第5章 学会编写循环程序

循环(或称重复)是计算机解题的一个强项。计算机运算速度快,最擅长做重复性的工作。人们正是利用了计算机的这一特长,把那些复杂的不易理解的求解过程转换为易于理解的多次重复操作。这样一方面可以降低问题的复杂性,降低程序设计的难度,减少程序书写及输入的工作量;另一方面可以充分发挥计算机运算速度快、能自动执行程序的优势。

循环控制有两种办法:计数法和标志法。如果知道要循环的次数,就可以用计数法,首先要确定循环次数,每次循环都要让循环次数变量变化(可以是递增,也可以是递减),然后和要完成的循环次数比较,一旦完成测试次数后就立即结束循环;如果不知道要循环的次数,就可以考虑用标志法,这是一种"有多少算多少"的办法。具体方法如下:测试前先设置一个标志变量,并将标志变量赋值为"没有测试完"(可用"0"代表);然后每测试一次,检查一下标志变量的值有无变化。标志变量只有"没有测试完"和"测试完"两个值,当某次测试找到需要的解或测试完最后一个对象后,就让标志变量变成"测试完"(可用"1"代表),计算机检测到这个标志变量不为0,就立刻跳出循环结构。

5.1 循环语句和循环控制

【例5.1】 求1+2+…+100=?

分析:被加的数有规律地递增,累加和s等于之前的累加和加上当前的被加数。

以下是被加数i和累加和s的变化情况。

	被加数 i	累加和 s
初始	i=0	s=0
计算	i=i+1=0+1=1	s=s+i=0+1=1
	i=i+1=1+1=2	s=s+i=1+2=3
	i=i+1=2+1=3	s=s+i=3+3=6
	i=i+1=3+1=4	s=s+i=6+4=10
	⋮	
	i=i+1=99+1=100	s=s+i=4950+100=5050
	i=100结束	

可以看到,i=i+1;s=s+i;重复出现,可采用循环实现。

循环结构包括 for、while、do…while 语句，goto 无条件转向语句和 if 配对也可构成循环语句。

5.1.1 用 if 语句和 goto 语句构成的循环

上一章学习了 if 语句，goto 语句，顾名思义就是跳转的意思，既然是跳转，肯定要有跳转的目的地，所以 goto 语句后面一定会跟一个要跳转的语句标号，但这个语句标号只是指明了要去的地方，肯定在要去的地方有一个语句标号(门牌号码)。

使用语句标号需要特别注意以下 3 点。

(1) 不能用整数做语句标号。

(2) 语句标号只能出现在 goto 所在函数内，且唯一。

(3) 语句标号只能加在可执行语句前面。

【例 5.2】 用 if 和 goto 语句构成循环，求 $\sum_{n=1}^{100} n$。

```
#include<stdio.h>
#define uchar unsigned char
#define uint unsigned int
main()
{
    uchar i=1;
    uint sum=0;
loop: if(i<=100)          //loop 为要跳转的目的地
    {
        sum=sum+i;
        i++;
        goto loop;        //执行跳转指令
    }
    printf("%d\n",sum);
}
```

由于 goto 语句会造成无序跳转，所以不推荐使用这种循环控制方式，而建议使用下面介绍的 3 种循环语句。

5.1.2 用 while 语句构成的循环

while 语句的一般格式如下：

```
while(表达式)
    循环体语句
```

while 语句的循环流程图如图 5.1 所示。

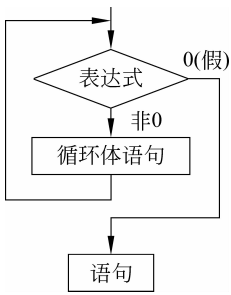

图 5.1 while 语句的循环流程图

如图 5.1 所示。while 语句首先计算条件表达式，如

果表达式的值为真，执行循环体语句；然后系统会自动回来再计算条件表达式，如果表达式的值仍然为真，再执行循环体语句；直到表达式的值为假，结束 while 语句，执行 while 循环体语句后面的语句。while 循环可能有以下几种特殊情况：

(1) 当第一次计算表达式且其值为 0 时，循环体里面的语句会一次也不执行。

(2) 当表达式的值恒为非 0 时，会出现无限循环，比如 while(1)，单片机的 C 语言 main()函数中就需要 while(1)来让系统只要供电就无限运行下去。

(3) 虽然表达式的值为非 0，即满足继续循环的条件，但可以通过 if 语句以及 goto、break、continue 语句跳出循环(后两种语句稍后介绍)。

使用 while 语句需要特别注意以下 3 点，这也是初学 C 语言最容易出问题的地方。

(1) 循环体里面如果包含一个以上语句且每条语句用分号结束时，应当用花括号将这些语句括起来，以复合语句的形式出现。若不括，则 while 语句的范围只到 while 后面的第一个分号处。如果不用花括号将循环体内的多条语句括起来，可以用逗号连接多条语句。请看例 5.3。

(2) 循环体中应有使循环趋向于结束的语句，否则会造成死循环。

(3) 千万不要在紧跟 while 的圆括号后面加“;”(除非是用于空循环)，加上“;”号后，即使满足循环条件也不会执行循环体语句。

【例 5.3】 分析以下程序的运行结果。

```
#include<stdio.h>
main()
{
    int x=0,y=0;
    while(x<15) y++,x+=++y;
    printf("%d,%d\n",y,x);
}
```

程序分析：分析循环程序最好画一个表格，如表 5.1 所示，可以得出程序的运行结果为 8,20。从该程序可以证实：用逗号可以将几个表达式连接起来，作为一个复合语句。

表 5.1 对例 5.3 程序的分析结果

循 环 次 数	y	x
循环之前	0	0
第一次	2	2
第二次	4	6
第三次	6	12
第四次	8	20

【例 5.4】 用 while 语句构成循环，求 $\sum_{n=1}^{100} n$。

```
#include<stdio.h>
#define uchar unsigned char
#define unint unsigned int
```

```
main( )
{
    uchar i=0;
    uint sum=0;
     while(i<100)
    {                  //循环体内多于一条语句,要把这几条语句用“{ }”括起来!
        sum=sum+i;
        i++;           //使循环趋于结束的语句,一定不能少!
    }
    printf(“1+2+…+100=%d\n”, sum);
}
```

【例 5.5】 显示 1～10 的平方。

```
#include<stdio.h>
#define uchar unsigned char
#define unint unsigned int
main( )
{
    uchar i=0;
    uint sum=0;
    while(i<10)
    {                  //循环体内多于一条语句,要把这几条语句用“{ }”括起来!
         printf("%d * %d=%d\n",i,i,i * i);
         i++;          //使循环趋于结束的语句,一定不能少!
    }
}
```

有时需要先试探一下,再决定是否继续循环。比如一个人竞聘某个生产线上的工作,工厂的负责人安排他先试着干一干,如果干得了就每天重复干下去,如果干不了就不招聘此人。do…while 语句就编写这类循环。

5.1.3 用 do…while 语句构成的循环

do…while 语句的一般格式如下:

```
do{循环体语句}
while(表达式);
```

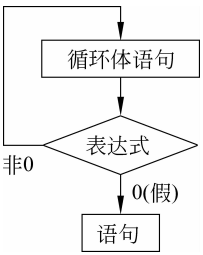

图 5.2 do…while 语句的循环流程图

do…while 语句的循环流程图如图 5.2 所示。

do…while 语句的执行过程如下:先执行循环体中的语句,然后判断条件表达式,条件成立再执行循环体;重复上述过程,直到条件不成立时结束循环。使用 while 语句时需要特别注意以下 3 点,

这也是初学 C 语言的人最容易出问题的地方。

(1) 循环体如果包含一个以上语句时,应当用花括号将这些语句括起来,以复合语句的形式出现。若不括,则仅重复执行 do 与 while 之间的一条语句。

(2) 在循环体中,应有使循环趋向于结束的语句,比如“i++;”。

(3) while(条件)后面的分号不要省。

【例 5.6】 用 do…while 语句构成循环,求 $\sum_{n=1}^{100} n$。

```
#include<stdio.h>
#define uchar unsigned char
#define unint unsigned int
main( )
{
    uchar i=1;
    uint sum=0;
    do
    {                  //循环体内多于一条语句,要把这几条语句用“{ }”括起来!
        sum=sum+i;
        i++;           //使循环趋于结束的语句,一定不能少!
    }while(i<100);
    printf(“1+2+…+100=%d\n”,sum);
}
```

在一般情况下,用 while 语句和 do…while 语句处理同一问题时,若二者的循环体部分是一样的,它们的结果也一样。但如果 while 后面的表达式一开始就为假(0 值)时,两种循环的结果是不同的。请看下面两个程序的对比。

```
main()
{
    int i,sum=0;
    scanf("%d",&i);
    do
    {
        sum+=i;
        i++;
    }while(i< =10);
    printf("%d",sum);
}
```

```
main()
{
    int i,sum=0;
    scanf("%d",&i);
    while(i<=10)
    {
        sum+=i;
        i++;
    }
    printf("%d",sum);
}
```

当 while 后面表达式第一次的值为“真”时,两种循环得到的结果相同。否则,二者结果不相同(指二者具有相同循环体的情况)。

5.1.4 用 for 语句构成的循环

for 语句的一般格式如下:

```
for([expr1] ;[expr2] ;[expr3]) 循环体语句;
for([循环变量赋初值] ;[循环条件] ;[使循环趋于结束的表达式])
    循环体语句;
```

for 语句的循环流程图如图 5.3 所示。

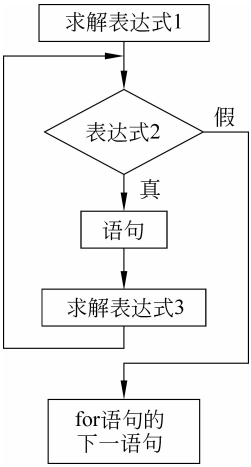

图 5.3　for 语句的循环流程图

从图 5.3 可以看出，for 循环的表达式 1 仅执行 1 次，表达式 1 的任务一般是做一些循环前的准备工作；表达式 2 和表达式 3 会多次执行，表达式 2 是判断循环是否继续的语句，一般是条件语句；表达式 3 是使循环趋于结束的语句。

for 语句的执行过程如下：for 语句首先执行表达式 1，即初始化循环，然后执行表达式 2，即执行条件表达式，如果表达式 2 的值为真，则执行 for 的循环体语句，然后执行表达式 3，表达式 3 执行后，系统会自动回来再执行表达式 2，如果表达式 2 的值仍然为真，再执行 for 的循环体语句，然后执行表达式 3，然后再执行表达式 2，如果表达式 2 的值为假时，结束 for 循环，执行 for 语句后面的下一语句。

使用 for 循环应该注意以下几点。

(1) 除了空循环之外，千万不要在紧跟 for 的圆括号后面加分号“;”。一些初学者把 for()与 printf()等函数等同看待，觉得 printf()后面要加“;”，在 for()的后面也给加了“;”，结果造成不能实现循环工作的目的。要避免犯这个错误，切记 for、while、do…while、if 等语句不是函数！它们就像说话一样，仅仅是说了半句话，就拿 for 语句来说，仅仅是说“对于…”，后面还有下文！再比如 if()，仅仅是说“如果…”，假如在 if()后面加了分号“;”，那就没了下文。假如一个人说了句“对于你们这些人来说…”，没有再说别的，会不会觉得这个人有问题？

(2) 在 for 循环中，如果语句多于一条，就一定要用花括号“{}”括起来，否则 for 循环只会多次执行 for()后面的第一条语句，其余语句只有在循环结束之后才被执行。要避免犯此错误，输入完 for()后面的圆括号后，马上回车，然后输入一对花括号(哪怕是循环体仅仅有 1 条语句也加上花括号)，然后将光标调到花括号内，输入循环体语句。

(3) for 循环中的初始化、条件表达式和增量都是选择项，可以缺省，但分号不能缺省。省略了初始化，表示不对循环控制变量赋初值；省略了条件表达式，则不做其他处理时便成为死循环；省略了使循环趋于结束的表达式 3 语句，可在循环体语句中加入表达式 3，如果不加，也会造成死循环。

(4) for 循环本质上也是当型循环结构，但比 while 循环更简洁，因为 for 循环的圆括号内不但可以写循环初始化语句、判断循环是否继续的语句、使循环趋于结束的语句，而且还可以通过逗号表达式将循环体语句写进去。

【例 5.7】 找出 1～500 之间满足除以 3 余 2、除以 5 余 3、除以 7 余 2 的所有整数。

编程思路：求余可通过运算符“%”来实现，假设循环变量为 i，对于题目要求，可以用下列条件来判断。

```
(i%3==2)&&(i%5==3)&&(i%7==2)
#include<stdio.h>
#define uint unsigned int
main()
{
    uint i,n=0;
    for(i=1;i<=500;i++)
    {
        if((i%3==2)&&(i%5==3)&&(i%7==2))
        printf("%6d",i);
    }
    printf("\n");
}
```

运行结果如图 5.4 所示。

图 5.4　运行结果

5.2　学会循环嵌套编程

有时一个循环里又嵌套着循环。以上几种循环都可以自己嵌套自己，还可以嵌套别的循环类型。

如 for 循环不但可以再嵌套 for 循环，也可以和 while 循环以及 do…while 循环相互嵌套。

5.2.1　while 循环嵌套 while 循环

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

【例 5.8】 用 while 嵌套 while 循环编写九九乘法表。

```
#include<stdio.h>
#define uchar unsigned char
main()
```

```
{
    uchar i=1,j;
    while(i<=9)
    {
        j=1;
        while(j<=9)
        {
            printf("%d*%d=%2d\t",i,j,i*j);
            j++;
        }

        printf("\n");
        i++;
    }
}
```

5.2.2 do…while 循环嵌套 do…while 循环

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

【例 5.9】 用 do…while 嵌套 do…while 循环编写九九乘法表。

```
#include<stdio.h>
#define u8 unsigned char
main()
{
    u8 i=1,j;
    do
    {
        j=1;
        do
        {
            printf("%d*%d=%2d\t",i,j,i*j);
            j++;
        }while(j<=9);

        printf("\n");
```

```
        i++;
    }while(i<=9);
}
```

5.2.3 for 循环嵌套 for 循环

```
for(; ;)
{
    for(; ;)
    {...}
}
```

请读者用 for 循环嵌套 for 循环编写九九乘法表。(建议将此题作为上机编程考试题之一)

【例 5.10】 中国古代数学史上著名的“百鸡问题”：鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问翁、母、雏鸡各几何？

编程思路：设公鸡、母鸡、小鸡的数目各为 i、j、k，根据题意有如下公式。

$$i+j+k=100$$

$$5i+3j+\frac{1}{3}k=100$$

另外可知，100 元钱如果全部都买公鸡，最多可买 20 只；如果全部都买母鸡，可买 33 只；如果全部都买小鸡可买 300 只，于是可以写出以下条件。

$$(i+j+k)==100\&\&(5*i+3*j+k/3)==100$$

由于 k/3 可能会造成小数，所以还应该加上一个条件：k%3==0，按照以上思路编出的程序如下：

```
#include<stdio.h>
#define uchar unsigned char
main()
{
    uchar i,j;
    int k;
    for(i=0;i<=20;i++)
    {
        for(j=0;j<=33;j++)
        {
            for(k=0;k<=300;k++)
                if((i+j+k)==100&&(5*i+3*j+k/3)==100&&(k%3==0))
                    printf("%5d%5d%5d\n",i,j,k);
        }
    }
}
```

但是，采用这种算法编写出来的程序总共循环了 20 * 33 * 200＝198 000 次，运算时间很长，仔细分析一下，其实并不需要三重循环，由于鸡的总数是 100，当 i 和 j 确定后，k 可以通过 100－i－j 来求，于是可以将程序改为如下内容。

```
#include<stdio.h>
#define uchar unsigned char
main()
{
    uchar i,j;
    int k;
    for(i=0;i<=20;i++)
    {
        for(j=0;j<=33;j++)
        {
            k=100-i-j;
                if((5*i+3*j+k/3)==100&&(k%3==0))
                    printf("%5d%5d%5d\n",i,j,k);
        }
    }
}
```

这样一来，程序运行时间大大缩短，运行结果，如图 5.5 所示。

图 5.5　运行结果

5.2.4　while 循环嵌套 do…while 循环

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

    }
```