第四章 循环结构程序设计

- **本章导读**

本章从实例分析着手使读者掌握循环结构编程的方法，掌握循环结构的while语句、do-while语句、for语句，强化培养编程思路。

- **本章主要知识点**

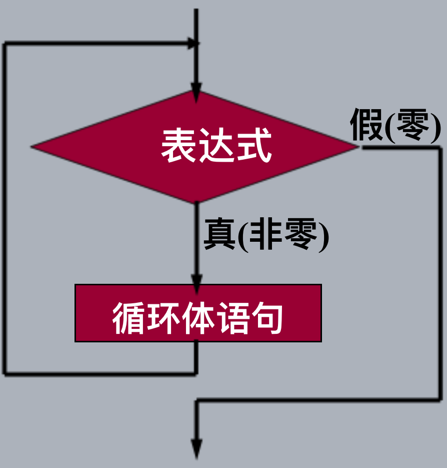
（1）while 语句
（2）do-while 语句
（3）for 语句
（4）循环结构的嵌套
（5）break 和 continue 语句

第四章 循环结构程序设计

4.1 while 语句

while 语句常称为“当型”循环语句。其语句形式如下：

```
while (表达式)
      循环体;
```



4.1 while 语句

说明：

（1） while(表达式) 中的“表达式”可以是任何符合C语言语法的表达式。当表达式的值为“非零”时继续执行循环体；当表达式的值为“零”时结束循环。

（2） while(表达式) 只能自动结合一条语句。当有多条语句时，必须用大括号括起来, 构成复合语句，因为复合语句在语法上相当于一条语句。

（3） while(表达式) 的后面不能随意加分号。如果加了分号，写成 while(表达式)；则表示此时的循环体是空语句，将可能引起程序运行时的逻辑错误。

4.1 while 语句

例1：使用while语句编程计算 “1+2+3+4+ … +100 ” 。

```
#include <stdio.h>
main( )
{
   int i = 1, sum = 0;
   while(i <= 100)
   {
  sum +=  i;
  i ++;
   }
   printf("sum=%d", sum);
}
```

循环的初始条件

循环的终止条件

初始条件向终止条件的变化

思考：

（1）如果没有给变量 i 和 sum 赋初值会有怎样的运行结果?

（2）如果while循环体没有用大括号括起来会有怎样的

4.1 while 语句

例2：从键盘输入一行字符（以回车作为输入结束标志），统计输入字符的个数。

```
#include <stdio.h>
main( )
{  int sum = 0;
   char  x;
   x = getchar( );
   while(x != '\n')
   {
  sum ++;
  x = getchar( );
   }
   printf("sum=%d", sum);
}
```

思考：

（1）如果没有循环体外的语句 x=getchar( ); 会怎样?

（2）如果没有循环体内的语句 x=getchar( ); 会怎样?

4.1 while 语句

总结：

（1） 循环结构程序设计的三要素：

- 循环的初始条件
- 循环的终止条件
- 初始条件变化到终止条件的控制语句

如果某一个要素的语句没有表达正确，则可能引起循环次数不正确、甚至死循环等错误。

4.1 while 语句

总结：

（2）三要素各自的执行次数

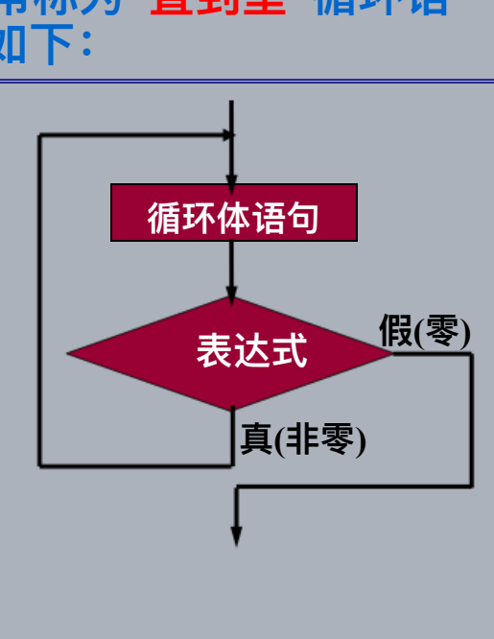
- 循环的初始条件只执行 1 次，且在循环开始前执行。
- 循环的终止条件执行 n+1 次，在每次循环体执行前执行。
- 循环初始条件向终止条件变化的控制语句执行 n 次，与循环体中的其它语句一样执行 n 次。

4.2 do-while 语句

do-while 语句常称为“直到型”循环语句。其语句形式如下：

```
do
{
    循环体;
} while (表达式) ;
```

此处的分号不能省略



4.2 do-while 语句

说明：

（1） do-while语句的执行过程是：先执行循环体，然后再判断表达式，如果表达式的值为真，则继续下一次循环；否则结束循环。

（2）do-while语句同样有循环结构的三要素。循环的初始条件仍然只执行1次；但是循环的终止条件和循环体一样，都是执行n次。

（3） while语句与do-while语句的区别是： 当第一次循环条件为假时，while语句的循环体执行0次，而do-while语句的循环体执行1次。

4.2 do-while 语句

例3：使用do-while语句编程计算 “1+2+3+4+ … +100 ”

```
#include <stdio.h>
main( )
{
   int i = 1, sum = 0;
   do
   {
  sum += i;
  i ++;
   } while(i <= 100);
   printf("sum=%d", sum);
}
```

循环的初始条件

初始条件向终止条件的变化

循环的终止条件

4.2 do-while 语句

例4：从键盘输入一行字符（以回车作为输入结束标志），统计输入字符的个数。

```
#include <stdio.h>
main( )
{  int sum = 0;
   char  x;
   do
   {
  x = getchar( ) ;
      sum ++;
   } while(x != '\n');
   printf("sum=%d", sum-1);
}
```

思考：

如果从键盘输入 abc<回车> 运行结果是什么？为什么？

N 项式求和问题举例（二）

例5：根据以下公式求∏ 的值，当某项小于0.0005时停止迭代。

$$\frac{\Pi}{2}=1+\frac{1}{3}+\frac{1\times2}{3\times5}+\frac{1\times2\times3}{3\times5\times7}+\ldots+\frac{1\times2\times\ldots\times n}{3\times5\times\ldots\times(2n+1)}$$

分析：

（1）该题是N项式求和问题，循环次数不确定。

（2）每一项的分子、分母可以借助前一项的分子、分母求得。

（3）注意某些变量需设初值，并考虑初值是多少?

N 项式求和问题举例（二）

```
main( )
{  float x, sum;
   int  zi, mu, i;
   sum=1;     zi=1;     mu=3;    i=1;  //循环的初始条件
   x = 1.0 * zi / mu;   //求第一次循环的当前项x
   while(x > 0.0005)  //循环的终止条件
   {
       sum = sum + x; //累加当前项
       i++;             //以下4条语句是为下一次循环作准备
       zi = zi * i;
       mu = mu * (2 * i + 1);
       x = 1.0 * zi / mu; //下一次循环做累加的当前项
   }
   printf(“pai = %f\n”,  2 * sum);
}
```

N 项式求和问题举例（一）

N项式求和问题解题方法总结：

（1） 需注意某些变量必须设初值，这些变量是执行第一次循环时放在赋值号右侧的变量，需要用这些变量的初值参与运算。

（2） N项式中的每一项都是由一定规律形成的，特别要注意后一项是否可以借助前一项求出。

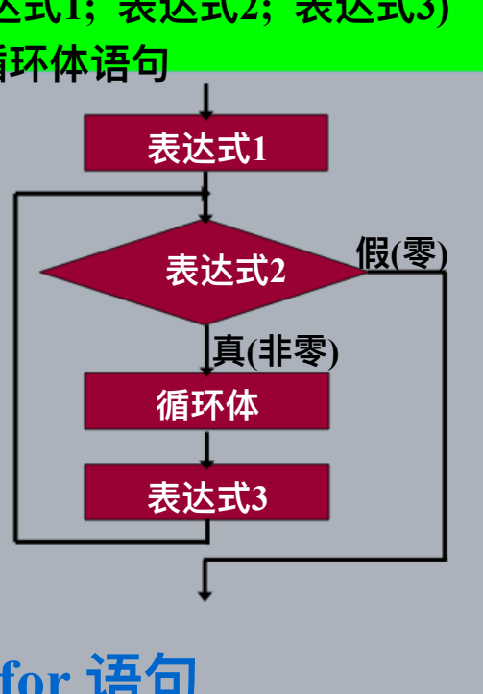
（3） 编程前判断循环次数是否确定? 循环次数确定和不确定两种情况的编程方法不同。

（4） 循环体中需完成的几步操作：①求当前项；②将当前项累加到总和变量里；③为下一次循环作准备。这几个步骤的先后次序根据具体情况需做调整。

4.3 for 语句

for语句是C 语言提供的一种在功能上比前面两种循环语句更强的一种循环语句。其语句形式为：

```
for (表达式1;  表达式2;  表达式3)
     循环体语句
```



4.3 for 语句

说明：

表达式1：也称初始表达式。常用于循环开始前为循环变量设置初值。

表达式2：也称终止表达式。控制循环执行的条件，决定循环何时终止。

表达式3：也称循环表达式。每次循环都执行，通常是初始条件逐步变化到终止条件的控制表达式。

执行次数：“表达式1”执行1次、“表达式2”执行 n+1次、“表达式3”执行n次。

4.3 for 语句

```
int  i=1, sum=0;
while(i <= 100)
{
   sum = sum + i;
   i++;
}
```

等价

```
int  i, sum=0;
for(i = 1; i <= 100; i++)
{
   sum = sum + i;
}
```

说明：

- for语句的三个表达式之间用分号隔开，注意最后一个表达式末尾不能加分号。
- 这三个表达式都可以缺省，如果缺省，则需要在程序中其它地方添加相应的语句。

4.3 for 语句

【for语句的多样性】

(1) 缺省表达式 1

```
int  i = 1, sum=0;
for(  ; i <= 100; i++)
{
   sum = sum + i;
}
```

缺省

(2) 缺省表达式2

```
int  i, sum=0;
for(i = 1 ;  ; i++)
{
   if(i > 100)
       break;
   sum = sum + i;
}
```

缺省

4.3 for 语句

(3) 缺省表达式3

```
int  i, sum=0;
for(i = 1; i <= 100;  )
{
   sum = sum + i;
   i++;
}
```

缺省

(4) 三个表达式都缺省

```
int  i = 1, sum=0;
for(  ;  ;  )
{
   sum = sum + i;
   i++;
   if(i > 100)
       break;
}
```

缺省

4.3 for 语句

(5) 缺省变量赋初值以及循环体为空语句

```
int  i, sum;
for(i=1, sum=0;  i<=100;  sum+=i, i++)
{
   ;
}
```

表达式1表达式逗号表达式

初始表达式

终止表达式

循环表达式

循环体是空语句

(6) 构成死循环的语句形式

```
for( ;  ; )
   循环体语句
```

等价

```
while(1)
   循环体语句
```

4.3 for 语句

while、do-while、for语句的比较：

格式	for(表达式 1；表达式 2；表达式 3) 语句;	while(表达式) 语句;	do {语句; }while(表达式);
循环类别	当型循环	当型循环	直到型循环
循环变量初值	一般在表达式 1 中	在 while 之前	在 do 之前
循环控制条件	表达式 2 的值	表达式的值	表达式的值
提前结束循环	break	break	break
改变循环条件	一般在表达式 3	循环体中用专门语句	循环体中用专门语句

说明：编程时如果循环次数确定，则优先选择for语句；如果循环次数不确定，则优先选择while语句或者do-while语句。

4.3 for 语句

例7：编程求斐波拉契数列的前20项。（斐波拉契数列的第1、2项分别为0、1，以后各项的值都是前两项之和）

分析：

斐波拉契数列的前20项为

0、1、1、2、3、5、8、13、21、34、55、……

- f1--第一个数　f2--第二个数　f3--第三个数
  f3 = f1 + f2;
- 以后只要改变f1，f2的值，即可求出下一个数
  f1 = f2;　　f2 = f3;　　f3 = f1 + f2;

4.3 for 语句

```
#include <stdio.h>
main( )
{
   int   f1=0, f2=1, f3, i;
   printf(“%d\n%d\n”,  f1, f2);  //打印第1、2项
   for(i=0; i<18; i++) //循环18次，求剩余的18项
   {
       f3 = f1 + f2; //每一项是前两项之和
       printf(“%d\n”,  f3); //打印刚求出的项
       f1 = f2; //为下一次循环做准备
       f2 = f3;
   }
}
```

变量 i 的作用是控制循环次数，表达式1、2可以修改

4.3 for 语句

例8：编程求所有的水仙花数。（所谓水仙花数是指一个三位数，其各位数字的立方和等于该数本身。例如：153是水仙花数，因为153=13+53+33）

分析：

- 该题是一个循环次数确定的程序，循环范围是100～999。
- 判断该数是否是水仙花数之前，首先要把一个三位数的个位、十位、百位分离出来。
- 求立方时可以调用库函数 pow()，注意添加头文件 math.h。

```
#include <stdio.h>
#include <math.h>
main( )
{
  int   i, ge, shi, bai, s;
  for(i=100; i<=999; i++) //控制循环范围
  {
      ge = i % 10;
  shi = i / 10 % 10;
  bai = i / 100;
  s = pow(ge, 3) + pow(shi, 3) + pow(bai, 3);
  if(i == s)
           printf(“%d\t”,  i); //打印水仙花数
  }
}
```

4.4 循环结构的嵌套

在一个循环内又完整地包含另一个循环，称为循环的嵌套，即循环体自身包含循环语句。

前面介绍的三种类型的循环结构（while语句、do-while语句、for语句）可以相互嵌套。

例如：

```
while( )
{  …
   while( )
   {…}
}
```

```
while( )
{  …
   do {
      …
   } while( );
   …
}
```

```
for(; ;)
{
   for(; ;)
   {…}
}
```

4.4 循环结构的嵌套

例9：用循环结构编程打印以下图形。

```
*
**
***
****
```

分析：

该图形由4行星号组成，每行星号的个数随着行数的变化而变化。

可以由循环的嵌套结构来解决问题，外层循环控制行数，内层循环控制每行星号的个数。

外循环变量 i (行号)的取值	1	2	3	4
内循环变量 j (星号)的循环次数	1次	2次	3次	4次

由此可推断：变量 j 的取值范围是1 ~ i

4.4 循环结构的嵌套

```
#include <stdio.h>
main( )
{
  int   i, j;
  for(i=1; i<=4; i++) //外循环控制行数
  {
      for(j=0; j<i; j++) //内循环控制星号个数
         printf(“*”); //打印一个星号
      printf(“\n”);       //每行末尾打印回车
  }
}
```

4.4 循环结构的嵌套

例10：用循环结构编程打印该图形

```
   *
  ***
 *****
*******
```

分析：

与上题编程的思路相似，外循环控制行数，有两个并列的内循环，一个内循环控制每行空格的个数，另一个内循环控制每行星号的个数。

外循环变量 i (行号)的取值	1	2	3	4
内循环变量 j (空格)的个数	3次	2次	1次	0次
内循环变量 k (星号)的个数	1次	3次	5次	7次

由此可推断：变量 j 的取值范围是1 ~ 4－i

变量 k 的取值范围是1 ~ 2*i－1

4.4 循环结构的嵌套

```
#include <stdio.h>
main( )
{
  int   i, j, k;
  for(i=1; i<=4; i++)  //外循环控制行数
  {
      for(j=0; j<4-i; j++)       //控制空格个数
          printf(“ ”);          //打印一个空格
      for(k=0; k<2*i-1; k++) //控制星号个数
          printf(“*”);           //打印一个星号
      printf(“\n”);     //每行末尾打印回车
  }
}
```

4.4 循环结构的嵌套

例11：输出由数字组成的如下所示的金字塔图案。

```
        1
       222
      33333
     4444444
    555555555
   66666666666
  7777777777777
 888888888888888
99999999999999999
```

4.4 循环结构的嵌套

```
#include<stdio.h>
void main()
{   int i, k, j;
   for (i = 1; i <= 9; i++) //外循环控制输出行数
   {  for (k = 1; k <= 10 - i; k++)//每行起始输出位置
      {
         printf(" ");   //输出空格符
      }
      for (j = 1;j <= 2 * i - 1; j++)//内循环控制输出个数
      {
         printf("%c",'0' + i);     //输出内容
      }
     printf("\n");              //换行
   }
```

4.5 break和continue语句

break 语句可以用在switch语句中，其作用是跳出switch语句。

break语句也可以用在循环结构中，其作用是结束break所在的那一层循环体，从而提前强制结束本层循环。

break 语句的一般形式是：

break;

4.5 break和continue语句

例12：求300以内能被17整除的最大数。

分析：该题的解题思路是按照 300 □ 1 的顺序依次判断哪个数能被17整除？只要找到第一个满足条件的数就停止循环，因此引入break语句编程。

```
main( )
{   int   i;
   for(i=300; i>=1; i--) //控制待判断的数的范围
   {
       if(i % 17 == 0)   //判断某数是否能被17整除
           break;         //提前结束循环
   }
   printf(“300以内能被17整除的最大数是:%d”, i );
}
```

例13 输出从1~1000内能同时被3和7整除的前10个数。

```
#include<stdio.h>
void main()
{  int i, n = 0;
   for (i = 1; i <= 1000; i++)
   {  if (i % 3 == 0 && i % 7 == 0)
      { printf("%d ", i);
                n++;
                if (n == 10)
           {
           break;
            }
      }
   }
}
```

4.5 break和continue语句

continue 语句可以用在循环结构中，其作用是结束continue所在那一层循环体的本次循环，从而提前进入下一次循环。

只要 continue 语句被执行，那么在执行本次循环时，放在该语句之后的其它语句将被略过，因此称为提前结束本次循环。

continue 语句的一般形式是：

continue;

4.5 break和continue语句

例14：打印300以内不能同时被5和17整除的数。

分析：该题的解题思路是按照 1 □ 300 或者 300 □ 1 的顺序依次判断，如果某个数能同时被5和17整除，则不做任何处理，其余的数打印，因此引入 continue 语句编程。

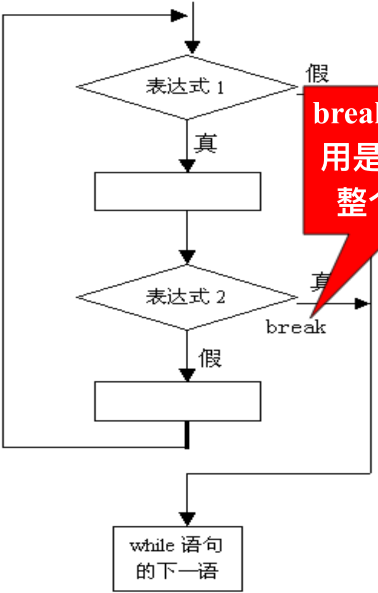
```
main( )
{  int   i;
   for(i=300; i>=1; i--) //或者 for(i=1; i<=300; i++)
   {
       if(i%5==0 && i%17==0)
           continue;
       printf(“%d\t”,  i );
   }
}
```

例15 输出从1~20内不能同时被2和3整除的数。

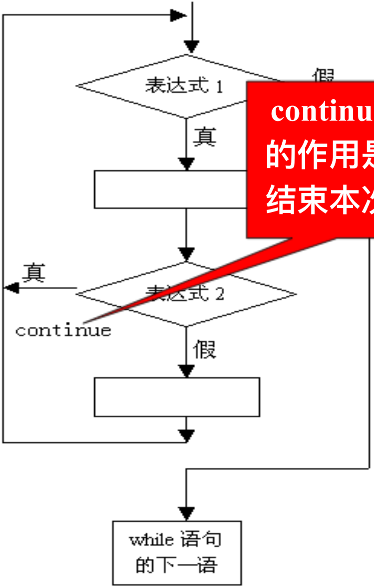
```
#include<stdio.h>
void main()
{
  int i, n = 0;
  for (i = 1; i <= 20; i++)
  {
     if (i % 2 == 0 && i % 3 == 0)
     {
        continue;
     }
     printf("%d ",i);
  }
}
```

4.5 break和continue语句

break 语句和 continue 语句的区别：



break语句的作用是提前结束整个循环体



continue语句的作用是提前结束本次循环

4.5 break和continue语句

重点和难点

例16：从键盘输入一个正整数，判断是否是素数。

分析：如果一个数只能被1和它自己整除，这样的数称为素数（也称质数）。

判断一个数x是否是素数的解题思路是：按照 2□x-1 的顺序依次判断，如果在此范围内只要找到一个数能被x除尽，则说明x不是素数（其余的数将不必再继续判断）；如果在此范围内没有一个数能被x除尽，则说明x是素数。

素数问题是 break 语句的典型应用。

4.5 break和continue语句

方法一：循环变量终值法

```
main( )
{   int   x, i;
  printf(“请输入一个正整数：”);
  scanf(“%d”, &x);
  for(i=2; i<x; i++)
  {
      if(x % i == 0) //如果此条件满足说明x不是素数
            break;
  }
  if(i == x) //循环变量是i，其变化的终值是x
        printf(“%d是素数\n”, x);
  else
        printf(“%d不是素数\n”, x);
}
```

如果执行了break，则循环是异常终止的

如果i==x，则循环是正常终止的

4.5 break和continue语句

思考：程序做如下改动是否正确？为什么？

```
main( )
{   int   x, i;
    printf("请输入一个正整数：");
    scanf("%d", &x);
    for(i=2; i<x; i++)
    {
        if(x % i == 0)
        {
            printf("%d不是素数\n", x);
            break;
        }
        else
            printf("%d是素数\n", x);
    }
}
```

回答：程序如此改动不正确。

若“x%i==0”为真，则的确能说明x不是素数，但并不能立即说明x是素数。

假设 x=9，当第一次循环时i=2，“9%2==0”为假，执行else

}

为假，执行else

4.5 break和continue语句

因此，程序可以设计为如下形式。

```
main( )
{   int   x, i;
    printf("请输入一个正整数：  ");                    (1)
    scanf("%d", &x);
    for(i=2; i<x; i++)
    {
        if(x % i == 0)
        {
            printf("%d不是素数\n", x);
            break;                                    (2)
        }
    }
   if(i == x)
        printf("%d是素数\n", x);
}
```

说明：

判断不是素数的条件比较简单，可以在 if 语句内完成。

判断是素数的条件比较严格，必须在整个循环结束之后才

4.5 break和continue语句

方法二：标记变量法

```
main( )
{  int  x, i, flag=1;  // flag=1 表示假设 x 是素数
   printf("请输入一个正整数：");
   scanf("%d", &x);
   for(i=2; i<x; i++)
   {
       if(x % i == 0)
       {
           flag = 0; // flag=0 表示标志 x 不是素数
           break;
       }
   }
   if(flag == 1)   printf("%d是素数\n", x);
   else            printf("%d不是素数\n", x);
}
```

4.6 循环结构练习题

例17：猜数游戏：任意设置一个整数，请用户从键盘上输入数据猜想设值的数是什么，告诉用户是猜大了还是小了。3 次以内猜对，用户获胜。否则，告诉用户设置的数据是什么。

分析：

（1）该程序的循环次数确定（最多循环3次），优先选择for语句编程。

（2）如果3次以内猜对，则提前结束循环，应使用break语句。

4.6 循环结构练习题

```
#include <stdio.h>
main( )
{
   int  num = 13,  x,  i;   // num是等待猜中的数值
   printf("输入的整数应在"10～20"范围内: \n");
   for(i=0;  i<3;  i++)
   {
  printf("第%d次输入: ",  i+1);
   scanf("%d",  &x);
   if(x == num)      { printf("正确!\n");   break; }
   else if(x > num)    printf("猜大了!\n");
  else                 printf("猜小了!\n");
  }
   if(i==3)   printf("失败! 该数是%d\n",  num);
```

4.6 循环结构练习题

例18: 将一张面值为100元的人民币等值换成100张5元、1元和0.5元的零钞，要求每种零钞不少于1张，问有哪几种组合?

分析：如果用x、y、z来分别代表5元、1元和0.5元零钞的张数，根据题意只能得到下面两个方程：

x+y+z=100

5x+y+0.5z=100

显然从数学上，本题无法得到解析求解，但用计算机便可方便地求出各种可能的解，这类问题属于穷举法问题。

```
#include<stdio.h>
void main()
{    int x, y, z, n;
   n = 0;
   for (x = 1; x < 20; x++)
   { for (y = 1; y < 100-x; y ++)
      { z = 100 - x - y;
                 if (5 * x + y + 0.5 * z == 100)
          { printf("  %-4d  %-4d  %-4d\n",x,y,z);
             n++;          }
      }
   }
   printf("Total  %d",n);
}
```

4代表宽度为4（也就是4位数，不足的话用空格凑，超出的话以实际需求为准），-代表左对齐右边补空格。

4.6 循环结构练习题

（1）以下程序段

```
 int  k=0;
 while(k=1)  k++;
```

该表达式永远为真，因而构成死循环

循环体执行的次数是

A) 无限次　　B) 有语法错，不能执行

C) 一次也不执行　D) 执行1次

A) 无限次

（2）以下程序段

```
 int x=3;
 do
 {  printf("%d  ", x-=2);  }while ( !(--x) );
```

(1) 打印 1，x=1 □ "!(--x)"为真 □ x=0

(2) 打印 -2，x=-2 □ "!(--x)"为假

其输出结果是

A) 1　B) 3 0　　C) 1 -2　D) 死循环

C) 1 -2

4.6 循环结构练习题

（3）以下程序的输出结果是

```
main( )
{    int  a=0, i;
     for(i=1; i<5; i++)
     {    switch(i)
          {  case 0:
              case 3:   a+=2;
              case 1:
              case 2:   a+=3;
              default: a+=5;
          }
     }
     printf("%d\n",a);
}
```

循环过程如下：

- i=1 □ case 1分支 □ a=3 □ default分支 □ a=8
- i=2 □ case 2分支 □ a=11 □ default分支 □ a=16
- i=3 □ case 3分支 □ a=18 □ case 2分支□ a=21 □default分支 □ a=26
- i=4 □ default分支 □ a=31

A) 31　B) 13　C) 10　D) 20

A) 31

4.6 循环结构练习题

（4）有以下程序段

```
main( )
{  int  k=4,n=0;
   for( ; n<k; )
   {  n++;
      if(n%3!=0)
               continue;
      k--;
   }
   printf("%d,%d",k,n);
}
```

循环过程如下：

- n=0, k=4 □ n=1 □ "n%3!=0"为真 □ 执行continue
- n=1, k=4 □ n=2 □ "n%3!=0"为真 □ 执行continue
- n=2, k=4 □ n=3 □ "n%3!=0"为假 □ 执行 k-- □ k=3
- n=3,k=3□"n<k"为假□结束循环

程序运行后的结果是

A) 1, 1　B) 2, 2　C) 3, 3　D) 4, 4

C) 3, 3

4.6 循环结构练习题

（5）以下程序中，while循环的循环次数是

```
main( )
{   int  i=0;
  while(i<10)
  {
   if(i<1)   continue;
  if(i==5)  break;
       i++;
  }
  ......
}
```

循环过程如下：

(1) i=0 □ 表达式"i < 10" 为真 □ 表达式"i < 1" 为真 □ 执行continue 语句

(2) i=0 □ 表达式"i < 10" 为真 □ 表达式"i < 1" 为真 □ 执行continue 语句

由于每次循环都执行continue语句，变量i的值始终为0，因此形成死循环。

A) 1　　B) 10

C) 6　　D) 死循环，不能确定次数

D) 死循环，

实验六

- 商品优惠消费，优惠比例f如右表：
- 已知商品价格x元，求优惠后实际应付多少钱y。

优惠比例(f)	优惠条件
0	X<100(元)
0.05	100<=x<500
0.1	500<=x<2000
0.15	2000<=x

实验解答（方法一）

```
#include<stdio.h>
#include<math.h>
 main()
 {
    float x,y;
    printf("请输入商品价格=");
    scanf("%f",&x);
    while(x<100)
    {
      y=x; break;
    }
```

```
   while(x>=100&&x<500)
       {
         y=x*(1.00-0.05);  break;
       }
       while(x>=500&&x<2000)
       {
          y=x*(1.00-0.1); break;
       }
        while(x>=2000)
       {
         y=x*(1.00-0.15); break;
       }

       printf("优惠后实际应付=%f\n",y);
   }
```

实验解答（方法二）

```
#include <stdio.h>
void main()
{
   int x,y;
   printf("请输入商品价格");
   scanf("%d",&x);
   if(x<100)
   {
     printf("此商品不优惠");
   }
   else if(100<=x&&500>x)
   {
     y=x-x*0.05;

printf("实收价格为:");
     printf("%d-%d*0.05=%d\n",x,x,y);
   }
     else if(500<=x&&2000>x)
     {
        y=x-x*0.1;
           printf("实收价格为:");
        printf("%d-%d*0.1=%d\n",x,x,y);
     }
     else if(x>=2000)
     {
        y=x-x*0.15;
           printf("实收价格为:");
        printf("%d-%d*0.15=%d\n",x,x,y);
     }
}
```

实验解答（方法三）

```
#include <stdio.h>
void main ()
{
   int x,y;
   printf("输入商品价格");
   scanf("%d,",&x);
```

```
    if(x<2000)
        {
           if(x<500)
           {
              if(x<100)
              {
                 y=x;
              }
              else y=0.95*x;
           }
           else y=0.9*x;
        }
        else  y=0.85*x;
        printf("价格为%d\n",y);
    }
```

4.1作业解答（第一种情况）

```
#include<stdio.h>
#include<math.h>
void main()
{
   double t,x,y;
   int i,n;
   t=0;n=1;
   do
   {
       y=1;
       for (i=1;i<=n;i++)
```

```
  {
      y=y*i;
         }
         x=pow(-1,n+1)/y;
         n++;
         t=t+x;
      }while(fabs(x)>0.000001);
      printf("计算结果为%f\n",t);
  }
```

第二种情况

```
#include<stdio.h>
#include<math.h>
main()
{
   double n=1.0,t=1.0,s=0;//定义变量并初始化
   while(fabs(t)>=pow(10,-6))//当t的绝对值大于
0.000001执行循环体//
   {
      t=t/n;//求当前项//
      s=s+t;//累加//
          n++;//为下一次循环做准备
            t=-t;
   }
   printf("s=%lf\n",s);
}
```

第三种情况

```
  #include<stdio.h>
  #include<math.h>
  main()
  {
     float s=1,t,n=1;
     int i=2,sign=-1;
     while(fabs(t)>0.000001)
     {
        n=n*i;
        i++;
        t=sign/n;
        sign=-sign;
        s=s+t;
     }
                                printf("s=%f\n",s);

                                }
```

第四种情况

```
#include<stdio.h>
#include<math.h>
main()
{
   float sum=1,m=1;
   int i=2;
   while(fabs(1.0/m)>0.000001)
   {
      m=m*(-i);
      sum=sum+1.0/m;
      i++;
   }
   printf("sum=%f\n",sum);
}
```

4.4 第三小题

```
#include<stdio.h>
main()
{
   int i,j,k;
   for(i=1;i<=4;i++)
   {
      for(j=1;j<5-i;j++)
         printf(" ");
      for(k=0;k<2*i-1;k++)
         printf("%c",'A'+i-1);
         printf("\n");
   }
}
```

4.4 第七小题

```
#include<stdio.h>
main()
{
   int n,m,i;
   for(n=6;n<=1000;n++)
      {
         m=1;
         for(i=2;i<=n/2;i++)
         {
            if(n%i==0)
               m=m+i;
         }
         if(m==n)
            printf("%d\t",n); } }
```