程序设计例

一、分支程序设计例：

【例 1】　编程实现分段函数

$$y=\begin{cases} x+1, & x<0 \\ 1, & 0\le x<1 \\ x^3 & 1\le x \end{cases}$$

程序代码如下：

```
01：  using System;
02：
03：  namespace CSHARP4_1
04：  {
05：      class Program
06：      {
07：          static void Main(string[] args)
08：          {
09：              double x, y;
10：              Console.WriteLine("请输入 x 的值 ：");
11：              x = Convert.ToDouble(Console.ReadLine());
12：              if (x < 0)
13：              {
14：                  y = x + 1;
15：                  Console.WriteLine("x={0},    y=x+1={1}", x, y);
16：              }
17：              else if (x < 1)              // 0 ≤x < 1
18：              {
19：                  y = 1;
20：                  Console.WriteLine("x={0},    y={1}", x, y);
21：              }
22：              else                                  // 1 ≤x
23：              {
24：                  y = x * x * x;
25：                  Console.WriteLine("x={0},    y=x*x*x={1}", x, y);
26：              }
27：          }
28：      }
29：  }
```

运行结果：

```
请输入 x 的值 ：
15
x=15,    y=x*x*x=3375
```

二、循环程序设计例

【例 1】　计算 $e=1+\frac{1}{1!}+\frac{1}{2!}+...+\frac{1}{n!}+...$，当通项 $\frac{1}{n!}<10^{-7}$ 时停止计算。

定义三个工作变量 e 、n 和 u，分别用于存放已计算出的结果近似值、当前项序号和当前通项值，则伪代码算法为:

```
e = 1.0;   n = 1;   u = 1.0;
while (通项 u 大于等于 10^-7)
{
    计算新的通项值  u = u/n;
    将新通项值加到结果近似值上;
        准备处理下一项  n = n+1;
    }
```

程序代码如下：

```
01：  using System;
02：
03：  namespace CSHARP4_3
04：  {
05：      class Program
06：      {
07：          static void Main(string[] args)
08：          {
09：              double e = 1.0;
10：              double u = 1.0;
11：              int n = 1;
12：              while (u >= 1.0E-7)
13：              {
14：                  u = u / n;
15：                  e = e + u;
16：                  n = n + 1;
17：              }
18：              Console.WriteLine("e = {0} ( n = {1} )", e, n);
19：          }
20：      }
21：  }
```

程序运行结果：

```
e = 2.71828182619849 ( n = 12 )
```

说明：根据计算结果同时打印出的项数 n，表明该级数收敛相当快，仅计算到前 12 项其截断误差便已小于 10^{-7}。

【例 2】 编写程序制作九九乘法表。

该题目需要 2 个变量间的多次计算，故需要两重循环，即循环语句中的循环体又包含另

一个循环语句。这种结构称为嵌套循环。循环的嵌套层次从语法上没有限制，但一般不超过3层，否则将影响可读性。

程序代码如下：

```
01：  using System;
02：
03：  namespace CSHARP4_6
04：  {
05：      class Program
06：      {
07：          static void Main(string[] args)
08：          {
09：              int i, j;
10：              for (i = 1; i < 10; i++)
11：              {
12：                  for (j = 1; j <= i; j++)
13：                      Console.Write("{0}*{1}={2}\t", j, i, i * j);
14：                  Console.WriteLine();
15：              }
16：          }
17：      }
18：  }
```

运行结果：

```
1*1=1
1*2=2   2*2=4
1*3=3   2*3=6   3*3=9
1*4=4   2*4=8   3*4=12  4*4=16
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

计算机首先按行输出，从第1行第第9行，输出数据中第3列正好是1～9。每行输出后中的数据是由第2层循环实现的，其中第1个数就是列数，比如在第5行，请寻找数字1、2、3、4、5，正好是列数。

三、综合设计

【例1】 找出2～10000之内的所有完全数。所谓完全数是该数各个真因子之和等于它本身的自然数，又称完美数或完备数（Perfect number）。例如：第一个完全数是6，它有约数1、2、3、6，除去它本身6外，其余3个数相加，1+2+3=6。第二个完全数是28，它有约数1、2、4、7、14、28，除去它本身28外，其余5个数相加，1+2+4+7+14=28。后面的

完全数还有 496、8128 等等。

程序首先计算整数的真因子之和。整数 1 肯定是因子，不需要判断，直接赋给因子之和 sum。计算整数 i 的真因子之和算法如下：

```
sum=1;
for(j=2;j<=i/2;j++)
{
    如果 j 是 i 的因子，将 j 加入因子之和 sum 中
}
```

然后根据整数 i 是否等于真因子之和 sum 来判断 i 是否是完全数。

程序代码如下：

```
01：  using System;
02：
03：  namespace CSHARP4_10
04：  {
05：      class Program
06：      {
07：          static void Main(string[] args)
08：          {
09：              int i, j, sum;
10：              for (i = 2; i < 10000; i++)
11：              {
12：                  sum = 1;
13：                  for (j = 2; j <= i / 2; j++)
14：                  {
15：                      if (i % j == 0)
16：                          sum = sum + j;
17：                  }
18：                  if (sum == i)
19：                  {
20：                      Console.Write("{0} = 1", i);
21：                      for (j = 2; j <= i / 2; j++)
22：                      {
23：                          if (i % j == 0)
24：                              Console.Write("+ {0}", j);
25：                      }
26：                      Console.WriteLine();
27：                  }
28：              }
29：          }
30：      }
31：  }
```

运行结果：

6=1+2+3

28=1+2+4+7+14

496=1+2+4+8+16+31+62+124+248

8128=1+2+4+8+16+32+64+127+254+508+1016+2032+4064

【例 2】 对于任意给定的一个正整数 n，统计其阶乘 n！的末尾中 0 的个数。

程序代码如下：

```
01：  using System;
02：
03：  namespace CSHARP4_11
04：  {
05：      class Program
06：      {
07：          static void Main(string[] args)
08：          {
09：              int n;
10：              int sum = 0;
11：              int i, k;
12：              Console.Write("Pleast input a positive number: ");
13：              n = Convert.ToInt32(Console.ReadLine());
14：              for (i = 5; i <= n; i = i + 5)
15：              {
16：                  int m = i;
17：                  for (k = 0; m % 5 == 0; k++)
18：                      m = m / 5;
19：                  sum = sum + k;
20：              }
21：              Console.WriteLine("The number of zero in {0}! is: {1}", n, sum);
22：          }
23：      }
24：  }
```

运行结果：

```
Pleast input a positive number: 50
The number of zero in 50! is: 12
```