如图 3-1 所示。

图 3-1　实验 3-1 运行效果

操作提示：

(1) $n!=n\times(n-1)\times(n-2)\times\cdots\times 2\times 1$。例如，$5!=5\times4\times3\times2\times1=120$。特别地，$0!=1$。

(2) 一般地，累乘的初值为 1，而累加的初值为 0。

(3) 如果输入的是负整数，则继续提示输入非负整数，直至 $n\geqslant 0$。

(4) 程序代码如图 3-2 所示。

实验 3-2　显示 Fibonacci 数列

实验要求：显示 Fibonacci 数列：1,1,2,3,5,8,…。当 Fibonacci 的值大于 10000 时停止显示。要求每行显示 5 项，运行效果如图 3-3 所示。

```
static void Main(string[] args)
{
    int i, n, fac = 1;
    string s;
    n = -1;
    while (n < 0)
    {
        Console.Write("请输入非负整数n: ");
        s = Console.ReadLine();
        n = int.Parse(s);
    }
    //方法一: for循环
    for (i = 1; i <= n; i++) fac *= i;
    Console.WriteLine("       for循环: {0}! = {1}", n, fac);
    //方法二: while循环
    i = 1; fac = 1;
    while (i <= n) fac *= i++;
    Console.WriteLine("     while循环: {0}! = {1}", n, fac);
    //方法三: do...while循环
    i = 1; fac = 1;
    do
    {
        fac *= i; i++;
    } while (i <= n);
    Console.WriteLine("do...while循环: {0}! = {1}", n, fac);
    Console.ReadKey();
}
```

图 3-2　实验 3-1 程序代码

```
  1   1   2   3   5
  8  13  21  34  55
 89 144 233 377 610
987 1597 2584 4181 6765
```

图 3-3　实验 3-2 运行效果

操作提示：Fibonacci 数列的生成规律为：

$$\begin{cases} F_1 = 1 & n = 1 \\ F_2 = 1 & n = 2 \\ F_n = F_{n-1} + F_{n-2} & n \geqslant 3 \end{cases}$$

程序代码如图 3-4 所示。

```
static void Main(string[] args)
{
    int f1 = 1, f2 = 1, f3, num = 2;
    Console.Write("{0,5}\t{0,5}\t", f1, f2);
    f3 = f1 + f2;
    while (f3 <= 10000)
    {
        Console.Write("{0,5}\t", f3);
        num++;
        if (num % 5 == 0) Console.WriteLine();
        f1 = f2;
        f2 = f3;
        f3 = f1 + f2;
    }
    Console.ReadKey();
}
```

图 3-4　实验 3-2 程序代码

实验 3-3　鸡兔同笼问题

实验要求：已知在同一个笼子里总共有 h 只鸡和兔，鸡和兔的总脚数为 f 只，其中 h 和 f 由用户输入，求鸡和兔各有多少只？要求使用两种方法：一是求解方程；二是利用循环进行枚举测试。运行效果如图 3-5 所示。

(a) 合理解　　(b) 无解

图 3-5　实验 3-3 运行效果

操作提示：

(1) 已知鸡和兔的总数为 h，它们的总脚数为 f。假设鸡有 c 只，兔有 r 只。

(2) 方法一：求解方程法。由公式：

$$\begin{cases} c + r = h \\ 2c + 4r = f \end{cases}$$

解得：

$$\begin{cases} r = \dfrac{f}{2} - h \\ c = h - r \end{cases}$$

由公式推得，鸡和兔的总脚数 f 必须是偶数，并且鸡和兔的只数必须是非负整数。

(3) 方法二：利用循环进行枚举测试。鸡的只数 c 取值范围为 0～h，兔的只数为 r，如果满足条件，则求得解。

(4) 程序代码如图 3-6 所示。

```
static void Main(string[] args)
{
    int c, r;                       //number of chicken & rabbit
    Console.Write("请输入总头数：  ");
    String s = Console.ReadLine();
    int h = int.Parse(s);           //total heads of chicken & rabbit
    int f = 1;
    while (f % 2 != 0)
    {
        Console.Write("请输入总脚数(必须是偶数)：");
        s = Console.ReadLine();
        f = int.Parse(s);           //total feet of chicken & rabbit
    }
    //方法一：利用循环
    bool solution = false;          //判断是否有解
    for (c = 0; c <= h; c++)
    {
        r = h - c;
        if (2 * c + 4 * r == f)
        {
            Console.WriteLine("方法一：鸡：{0} 只， 兔：{1} 只", c, r);
            solution = true;
        }
    }
    if (!solution) Console.WriteLine("方法一：无解，请重新运行测试！");
    //方法二：解方程
    r = f / 2 - h;
    c = h - r;
    solution = false;               //判断是否有解
    if (r >= 0 && c >= 0)
    {
        Console.WriteLine("方法二：鸡：{0} 只， 兔：{1} 只", c, r);
        solution = true;
    }
    if (!solution) Console.WriteLine("方法二：无解，请重新运行测试！");
    Console.ReadKey();
}
```

图 3-6　实验 3-3 程序代码

实验 3-4　利用级数和求 π

实验要求：使用格利高利公式求 π 的近似值，直到最后一项的绝对值小于 10^{-6} 为止。运行效果如图 3-7 所示。

$$\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

操作提示：程序代码如图 3-8 所示。

图 3-7　实验 3-4 运行效果

```
static void Main(string[] args)
{
    float n, t, pi;
    int s;
    pi = 0; t = 1; n = 1; s = 1;
    while (Math.Abs(t) >= Math.Pow(10, -6))
    {
        pi += t;
        n += 2;
        s = -s;
        t = s / n;
    }
    pi *= 4;
    Console.WriteLine("pi = {0}", pi);
    Console.ReadKey();
}
```

图 3-8　实验 3-4 程序代码

实验 3-5　求最大公约数和最小公倍数

实验要求：产生两个 1～100(包含 1 和 100)的随机数 a 和 b，求这两个整数的最大公约数和最小公倍数。运行效果如图 3-9 所示。

操作提示：程序代码如图 3-10 所示。

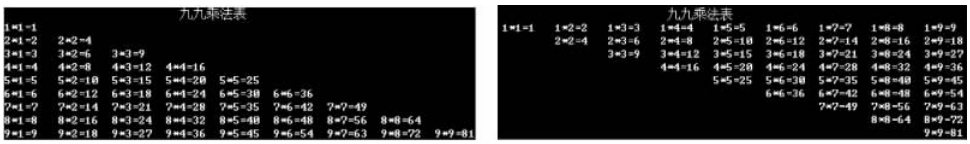
图 3-9　实验 3-5 运行效果

```
static void Main(string[] args)
{
    int m, n, r, m1, n1;
    Random rNum = new Random();
    m1 = rNum.Next(1,101);  //产生1~100之间的随机数a
    n1 = rNum.Next(1,101);  //产生1~100之间的随机数b
    Console.WriteLine("整数1 = {0}, 整数2 = {1}", m1, n1);
    if (m1 > n1)
    {
        m = m1; n = n1;
    }
    else
    {
        m = n1; n = m1;
    }
    do
    {
        r = m % n;
        m = n;
        n = r;
    } while (r != 0);
    Console.WriteLine("最大公约数 = {0}, 最小公倍数 = {1}", m, m1 * n1 / m);
    Console.ReadKey();
}
```

图 3-10　实验 3-5 程序代码

实验 3-6　打印九九乘法表

实验要求：利用嵌套循环打印运行效果如图 3-11 所示的呈下三角和呈上三角的九九乘法表。



(a) 呈下三角的九九乘法表　　(b) 呈上三角的九九乘法表

图 3-11　实验 3-6 运行效果

操作提示：程序代码如图 3-12 所示。

```
static void Main(string[] args)
{
    String s;
    //下三角
    Console.WriteLine("                    九九乘法表");
    for (int i = 1; i <= 9; i++)
    {
        s = "";
        for (int j = 1; j <= i; j++)
        {
            s += (String.Format("{0}*{1}={2}", i, j, i * j)).PadRight(8);
        }
        Console.WriteLine(s);
    }
    //上三角
    Console.WriteLine();
    Console.WriteLine("                    九九乘法表");
    for (int i = 1; i <= 9; i++)
    {
        s = "";
        s += s.PadRight(8 * (i - 1) + 1);
        for (int j = i; j <= 9; j++)
        {
            s += (String.Format("{0}*{1}={2}", i, j, i * j)).PadRight(8);
        }
        Console.WriteLine(s);
    }
    Console.ReadKey();
}
```

图 3-12　实验 3-6 程序代码

实验 3-7　素数的判断

实验要求：分别利用 for 循环和 while 循环显示 1～100 的所有素数，要求每行显示 10 项。运行效果如图 3-13 所示。

```
方法一：1~100间所有的素数为：
   2    3    5    7   11   13   17   19   23   29
  31   37   41   43   47   53   59   61   67   71
  73   79   83   89   97
方法二：1~100间所有的素数为：
   2    3    5    7   11   13   17   19   23   29
  31   37   41   43   47   53   59   61   67   71
  73   79   83   89   97
```

图 3-13　实验 3-7 运行效果

操作提示：

(1) 所谓素数(或称为质数)是指除了1和该数本身，不能被任何整数整除的正整数。判断一个正整数 m 是否为素数，只要判断 m 可否被 $2\sim\sqrt{m}$ 中的任何一个整数整除，如果 m 不能被此范围中任何一个整数整除，m 即为素数，否则 m 为合数。

(2) 程序代码如图3-14所示。

```
static void Main(string[] args)
{
    int m, k, i, num = 0;
    //方法一：利用for循环和break语句
    Console.WriteLine("方法一：1~100间所有的素数为：");
    for (m = 2; m <= 100; m++)
    {
        k = (int)(Math.Sqrt(m));
        for (i = 2; i <= k; i++)
            if (m % i == 0) break;
        if (i == k + 1)
        {
            Console.Write("{0,5}", m);
            num++;
            if (num % 10 == 0) Console.WriteLine();
        }
    }
    //方法二：利用while循环和boolean变量
    Console.WriteLine("\n方法二：1~100间所有的素数为：");
    num = 0;
    for (m = 2; m <= 100; m++)
    {
        bool flag = true;           //假设整数为素数
        k = (int)(Math.Sqrt(m));
        i = 2;
        while (i <= k && flag == true)
        {
            if (m % i == 0) flag = false; //可以整除，肯定不是素数
            else i++;
        }
        if (flag == true)
        {
            Console.Write("{0,5}", m);
            num++;
            if (num % 10 == 0) Console.WriteLine();
        }
    }
    Console.ReadKey();
}
```

图3-14　实验3-7程序代码

实验3-8　异常处理

实验要求：输入任意两个整数，求两者的商。使用异常处理机制捕捉零除异常和参数格式异常。运行效果如图3-15所示。

(a) 输入格式异常　(b) 零除异常　(c) 正确的商

图3-15　实验3-8运行效果

操作提示：

(1) 当试图用0除整数值或十进制数值时将引发DivideByZeroException异常。

(2) 当方法调用中参数的格式不符合对应的形参类型的格式时将引发FormatException异常。

（3）程序代码如图 3-16 所示。

```
static void Main(string[] args)
{
    int i, j, k;
    Console.Write("请输入被除数：");
    try
    {
        String s = Console.ReadLine();
        i = int.Parse(s);        //被除数
        Console.Write("请输入除数：");
        s = Console.ReadLine();
        j = int.Parse(s);        //除数
        k = i / j;
        Console.WriteLine("被除数/除数 = {0}", k);
    }
    catch (FormatException e1)
    {
        Console.WriteLine("输入格式不正确！");
    }
    catch (DivideByZeroException e2)
    {
        Console.WriteLine("除数不能为0！");
    }
    Console.ReadKey();
}
```

图 3-16　实验 3-8 程序代码