第 5 章　循环结构程序设计

- 掌握循环结构的基本特点
- 掌握 3 种循环语句：for 语句、while 语句和 do-while 语句
- 学会利用 for 语句、while 语句和 do-while 语句设计简单的循环程序
- 了解 goto 语句构成的循环

5.1　循环的基本概念

【问题】如何计算 1+2+3+4+…+100。

5.1.1　方法的探索

可以实现的方法有很多，但是必须建立计算机能够处理的模型。

我们不妨将变量 s 看做一个初始状态为空的盒子，然后依次向盒子里投入硬币，第一次 1 枚，第二次 2 枚，…，最后一次 100 枚，最后盒子里面的硬币的数目就是要求的结果。

我们不妨用变量 i 记录每次投的硬币数，事实上每次的投币操作都可以看成以下两条语句的执行：

```
s = s + i;   /* 投入 i 枚硬币到 s 中*/
i = i + 1;   /* 计算下次投币数 */
```

将以上两条语句运行 100 次就完成了任务，就相当于投币 100 次。

如何让以上两条语句运行 100 次？当然我们不能写 100 次语句，这就需要循环结构才能解决这个问题，写法如下：

```
s=0;i=1;
while(i<=100)
{
   s=s+i;
   i=i+1;
}
```

5.1.2　循环结构语句

重复执行语句需要构造循环结构，在 C 语言中循环语句共有 3 个，即 for 语句、while 语句和 do-while 语句。

5.2　while 循环

while 循环通过 while 语句实现。while 循环又称为“当型”循环。

while 语句的一般格式为：

```
while (表达式)
    语句
```

其中，括号后面的语句可以是一条语句，也可以是复合语句。它们都称为循环体。

while 语句的执行过程为：

（1）计算并判断表达式的值。若值为 0，则结束循环，退出 while 语句；若值为非 0，则执行循环体。

（2）转步骤（1）。

流程图如图 5-1 所示。

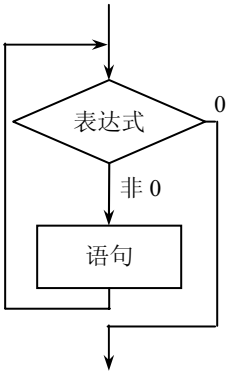

图 5-1　while 循环流程图

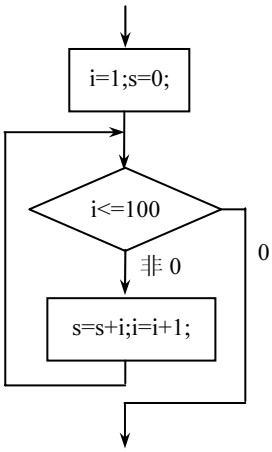

图 5-2　计算 1+2+3+…+100 的循环流程图

【例 5-1】计算 s=1+2+3+…+100。

计算流程图如图 5-2 所示。

```
/*  e5-1.c   */
#include <stdio.h>
void main()
{
   int i,s;
   i=1;
   s=0;
   while(i<=100)       /* 循环控制 */
   {
         s=s+i;
         i=i+1;
   }
   printf("s=%d\n",s);
}
```

```
s=5050
```

图 5-3　例 5-1 的运行结果

程序的运行结果如图 5-3 所示。

注意：

（1）循环体包括一条或多条语句，多条语句必须用一对花括号“{}”括起来。

（2）合理的循环是有限次循环。如果循环不能退出，则称为“死循环”，在程序设计中

应该避免出现。例如，上例中的循环条件为 i<=100，i 从 1 逐渐增加到 100，当 i 等于 101 时，不满足 i<=100 的条件从而退出循环。如果将循环条件改成 i>=1，由于 i 每次都是加 1，其趋势为递增，所以条件等于虚设，循环将一直执行下去，变成“死循环”。

（3）控制循环执行的次数因素包括循环中的循环条件、控制循环的主要变量的初值和终值以及每次变化的幅度等。例如，上例中 i 有效地控制了循环的运行，i 从 1 循环到 100，每次加 1，循环运行了 100 次，i 也可以称为循环变量。

如果只有一个循环变量，而且循环变量每次有固定的增加和减少，则循环的次数可以用以下公式计算:

循环次数 =(终值-初值)/ 步长 +1

步长为循环变量每次增加或减少的值，例如，上例循环次数为：(100–1)/1 + 1，即 100 次。步长可以为负数，例如，以下程序语句:

```
int i,s;
i=100;s=0;
while(i>=1)            /* 循环控制 */
{
   s=s+i;
   i=i-1;
}
…
```

相当于先投 100 枚，每次递减 1 枚，最后一次投入 1 枚。因此根据循环变量的增减特性可以将循环分为递增和递减循环。

【思考】

（1）如何修改以上程序计算 1 到 1000 的数的和？

（2）如何修改以上程序计算 1 到 100 之间所有奇数的和？

【例 5-2】计算 1 到 100 之间所有 3 的倍数的和。

```
/*  e5-2.c   */
#include <stdio.h>
void main()
{
   int i,s;
   i=3;s=0;
   while(i<=99)            /* 循环控制 */
   {
      s=s+i;
      i=i+3;
   }
   printf("s=%d\n",s);
}
```

```
s=1683
```

图 5-4　例 5-2 的运行结果

程序的运行结果如图 5-4 所示。

事实上，语句可以再继续复杂化，在循环中加入选择结构语句 if … else …来解决问题，例如，上面的程序也可以设计成：

```
#include <stdio.h>
void main()
```

```
{
   int i,s;
   i=1;s=0;
   while(i<=100)
   {
      if (i%3==0)            /* 判断是否为 3 的倍数 */
      s=s+i;
      i=i+1;
   }
   printf("s=%d\n",s);
}
```

5.3 do-while 循环

do-while 循环是循环的另外一种形式，又称为“直到型”循环。

do-while 语句的一般格式为：

```
do
{
 语句
} while(表达式);
```

do-while 语句的执行过程为：先执行循环体语句再判断表达式的值。若值为 0，则结束循环，退出 do-while 语句；若值为非 0，则继续执行循环体。

流程图如图 5-5 所示。

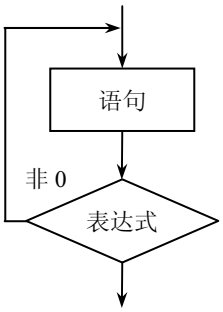

图 5-5　do-while 循环流程图

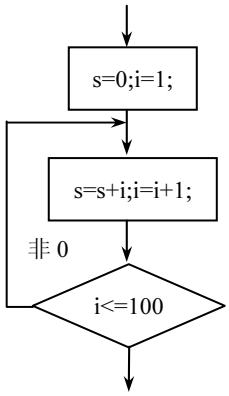

图 5-6　计算 1+2+3+…+100 的循环流程图

【例 5-3】计算 s=1+2+3+…+100。

计算流程图如图 5-6 所示。

```
/* e5_3.c  */
#include <stdio.h>
void main()
{
   int i,s;
   i=1;s=0;
   do
```

```
    {
      s=s+i;
      i=i+1;
    } while(i<=100);       /* 循环控制 */
    printf("s=%d\n",s);
  }
```

程序的结果同例 5-1。

注意：

（1）do-while 循环和 while 循环可以完成相同的任务。例如上面的程序都可以计算出 1 到 100 的数的和。

（2）do-while 循环的循环条件的判断在循环体的后面，所以和 while 循环有区别，例如下面的两个程序：

```
int i=1;s=0;                        int i=1,s=0;
while（i<1）                        do
{                                   {
   s=s+i;                              s=s+i;
   i=i+1;                              i=i+1;
}                                   } while(i<1);
printf（"s=%d\n",s）;               printf("s=%d\n",s);
```

左边的程序运行结果为：s=0，而右边的程序运行结果为 s=1。

这是由于 do-while 循环的循环体至少运行一次后再判断循环条件是否为真，从而决定是否退出循环；while 循环首先判断循环条件是否满足，所以当第一次运行时条件为假时就立即退出循环，从而可能循环次数为 0。

【思考】观察下面的程序，其运行结果是什么？

```
#include <stdio.h>
void main()
{
   int i=1,s=0;
   do
   {
      if(i%2)
      s=s+i;
      i=i+1;
   } while(i<=3);
   printf("s=%d\n",s);
}
```

由于 i 有两次满足 i%2 条件的机会，所以最后 s=4。如果循环条件改为 i<1，则 s=1，改成 while 循环则 s=0。注意 i%2 相当于 i%2==1。

5.4 for 循环

for 循环是循环的一种标准形式，其语法如下：

```
for（①;②;③）
 ④
```

- 表达式①通常用于循环的初始化。包括循环变量的赋初值、其他变量的准备等。
- 表达式②循环的条件判断式，如果为空则相当于真值。
- 表达式③，通常设计为循环的调整部分，主要是循环变量的变化部分。
- 循环体④由一条或多条语句构成，多条语句需要用一对花括号括起来。

执行次序如图 5-7 所示。

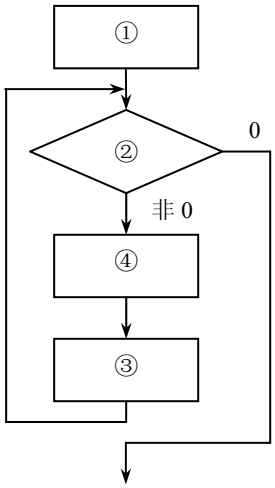

图 5-7　for 循环流程图

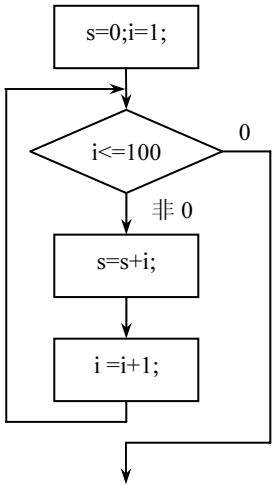

图 5-8　计算 1+2+3+…+100 的循环流程图

【例 5-4】计算 s=1+2+3+…+100。

计算流程如图 5-8 所示。

```
/*  e5_4.c   */
#include <stdio.h>
void main()
{
   int i,s;
   for(i=1,s=0;i<=100;i++)
      s = s + i;
   printf("s=%d\n",s);
}
```

程序的运行结果同例 5-1。

注意：

（1）for 循环可以用以下 while 循环代替:

```
 ①;
 while (②)
{
  ④;
  ③;
}
```

（2）表达式①可以是多个表达式构成的逗号表达式，例如 i=1,s=0;。

（3）①、②、③构成循环的控制部分，3 个表达式之间用分号分隔。

（4）表达式①可以放在 for 循环的前面，但后面的分号不能少，例如:

```
①;
for（;②; ③）④;
```

（5）表达式②也可以省略，相当于②始终为真值，从而构成无条件循环，循环将不能终止，需要采取其他措施。

（6）表达式③也可以省略，但作为循环变量的调整功能不能缺少，可以在循环体中完成，例如下面的 for 循环。

```
for (i=1,s=0;i<=100;)
   s = s + i++;
```

（7）如果表达式①和③都省略的话，相当于 while 循环，例如下面的程序形式:

```
i=1,s=0;
for (;i<=100;)  /*相当于 while(i<=100)*/
   s = s + i++;
```

（8）①、②、③均省略，即:

```
for (;;) ④;。
```

相当于 while (1) ④;。循环的所有控制和计算功能都必须在循环体④中完成，这样的循环适合于随机退出循环程序的情况。

（9）表达式④也可以省略，但必须至少保留一个分号，即:

```
for (①;②;③) ;
```

一个分号即是一条空语句。

（10）如果①、②、③、④均省略，即如以下形式:

```
for (;;);
```

这将构成一个死循环。

for 循环是一种优秀的循环结构，是 3 种循环语句中形式上最为规范的一种循环结构，C 语言给与 for 循环非常灵活的形式和强大的功能，比其他语言要强得多。

for 循环的 4 个部分并不是严格划分的，允许有一定的交叉，但不建议破坏划分的功能结构，在程序设计中尽量遵守，从而使程序易于控制和维护，并且具有其他两种循环难得的易读性。

【例 5-5】计算 1+1.5+2.0+2.5+3.0+3.5+…+10。

```
/*  e5_5.c   */
#include <stdio.h>
void main()
{
   float i, s;   /*i,s 需要定义为 float 型 */
   for(i=1,s=0;i<=10;i=i+0.5)
      s = s + i;
   printf("s=%f\n",s);
}
```

程序的运行结果如图 5-9 所示。

```
s=104.500000
```

图 5-9　例 5-5 的运行结果

5.5 循环嵌套

循环体可以被复杂化为另外一个循环，这就是循环的嵌套，例如下面的嵌套形式：

（1）
```
while ()
{…
  while ()
  …
}
```

（2）
```
for (;;)
{
  …
  for (;;)
  …
}
```

（3）
```
do{
 …
 do{
 …
 }while ();
 …
}while ();
```

（4）
```
while ()
{…
  for (;;)
  …
}
```

（5）
```
for (;;)
{
  …
  while (;;)
  …
}
```

（6）
```
do{
 …
 for (;;);
 …
 }while ();
```

循环嵌套实际上是语句的复杂化，循环原来的一条语句复杂化成另外一个循环结构。

【例 5-6】计算 s=1+(1+2)+(1+2+3)+…+(1+2+3+4+…+10)。

```
/*  e5_6.c   */
#include <stdio.h>
void main()
{
   int i,j,s;
   for(i=1,s=0;i<=10;i++)
     for(j=1;j<=i;j++)
        s=s+j;
   printf("s=%d\n",s);
}
```

```
s=220
```

图 5-10　例 5-6 的运行结果

程序的运行结果如图 5-10 所示。

以上程序由两个 for 循环嵌套构成，外面的循环 i 从 1 到 10，里面的循环 j 从 1 到 i，执行过程如表 5-1 所示。

表 5-1　循环嵌套中变量跟踪表

i	j	s
1	1	(1)
2	1	(1)+(1)
	2	(1)+(1+2)

续表

i	j	s
3	1	(1)+(1+2)+(1)
	2	(1)+(1+2)+(1+2)
	3	(1)+(1+2)+(1+2+3)
…	…	…
10	1	(1)+(1+2)+(…)+(1)
	2	(1)+(1+2)+(…)+(1+2)
	3	(1)+(1+2)+(…)+(1+2+3)
…	…	…
	10	(1)+(1+2)+(…)+(1+2+3+…+10)

j 的终值是 i 的值，从而每次内循环计算的和的范围由外循环的循环变量 i 决定，这就是循环的特点：重复执行相同的语句，但并非重复相同的运算，循环体的计算是变化的，当然这种变化是有规律的、受循环控制的。这就像绕操场跑步一样，同样跑 10 圈，而每圈跑的步数是不一样的，相当于 10 次外循环内有 n 步小循环，由于体力下降，n 可能每次在增加。

5.6 break 语句、continue 语句和 goto 语句

5.6.1 break 语句

switch 结构中可以用 break 语句跳出结构去执行 switch 语句的下一条语句。实际上，break 语句也可以用来从循环体中跳出，常常和 if 语句配合使用。例如：

```
for（i=1;i<100;i++）
  if（i>100）break;
```

当变量 i>100 时退出循环。

break 语句不能用于循环语句和 switch 语句之外的任何其他语句中。

5.6.2 continue 语句

与 break 语句退出循环不同的是，continue 语句只结束本次循环，接着进行下一次循环的判断，如果满足循环条件，继续循环，否则退出循环。

【例 5.7】阅读下面程序，写出运行结果。

```
/*  e5_7.c   */
#include <stdio.h>
void main()
{
   int i,s;
   for(i=1,s=0;i<=10;i++)
   {
```

```
    if(i%2==0)
      continue;
    if(i%10==7)
      break;
    s=s+i;
   }
   Printf("s=%d\n",s);
}
```

程序流程图如图 5-11 所示。

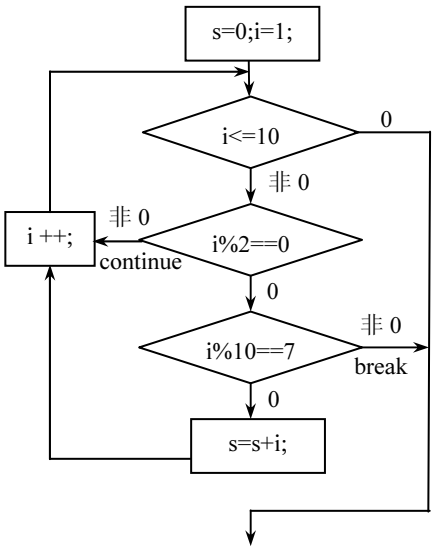

图 5-11　例 5-7 程序流程图

程序的运行结果如图 5-12 所示。

程序中当 i 是偶数的时候，结束本次循环，继续下一个循环；当 i 的个位数是 7 的时候结束循环退出；其他情况累加到 s 中，所以实际累加的数只有 1、3、5，结果为 9。

```
s=9
```

图 5-12　例 5-7 的运行结果

5.6.3　goto 语句

goto 语句为无条件转向语句，形式为：

goto 语句标号

语句标号用标识符表示，命名规则同变量名。例如下面的程序段：

```
i=1;
s=0;
sum:if (i<=10)
{
  s=s+i;
  i=i+1;
  goto sum;  /* sum 就是标识符*/
}
…
```

但对于结构化程序的设计，不主张使用 goto 语句，否则会导致程序流程混乱、可读性差，一般用在特殊的场合，且不宜多用。

5.7 程序举例

【例 5-8】计算 s=1×2×3×4×…×8。

题目实际上是求 8!，程序如下：

```
/*  e5_8.c   */
#include <stdio.h>
void main()
{
   int i;
   long s;
   for(i=1,s=1;i<=8;i++)
         s = s * i;
   printf("s=%ld\n",s);
}
```

程序的运行结果如图 5-13 所示。

计算阶乘的方法与求和差不多，但求和时累加器 s 初始化为 0，求阶乘时累乘器初始化为 1，循环的构造很相似，由于阶乘的值很容易放大，所以数据类型定义为 long 型。

【例 5-9】打印如图 5-14 所示的图形。

```
s=40320
```

图 5-13　例 5-8 的运行结果

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

图 5-14　例 5-9 要求打印的图形

【分析】程序需要输出 5 行星号，但每行输出的个数不等，其规律如下：

行数 = 星号数

利用循环的嵌套可以完成，程序如下：

```
/*  e5_9.c   */
#include <stdio.h>
void main()
{
   int i,j;
   for (i=1;i<=5;i++)
   {
         for (j=1;j<=i;j++)
               printf("*");
      printf("\n");    /* 每输出一行需要换行 */
   }
}
```

【思考】如何输出如图 5-15 所示的图形。

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

图 5-15　思考题要求打印的图形

【分析】对于此类图形的输出通常需要嵌套的循环才能解决问题，其中需要找到以下规律：

（1）星号前的空格个数与行数之间的关系。

（2）星号个数与行数之间的关系。

假设行号 i 从 0 开始循环到 3，则本题规律如下：

（1）星号前的空格个数为 i 个。

（2）星号个数为 2*i+1 个。

程序自然设计为：

```
for(i=0;i<=3;i++)              /*控制输出 4 行*/
{
   for(j=0;j<i;j++)       /*输出 i 个空格*/
   printf("□");           /*□表示空格*/
   for(j=0;j<2*i+1;j++)  /*输出 2*i+1 个星号*/
   printf("*");
   printf("\n");               /*输出一行后需要换行*/
}
```

【例 5-10】计算 100 以内的所有素数之和。

【分析】素数可以从定义来判断，除了 1 和本身之外，没有其他因子，所以程序的任务是依次判断 1 到 100 之间所有的数是否为素数，如果是，将其累加。最后输出累加的和。

需要嵌套的循环，外循环控制产生 1 到 100 的数的循环，循环变量的值也正是内循环需要判断的对象；内循环首先判断当前的循环变量的值是否为素数，是则累加。循环结束后输出累加的和。

程序如下：

```
/*  e5_10.c */
#include <stdio.h>
void main()
{
   int i,j,s=0;
   for(i=2;i<=100;i++)        /* 设置循环产生 2～100 之间的数 */
   {
         for(j=2;j<=i-1;j++)  /* 用 2 ～ i-1 的数去除 i */
               if(i%j==0)
                     break;           /* 有能整除 i 的 j，说明 i 不是素数，退出 */
         if(j>i-1)                    /* i 是素数，因为 2 ～ i-1 没有 i 的因子 */
               s=s+i;
   }
   printf("%d\n",s);
}
```

程序的运行结果如图 5-16 所示。

```
1060
```

图 5-16　例 5-10 的运行结果

以上程序中 j<=i-1 也可以改成 j<=i/2 或 j<=sqrt(i)。

【例 5-11】计算 Fibonacci 数列前 20 项的和。

Fibonacci 数列的特点是：前两个数为 1 和 1，从第 3 个数开始，每个数都是前面两个数的和，即：

F1=1,F2=1　（n=1 或 2）

Fn=Fn-1+Fn-2　（n>=3）

很显然，Fibonacci 数列依次为：1,1,2,3,5,8,13,21,34…

程序如下：

```
/*  e5_11.c   */
#include <stdio.h>
void main()
{
   int f1,f2,f;
   int i;
   long s;
   f1=f2=1;
   s=f1+f2;
   for(i=1;i<=18;i++)  /*已经有两个数，只要再产生 18 个数即可*/
   {
         f=f1+f2;   /*得到一个新数*/
         s=s+f;
         f1=f2;    /*重置两个数*/
         f2=f;
   }
   printf("%ld\n",s);
}
```

程序的运行结果如图 5-17 所示。

```
17710
```

图 5-17　例 5-11 的运行结果

循环结构是面向过程编程中 3 种结构中最重要的一种结构，学好它是学好这门课程的关键。本章介绍的内容主要包括:

（1）3 种循环结构 while、do-while 和 for 循环（goto 也可以构成循环，通常不用）。

（2）break 语句、continue 语句和 goto 语句。

（3）while 循环和 do-while 循环的条件判断一个在前，一个在后，为导致循环体执行的次数不同，需要密切注意。

（4）for 循环为标准的功能很强的循环，通常用于可控制的循环，对于程序的维护和阅读都是最佳选择。

（5）break 语句和 continue 语句可以改变循环运行的方向，主要用于特殊情况的处理，但不能控制 if 和 goto 构成的循环。

循环结构的实质是重复执行一系列语句，这种重复性是在循环条件的控制之下完成的，目的是完成指定的任务，所以利用循环结构设计程序的关键就在于如何控制循环的条件，在恰当的时机由“真”变“假”，从而退出循环。

习题五

一、选择题

1．for(i=0;i<10;i++);结束后，i 的值是（　　）。

A．9　　B．10　　C．11　　D．12

2．下面的程序段循环次数是（　　）。

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

A．5　　B．6

C．4　　D．死循环，不能确定次数

3．下面程序的输出结果是（　　）。

```
#include <stdio.h>
void main()
{
   int s,k;
   for(s=1,k=2;k<5;k++)
      s+=k;
   printf("%d\n",s);
}
```

A．1　　B．9　　C．10　　D．15

4．要使下面程序段输出 10 个整数，则在下划线处填入正确的数是（　　）。

```
for(i=0;i<=______;)
   printf("%d\n",i+=2);
```

A．9　　B．10　　C．18　　D．20

5．运行下列程序：

```
#include <stdio.h>
```

```
void main()
{
   int i=10,j=0;
   do
   {
      j=j+i;
      i--;
   }while(i>5);
   printf("%d\n",j);
}
```

则输出结果是（　　）。

A．45　　B．40　　C．34　　D．55

6．运行下列程序：

```
#include <stdio.h>
void main()
{
   int  k=0,a=1;
   while(k<10)
   {
      for(;;)
      {
         if((k%10)==0)
            break;
         else
            k--;
      }
      k+=11;
      a+=k;
   }
   printf("%d %d\n",k,a);
}
```

则输出结果是（　　）。

A．21 32　　B．21 33　　C．11 12　　D．10 11

7．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 后的表达式为非零时结束循环

D．用 do-while 语句构成的循环，在 while 后的表达式为零时结束循环

8．有如下程序：

```
#include <stdio.h>
void main()
{
   int x=3;
   do
```

```
    {
      printf("%d",x--);
    }while(!x);
  }
```

该程序的执行结果是（　　）。

A．3　2　1　　B．2　1　0　　C．3　　D．2

9．若 k 为整型变量，则下面 while 循环执行的次数为（　　）。

```
k=10;
while（k==0）k=k-1;
```

A．0 次　　B．1 次　　C．10 次　　D．无限次

10．下面有关 for 循环的正确描述是（　　）。

A．for 循环只能用于循环次数已经确定的情况

B．for 循环是先执行循环体语句，后判断表达式

C．在 for 循环中，不能用 break 语句跳出循环体

D．for 循环的循环体语句中，可以包含多条语句，但必须用花括号括起来

11．对 for(表达式 1;　; 表达式 3)可理解为（　　）。

A．for(表达式 1;0;表达式 3)　　B．for(表达式 1;1;表达式 3)

C．for(表达式 1;表达式 1;表达式 3)　D．for(表达式 1;表达式 2;表达式 3)

12．若 i 为整型变量，则以下循环执行次数是（　　）。

```
for(i=2; i==0;)  printf("%d",i- -);
```

A．无限次　　B．0 次　　C．1 次　　D．2 次

13．以下循环体的执行次数是（　　）。

```
#include <stdio.h>
void main()
{
  int i,j;
  for(i=0,j=3;i<=j;i+=2,j--)
    printf("%d \n",i);
}
```

A．3　　B．2　　C．1　　D．0

14．执行以下程序后，输出结果是（　　）。

```
#include <stdio.h>
void main()
{
  int y=10;
  do {y--;} while(--y);
  printf("%d\n",y--);
}
```

A．-1　　B．1　　C．8　　D．0

15．以下程序的输出结果是（　　）。

```
#include <stdio.h>
void main()
{
```

```
    int a,b;
    for(a=1,b=1;a<=100;a++)
    {
      if(b>=10)
        break;
      if(b%3==1)
      {
        b+=3;
        continue;
      }
    }
    printf("%d \n",a);
  }
```

A．101　　B．3　　C．4　　D．5

二、填空题

1．循环的 3 个常见语句分别是_____、_____和_____。

2．下面程序的运行结果为_____。

```
#include <stdio.h>
void main()
{
  int a=10, y=0;
  do
  {
    a+=2; y+=a;
    if (y>50) break;
  } while (a<14);
  printf("a=%d, y=%d\n", a, y);
}
```

3．从键盘输入 1□2□3□4□5□-1<回车>，“□”代表空格，则下面程序的运行结果是_____。

```
#include <stdio.h>
void main()
{
  int n, k=0;
  do
  {
    scanf("%d", &n);
    k+=n;
  }while (n!=-1);
  printf("k=%d, n=%d", k, n);
}
```

4．下面程序的运行结果为_____。

```
#include <stdio.h>
void main()
```

```
{
  int i, j, s=0;
  for ( i=1, j=5; i<j; i++, j-- )
    s+=i*10+j;
  printf("%d\n", s);
}
```

5．下面程序的运行结果为_____。

```
#include <stdio.h>
void main()
{
  int i=10, s=0;
  for ( ; --i; )
    if ( i%3==0)
    s+=i;
  s++;
  printf("s=%f\n", s);
}
```

6．下面程序的运行结果为_____。

```
#include <stdio.h>
void main()
{
  int a=2, n=5, s;
  s=a;
  for ( ; --n; )
    s=s*10+a;
  printf("%d", s);
}
```

7．下面程序运行时，循环体语句“a++;”运行的次数为_____。

```
#include <stdio.h>
void main()
{
  int i, j,a=0;
  for ( i=0; i<2; i++ )
    for ( j=4; j>=0; j-- )
       a++;
}
```

8．下面的程序运行后，a 的值为_____。

```
#include <stdio.h>
void main()
{
  int i, j,a=0;
  for ( i=0; i<2; i++ ) a++;
    for ( j=4; j>=0; j-- ) a++;
}
```

9．下面程序的运行结果为_____。

```
int i=1, s=3;
do
{
   s+=i++;
   if (s%7==0)  continue;
   else ++i;
} while (s<15);
printf("%d", i);
```

10．当运行以下程序时，从键盘输入 China#<回车>，则下面程序的运行结果是_____。

```
#include <stdio.h>
void main()
{
   int v1=0, v2=0;
   char c;
   while ( (c=getchar())!= '#')
   {
      switch (c)
       {
         case 'a':
         case 'h':
         default : v1++;
         case 'o': v2++;
      }
   }
   printf("%d,%d\n", v1, v2);
}
```

三、改错题

1．for(i=0,i<5,i++) j++;

2．while(j<10);{j++;i=j;}

3．do{j++;a=j;}while(j<10)

4．用下列程序段实现求 5!：

```
int s=1,i=1;
while(i<=5)
s*=i;
i++;
```

5．下列程序段实现求半径 r=1 到 r=10 的圆面积，直到面积大于 100 为止。

```
for(r=1;r<=10;r++)
{
   s=3.14159*r*r;
   if(s>100) continue;
   printf("%f",s);
}
```

四、编程题

1．求 1-2+3-4+5-6+7+…+99-100。

2．任意输入 10 个数，分别计算输出其中正数和负数的和。

3．计算 1～100 以内所有含 6 的数的和。

4．输出所有的三位水仙花数。所谓水仙花数是指所有位的数字的立方之和等于该数，例如：

$153=1^3+5^3+3^3$

5．编写程序输出下面的图形。

```
1
23
456
7890
```

6．编写程序输出下面的图形。

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