第 5 章　循环结构程序设计

教学目标

掌握 while 语句、do-while 语句和 for 语句的格式和执行过程，能够运用这些语句编写出循环结构程序，并掌握循环结构程序设计的基本方法。

本章要点

- while 语句
- do-while 语句
- for 语句
- continue 及 break 语句
- 循环结构程序设计

人们在处理事务过程中，常常需要完成重复性、规律性的操作，例如求若干数的和、求一个数的阶乘值等，这些都需要用到循环语句。几乎所有的实用程序都包含循环。循环结构是结构化程序设计的基本结构之一，它和顺序结构、选择结构共同作为各种复杂程序的基本构造单元。因此熟练掌握选择结构和循环结构的概念及使用是程序设计的最基本的要求。

在 C 语言中有三种循环语句：while 语句，do-while 语句和 for 语句。另外，用 goto 语句和 if 语句也可以构成循环。

5.1　while 语句

while 语句是通过判断循环控制条件是否满足来决定是否执行循环。其一般形式如下：

```
while(表达式)
    循环体语句
```

这里，表达式为循环控制条件，当表达式的值为非 0 值(表示循环条件满足)，就执行 while 中的循环体语句；当表达式的值为 0 值(表示循环条件不满足)，就转去执行 while 语句的下一句。while 语句的控制流程如图 5-1 所示。

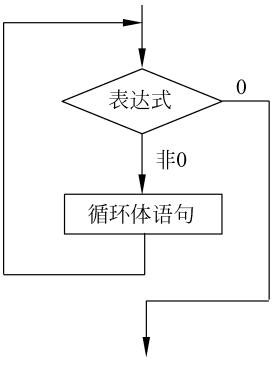

图 5-1　while 语句控制流程

从图 5-1 中看出，当程序执行到 while 语句时，首先计算表达式，如果表达式的值为非 0，就执行循环体语句，然后自动回到表达式处再进行表达式的计算，若表达式的值还为非 0 值，再执行循环体语句，如此反复直到表达式的值为 0，才结束循环，转去执行程序中 while 语句的下一句。对于 while 语句，若表达式的值一开始就为 0，则循环体语句一次都不被执行。

C语言规定，循环体语句只能是一条语句，如果由多条语句构成一个循环体，应该用大括号括起来构成一条复合语句。

例 5.1 写一个程序，输入10个学生的成绩，求平均成绩。

算法分析：

(1) 人数从0计。

(2) 当“人数<10”时，做以下操作：

① 输入一个分数；

② 累计总分；

③ 人数加1。

(3) 重复第(2)步操作，直到“人数为10”结束。

参考程序如下：

```
#include<stdio.h>
main()
{   int n=0;                          /*n用来存放学生数,初值为0*/
    float score,average=0;            /* average用来存放平均成绩,初值为0*/
    while(n<10)                       /*当没有输入完10个学生成绩时继续循环*/
    {   scanf("%f",&score);           /*输入一个学生的分数*/
        average+=score;               /*累计总分*/
        n++;                          /*学生数加1*/
    }
    average/=n;                       /*求平均成绩average*/
    printf("%6.2f\n",average);        /*输出average,保留两位小数*/
}
```

5.2 do-while语句

do-while语句是先执行循环体语句，再通过判断表达式的值来决定是否继续循环。其一般形式如下：

```
do
    循环体语句
while(表达式);
```

do后面是循环体语句，while后面的“表达式”为循环控制条件。

do-while语句的执行过程是：先执行一次循环体语句，然后计算表达式的值，当表达式的值为非0(“真”)时，便重复执行一次循环体语句。如此反复，直到表达式的值为0时，结束循环。do-while语句控制流程如图5-2所示。

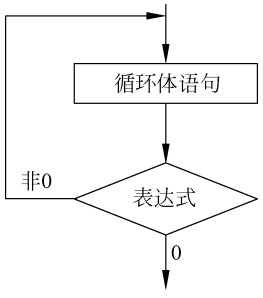

图5-2 do-while语句控制流程

do-while中的循环体语句至少要被执行一次，因为它是先执行循环体语句，再判断表达式。当循环体部分由多条语句组成时，也必须用大括号括起来，使其构成一条复合语句。

对于例5.1用do-while语句编写程序如下：

```
#include<stdio.h>
main()
{   int n=0;                          /*n用来存放学生数,初值为0*/
    float score,average=0;            /* average用来存放平均成绩,初值为0*/
    do
    {   scanf("%f",&score);           /*输入一个学生的分数*/
        average+=score;               /*累计总分*/
        n++;                          /*学生数加1*/
    } while(n<10);                    /*当没有输入完10个学生成绩时继续循环*/
    average/=n;                       /*求平均成绩average*/
    printf("%6.2f\n",average);        /*输出average,保留两位小数*/
}
```

注意：在使用 while 语句和 do-while 语句时，循环控制表达式中的变量值，必须在循环体内有所改变，否则会造成死循环。例如：

```
i=1;
while(i<=10)
    printf("%3d\n",i);
    i++;
```

这个循环永远不会结束，是一个死循环，不断输出“1”这个值。因为语句“i++;”不属于循环体中的语句，循环控制表达式中的 i 值没有在循环体内被改变。应该改为：

```
i=1;
while(i<=10)
{   printf("%3d\n",i);
    i++;
}
```

这条循环语句执行的结果是：以每个数占 3 个字符宽输出 1～10 的 10 个数。也可以将它改成 do-while 语句：

```
i=1;
do
{   printf("%3d\n",i);
    i++;
}while(i<=10);
```

5.3 for 语句

5.3.1 for 语句的一般形式

for 语句是 C 语言中最有特色的循环语句，使用最为灵活方便，因此是程序中为了实现循环而使用最多的循环语句。for 语句的一般形式为：

```
for(表达式1;表达式2;表达式3)
    循环体语句
```

其控制流程如图 5-3 所示。

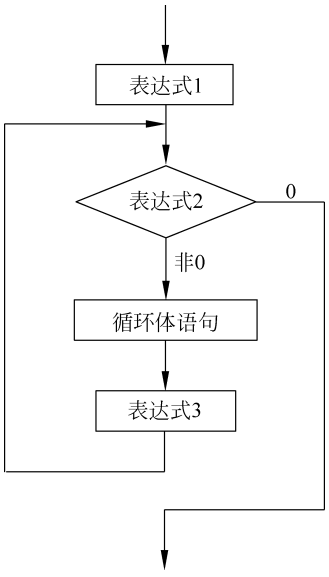

图 5-3 C 程序设计的一般步骤

5.3.2 for 语句中的各部分含义

表达式 1：初值表达式，用于循环开始前，为循环变量设置初始值。

表达式 2：循环控制表达式，它控制循环执行的条件，决定循环次数。

表达式 3：修改循环控制变量值表达式。

循环体语句：被重复执行的语句。

5.3.3 for 语句的执行过程

for 语句的执行过程如下：

(1) 计算表达式 1 的值；

(2) 计算表达式 2 的值，若表达式 2 的值为 0("假")，则结束 for 循环；

(3) 执行循环体语句；

(4) 计算表达式 3，然后转向步骤(2)。

for 语句中循环体部分由多条语句组成时，也必须用大括号括起来，使其构成一条复合语句。

对于例 5.1 用 for 语句编写程序如下：

```
#include<stdio.h>
main()
{   int n;
    float score,average=0;
    for(n=1;n<=10;n++)                    /*从第一个学生到最后一个学生*/
    {   scanf("%f",&score);
        average+=score;
    }
    average/=10;                          /*求平均成绩*/
    printf("%6.2f\n",average);
}
```

思考：这里求平均成绩的语句是"average/=10;"，而不是例 5.1 中的"average/=n;"，为什么要做这样的改变？

5.3.4 for 语句与 while 语句的比较

for 语句等价于下列语句序列：

```
表达式 1;
while(表达式 2)
{   循环体语句
    表达式 3;
}
```

可以看出，for 语句可以取代 while 语句，而 for 语句结构显得整齐、紧凑、清晰。

5.3.5 for 语句应用举例

例 5.2 用 for 语句，求 s=1+2+3+…+100 的值。

分析：设 s 的初值为 0，循环控制变量 i 从 1 增加到 100，循环体为：

```
s=s+i;                          /* i=1,2,…,100 */
```

参考程序如下：

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

例 5.3 用 for 语句求 n!。

分析：对于 i(1≤i≤n)，i! 可以表示成：i! =i*(i−1)!，如果用变量 fact 存放 i!，则 fact 的初值应为 1。

参考程序如下：

```
#include<stdio.h>
main()
{   int i,n;
    long fact=1;                /* 阶乘的值增加很快,为防止溢出,还可定义为 float 型 */
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        fact=fact*i;
    printf("%d!=%ld\n",n,fact);
}
```

运行该程序，若输入 10↙，则输出结果为：

```
10!=3628800
```

5.3.6 for 语句的变形

1. 表达式的省略

for 语句中的三个表达式，可以根据情况省略其中一个或两个，也可全都省略。当表达式被省略时，其后的分号不可省略。

对于例 5.2，其循环语句可以写成如下几种形式：

(1) 省略表达式 1。

```
i=1;                            /* 在 for 语句之前给循环变量赋初值 */
```

```
for(;i<=100;i++)            /*此处省略了表达式1*/
    s=s+i;
```

(2) 省略表达式3。

```
for(i=1;i<=100;)            /*此处省略了表达式3*/
{   s=s+i;
    i++;                    /*修改循环控制变量*/
}
```

(3) 省略表达式1和表达式3。

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

(4) 将三个表达式全都省略。

如果将三个表达式全都省略,则for语句就没有了循环控制条件,循环将无限进行下去,此时可在循环体中利用break语句来终止循环。例如:

```
i=1;
for(;;)                     /*此处省略了三个表达式*/
{   s=s+i;
    i++;
    if(i>100) break;        /*如果i>100,则退出循环*/
}
```

2. for语句中的逗号表达式

逗号表达式的主要应用就是在for语句中。for语句中的表达式1和表达式3可以是逗号表达式。例如:

```
for(s=0,i=1;i<=100;i++)  /*此处表达式1为逗号表达式*/
    s=s+i;
```

3. 循环体为空语句

对for语句,循环体为空语句的一般形式为:

```
for(表达式1;表达式2;表达式3)
    ;
```

例如:求s=1+2+3+…+100可以用如下循环语句完成:

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

上述for语句的循环体为空语句,不做任何操作。实际上是把求累加和的运算放入表达式3中了。

循环体语句为空语句的情况在while语句和do-while语句中也经常被使用。这是C

语言的一个特点。例如：

```
while(putchar(getchar())!='#');
```

这个循环语句的作用是在显示器上复制输入的字符，当输入的字符为'#'时，结束循环。这里循环体是空语句。

5.4 break 语句、continue 语句和 goto 语句

这三条语句的功能是改变程序的执行顺序，使程序的执行从其所在的位置转向另一处。

5.4.1 break 语句

break 语句的形式为：

```
break;
```

break 语句是限定转向语句，它使流程跳出所在的循环结构，把流程转向所在结构之后。前面已经在 switch 语句中使用过 break 语句，目的是使流程跳出 switch 结构。break 语句在循环结构中的作用是跳出所在的循环结构，转向执行该循环结构后面的语句。例如：

```
#include<stdio.h>
main()
{   int i=1,s=0;
    for(;;)
    {   s=s+i;
        i++;
        if(i>100) break;        /*如果 i>100,则退出循环*/
    }
    printf("s=%d\n",s);
}
```

在本程序执行中，当 i>100 时，强行终止 for 循环（从循环体中跳出），继续执行 for 语句的下一条语句，即输出 s 的值。

例 5.4 求圆的面积。

参考程序如下：

```
#include<stdio.h>
#define PI 3.1415926
main()
{   int r;
    float s;
    for(r=1;r<=10;r++)
    {   s=PI*r*r;
        if(s>100) break;        /*如果 s>100,则退出循环*/
        printf("s=%.2f\n",s);
    }
}
```

本程序在执行时，半径从 1 到 10 变化，对于每一个半径值，求出相应的圆面积值 s。如果 s＞100，则退出循环，否则输出 s。从程序中看出 for 循环有两种结束方式，一是当 r＞10 时，二是当 s＞100 时。

break 语句不能用于循环语句和 switch 语句之外的任何其他语句。如果在多重(层)嵌套的结构中使用 break 语句，则 break 仅仅退出所在的那层结构，即 break 语句不能使程序控制退出一层以上的结构。

5.4.2 continue 语句

continue 语句的形式为：

```
continue;
```

continue 语句被称为继续语句。该语句的功能是使本次循环提前结束，即跳过循环体中 continue 语句后面尚未执行的循环体语句，继续进行下一次循环的条件判断。continue 语句只能出现在循环体语句中，不能用在其他的地方。

例 5.5 显示输入的字符，如果按 Esc 键，则退出循环；如果按 Enter 键，则不做任何处理，继续输入下一个字符。

```
#include<stdio.h>
#include "conio.h"
main()
{   char ch;
    for(;;)
    {   ch=getch();              /*将输入的字符放入 ch 中*/
        if(ch==27) break;        /*Esc 键的 ASCII 码为 27*/
        if(ch==13) continue;     /*Enter 键的 ASCII 码为 13*/
        putch(ch);               /*显示输入的字符*/
    }
    getch();                     /*程序暂停，按任意键继续，目的是查看程序的运行情况*/
}
```

说明：

getch()和 putch()的作用与 getchar()和 putchar()相似。不同的是：

(1) getch()不显示键盘输入的字符。

(2) getchar()输入字符时要按 Enter 键，计算机才会响应，而用 getch()时，输入字符不需要按 Enter 键。

(3) 需要的头文件不同。使用 getch()和 putch()时，所需的头文件是“conio.h”，而使用 getchar()和 putchar()时，所需的头文件是“stdio.h”。

在实际编程中，continue 语句很少使用。实际上，例 5.5 中的循环体语句

```
if(ch==13) continue;
putch(ch);
```

改为

```
if(ch!=13) putch(ch);
```

程序的功能是一样的。

5.4.3 goto 语句

goto 语句被称为无条件转移语句，它的一般形式为：

```
goto 标号;
```

执行 goto 语句使程序流程转移到相应标号所在的语句，并从该语句继续执行。语句标号用标识符表示。带标号语句的形式是：

```
标号:语句
```

即标号和语句之间用冒号隔开。

下面的程序是用 goto 语句来求 s=1+2+3+…+100 的值。

```
#include<stdio.h>
main()
{       int i=1,s=0;
    loop: s=s+i;
       i++;
       if(i<=100)
          goto loop;
       printf("s=%d\n",s);
}
```

goto 语句只能使流程在函数内转移，不得转移到该函数外。当需要从多重嵌套的结构中转移到最外层时，可以使用 goto 语句。

大量使用 goto 语句会打乱各种有效的控制语句，导致程序结构不清晰，程序的可读性变差，再加上 goto 语句可以用别的语句代替，因此要尽量避免使用 goto 语句。

5.5 循环的嵌套

在循环体语句中又包含另一个完整的循环结构的形式，称为循环的嵌套。嵌套在循环体内的循环结构称为内循环，外面的循环结构称为外循环。如果内循环体中又有嵌套的循环语句，则构成多重循环。while、do-while 和 for 三种循环可以互相嵌套。

例 5.6 编写程序输出如下图形。

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

算法分析：

(1) 用循环控制变量 i(1≤i≤5)控制图形的行数：

```
for(i=1;i<=5;i++)
    输出第 i 行;
```

(2) 每行上'*'的个数随着控制变量 i 值的变化而变化。

```
i=1 时,执行 1 次 putchar('*');
i=2 时,执行 2 次 putchar('*');
    ⋮
i=5 时,执行 5 次 putchar('*');
```

如果用循环控制变量 j(1≤j≤i)来控制图形中每行'*'的个数,则内循环体语句应该如下:

```
for(j=1;j<=i;j++)
    putchar('*');
```

完整的程序为:

```
#include<stdio.h>
main()
{   int i,j;
    for(i=1;i<=5;i++)
    {   for(j=1;j<=i;j++)
            putchar('*');    /*或 printf("*");*/
        putchar('\n');       /*或 printf("\n");*/
    }
}
```

本例中是两重 for 循环嵌套。其实三种循环语句可以互相嵌套。例如:

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

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

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

循环嵌套的程序中,要求内循环必须被包含在外层循环的循环体中,不允许出现内外层循环体交叉的情况。例如:

```
do
    {   …
```

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

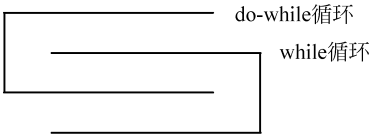

图 5-4　循环交叉为非法结构

在 do-while 循环体内开始 while 循环，但是 do-while 循环结束在 while 循环体内，它们相互交叉，这是非法的结构，如图 5-4 所示。

5.6　程 序 举 例

例 5.7　输出 1～20 中能被 3 整除的数，并求出它们的和。

```
#include<stdio.h>
main()
{   int n,s=0;
    for(n=1;n<=20;n++)
        if(n%3==0)
        {   printf("%4d",n);
            s=s+n;
        }
    printf("\ns=%d\n",s);
}
```

例 5.8　求 3～100 间的全部素数。

算法分析：

(1) 素数是只能被 1 和本身整除的自然数(1 除外)。例如 2,3,5,7 是素数。1,4,6,8,10 不是素数。

(2) 判断某数 i 是否为素数的简单办法是：用 2,3,4,…,i－1 这些数据逐个去除 i，只要被其中的一个数整除了，则 i 就不是素数。数学上已证明，对于自然数 i 只需用 2,3,4,…,$i^{1/2}$ 测试，即从 2 开始到 i 的平方根的值即可。

程序如下：

```
#include<stdio.h>
#include "math.h"
main()
{   int i,m,n=0,k;
    for(m=3;m<=100;m+=2)          /* m+=2 是将其中的偶数直接跳过 */
    {   k=sqrt(m);
        for(i=2;i<=k;i++)
            if(m%i==0) break;     /* m 不是素数,结束内循环 */
        if(i>k)                   /* 区分内循环是正常结束的还是执行 break 结束的 */
        {   printf("%5d",m);
            n++;
            if(n%10==0) printf("\n");/* 控制每行输出 10 个数据 */
        }
    }
    printf("\n");
}
```

例 5.9 用 π/4≈1－1/3＋1/5－1/7＋…公式求得近似值，直到最后一项的绝对值小于 10^{-6} 为止。

参考程序如下：

```
#include<stdio.h>
#include "math.h"
main()
{   int s;
    float n,t,pi;
    t=1;pi= 0;n=1.0;s=1;
    while(fabs(t)>=1e-6)
    {   pi=pi+t;
        n=n+2;
        s=-s;
        t=s/n;
    }
    pi=pi*4;
    printf("pi=%10.6f\n",pi);
}
```

运行结果为：

```
pi=3.141594
```

例 5.10 求 Fibonacci 数列：1，1，2，3，5，8，…前 20 个数。

算法分析：

$f_1=1$ (n=1)

$f_2=1$ (n=2)

$f_n=f_{n-1}+f_{n-2}$ (n>=3)

参考程序如下：

```
#include<stdio.h>
main()
{   int f1,f2,i;
    f1=1;f2=1;
    for(i=1;i<=10;i++)
    {   printf("%8d%8d",f1,f2);          /*一次输出数列的两项值*/
        if(i%2==0) printf("\n");         /*控制每行输出4项值*/
        f1=f1+f2;                        /*产生下一项值*/
        f2=f2+f1;                        /*产生再下一项值*/
    }
}
```

例 5.11 打印出“九九乘法表”。

```
*    1    2    3    4    5    6    7    8    9
1    1
2    2    4
3    3    6    9
```

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

分析：观察输出的结果可以发现，第 1 行输出的是表头，它确定了各列输出的位置，可用一个循环语句实现 1～9 列中列号的输出。从第 2 行开始输出表的内容，即各行的行号 i 和 i 个数的乘积，可用二重循环实现，外循环控制输出 1～9 行，内循环控制输出第 i 行的 1～i 个乘积。输出时要注意格式控制，让各列对齐。

参考程序如下：

```
#include<stdio.h>
main()
{
  int i,j;
  printf("%c",'*');                          /*输出第 1 行的乘号*/
  for(i=1;i<=9;i++)
   printf("%4d",i);                          /*输出第 1 行的被乘数*/
  printf("\n");
  for(i=1;i<=9;i++)                          /*外循环,控制输出 9 行*/
  {
    printf("%d",i);                          /*输出第 i 行的乘数*/
    for(j=1;j<=i;j++)                        /*内循环,控制输出第 i 行的 i 列乘积*/
      printf("%4d",i*j);
    printf("\n");
  }
}
```

例 5.12　用穷举法解决搬砖问题：36 块砖，36 人搬，男搬 4 块，女搬 3 块，两个小孩抬 1 砖，要求一次搬完，问需要男、女、小孩各多少人？

穷举法的基本思想是对问题的所有可能的状态一一测试，直到找到解或将全部的可能状态都测试过为止。

对于搬砖问题，可设 3 个变量 m、w 和 c，分别表示男人、女人和小孩的人数，根据题意可列出以下两个方程：

$$\begin{cases} m+w+c=36 & ① \\ 4\times m+3\times w+c/2=36 & ② \end{cases}$$

这个问题可列两个方程，但是却有 3 个变量，是不能通过解方程的方法求解的，但是由于 3 个变量 m、w 和 c 都应该是整型变量，可以将各种可能的取值一一列举出来进行测试，从而找到这个问题的解。具体方法是：

首先，确定各变量的取值范围，依题意可知：

$$\begin{cases} 1\leqslant m<9 \\ 1\leqslant w<12 \end{cases}$$

由于当变量 m 和 w 的值一旦确定，即可根据方程①确定变量 c 的值为 c=36－m－w。所以不用确定变量 c 的取值范围。

然后，将变量 m、w、c 的各种可能的取值代入方程②中，若等式成立，则这时变量 m、w 和 c 的值即是该问题的解。

参考程序如下：

```
#include<stdio.h>
main()
{
int m,w,c;
  printf("%10s%10s%10s\n","men","women","children");
  for(m=1;m<9;m++)
    for(w=1;w<12;w++)
     {
      c=36-m-w;
      if(m*4+w*3+c/2.0==36) /*或写成 if((c%2==0)&&(m*4+w*3+c/2==36))*/
        printf("%10d%10d%10d\n",m,w,c);
     }
}
```

运行结果为：

```
men       women      children
  3           3            30
```

若将程序中的语句“if(m*4+w*3+c/2.0==36) printf(“%10d%10d%10d\\n”,m,w,c);”中的 c/2.0 写成 c/2，则运行结果为：

```
men       women      children
  1           6            29
  3           3            30
```

显然这时的第一个结果并不符合题意，这是由于在 C 语言中 c/2.0 和 c/2 是有区别的，而在数学上它们是没有区别的。

例 5.13 译密码。为了使电文保密，往往按一定规律将其转换成密码，收报人再按约定的规律将其译回原文。已知电文加密规律为：将字母变成其后面的第 4 个字母，其他字符保持不变。例如 A→E，B－F，a→e 等，将“china!”转换为“glmre!”。

```
#include<stdio.h>
main()
{   char c;
    while((c=getchar())!='\n')
    {   if((c>='a'&&c<='z')||(c>='A'&&c<='Z'))
        {   c=c+4;
            if(c>'Z'&&c<='Z'+4||c>'z')
                c=c-26;
        }
        printf("%c",c);
    }
}
```

运行结果为：

china!↙
glmre!

本章小结

1. 本章主要介绍了 3 种构成循环的语句：while、do…while 和 for 语句，读者首先应熟练掌握语句的一般形式和语句的执行过程。对于用 if 和 goto 语句构成的循环结构，只要做一般性的了解即可。

2. 本章还介绍了循环结构中常用到的两个语句：break 语句和 continue 语句，读者要注意两者的区别：break 语句是结束整个循环，continue 语句只是结束本次循环。

3. 在介绍循环程序设计方法的同时，本章还介绍了循环嵌套的概念，读者应重点掌握用 for 语句构成的二重循环的应用，对于多重循环只要求做一般性的了解。

习　题

一、选择题

1. 在以下给出的表达式中，与 while(E)中的(E)不等价的表达式是(　　)。

A) (!E==0)　　B) (E>0||E<0)　　C) (E==0)　　D) (E!=0)

2. 要求通过 while 循环不断读入字符，当读入字母 N 时结束循环。若变量已正确定义，以下程序段正确的是(　　)。

A) while((ch=getchar())!='N')printf("%c",ch);

B) while(ch=getchar()!='N')printf("%c",ch);

C) while(ch=getchar()=='N')printf("%c",ch);

D) while((ch=getchar()=='N')printf("%d",ch);

3. 设变量已正确定义，则以下能正确计算 f=n! 的程序段是(　　)。

A)
```
f=0;
for(i=1; i<=n; i++)f*=i;
```

B)
```
f=1;
for(i=1; i<n; i++) f*=i;
```

C)
```
f=1;
for(i=n; i>1; i++} f*=i;
```

D)
```
f=1;
for(i=n; i>=2; i--) f*=i;
```

4. 有以下程序段：

```
#include<stdio.h>
main()
{int i=0,s=0;
  for(;;)
   {
    if(i==3||i==5)continue;
    if(i==6)break;
    i++;
```

```
    s+=i;
  };
printf("%d\n",s);
}
```

程序运行后的输出结果是(　　)。

A) 10　　B) 13

C) 21　　D) 程序进入死循环

5. 有以下程序：

```
#include<stdio.h>
main()
{
  int x=0,y=5,z=3;
  while(z-->0&&++x<5)  y=y-1;
  printf("%d,%d,%d\n",x,y,z);
}
```

程序执行后的输出结果是(　　)。

A) 3,2,0　　B) 3,2,-1　　C) 4,3,-1　　D) 5,-2,-5

6. 有以下程序：

```
#include<stdio.h>
main()
{ int n,t=1,s=0;
  scanf("%d",&n);
  do{s=s+t;t=t-2;}
  while(t!=n);
}
```

为使此程序不陷入死循环,从键盘输入的数据应该是(　　)。

A) 任意正奇数　　B) 任意负偶数　　C) 任意正偶数　　D) 任意负奇数

7. 若变量已正确定义,要求程序段完成求5!的计算,不能完成此操作的程序段是(　　)。

A) for(i=1,p=1; i<=5; i++)p*=i;

B) for(i=1; i<=5; i++){p=l; P*=i; }

C) i=l; p=1; while(i<=5){p*=i; i++; }

D) i=1; p=1; do{p*=i; i++; }while(i<=5);

8. 有以下程序：

```
#include<stdio.h>
main()
{
  int i,n=0;
  for(i=2;i<5;i++)
  {
    do{if(i%3) continue;
    n++;
```

```
    }while(!i);
  n++;
  }
  printf("n=%d\n",n);
}
```

程序执行后的输出结果是(　　)。

A）n=5　　B）n=2　　C）n=3　　D）n=4

二、填空题

1. 以下程序的功能是：输出 100 以内(不含 100)能被 3 整除且个位数为 6 的所有整数，请填空。

```
#include<stdio.h>
main()
{
  int i,j;
  for(i=0;_____;i++)
  {j=i*10+6;
  if(_____)continue;
  printf("%d ",j);
  }
}
```

2. 以下程序的功能是计算：s=1+12+123+1234+12345，请填空。

```
#include<stdio.h>
main()
{
  int t=0,s=0,i;
  for(i=1;i<=5;i++)
  {t=i+_____;s=s+t;}
  printf("s=%d\n",s);
}
```

三、编程题

1. 求 s=1+2+4+8+…+64 的值。

2. 求 s=1+1/2+1/3+…+1/100 的值。

3. 求 T=1!+2!+3!+…+10! 的值。

4. 编程实现 100 依次减去 1,2,3,…,x，直到其结果第一次变负时，输出相应的 x 值。

5. 打印出所有的“水仙花数”。所谓“水仙花数”是指一个 3 位数，其中各位数字的立方和等于该数本身。例如 $153=1^3+5^3+3^3$。

6. 以下面的格式，输出九九乘法表。

```
1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
   ⋮
1*9=9 2*9=18 3*9=27 … 9*9=81
```

7. 打印如下图形。

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

8. 求出前100个素数(第一个素数是2)。

9. 输入10个整数,统计出其中正数、负数和零的个数。

10. 输入一串整数,直到输入为0结束。求出其中正数之和及负数之和。

11. 输入10个学生的成绩,统计出其中成绩在60分以下、60～90分以及90分以上的学生人数。