6 循环控制

1.1 概述

循环结构是程序中一种很重要的结构。其特点是，在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。C语言提供了多种循环语句，可以组成各种不同形式的循环结构。

1) 用 goto 语句和 if 语句构成循环；
2) 用 while 语句；
3) 用 do-while 语句；
4) 用 for 语句；

1.1 goto 语句以及用 goto 语句构成循环

goto 语句是一种无条件转移语句，与 BASIC 中的 goto 语句相似。goto 语句的使用格式为：

```
goto　语句标号；
```

其中标号是一个有效的标识符，这个标识符加上一个":"一起出现在函数内某处，执行 goto 语句后，程序将跳转到该标号处并执行其后的语句。另外标号必须与 goto 语句同处于一个函数中，但可以不在一个循环层中。通常 goto 语句与 if 条件语句连用，当满足某一条件时，程序跳到标号处运行。

goto 语句通常不用，主要因为它将使程序层次不清，且不易读，但在多层嵌套退出时，用 goto 语句则比较合理。

【例 6.1】用 goto 语句和 if 语句构成循环，$\sum_{n=1}^{100} n$。

```
main()
{
```

```
        int i,sum=0;
        i=1;
loop:   if(i<=100)
          {sum=sum+i;
           i++;
           goto loop;}
         printf("%d\n",sum);
}
```

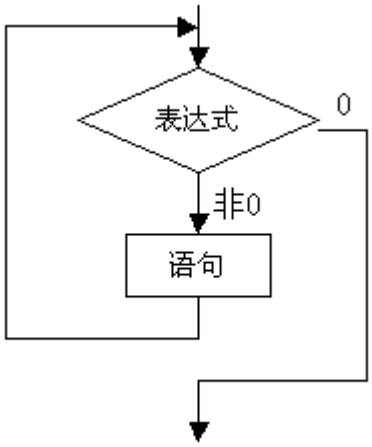
1.1 while 语句

while 语句的一般形式为：

while(表达式)语句

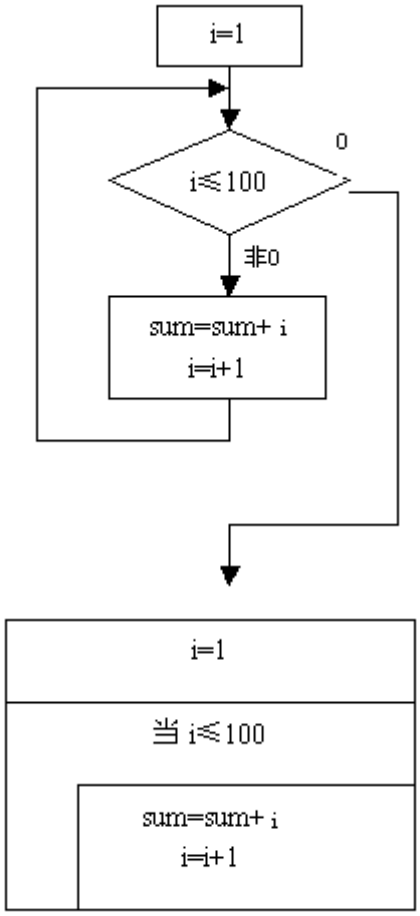
其中表达式是循环条件，语句为循环体。

while 语句的语义是：计算表达式的值，当值为真(非 0)时， 执行循环体语句。其执行过程可用下图表示。



【例 6.2】用 while 语句求 $\sum_{n=1}^{100} n$ 。

用传统流程图和 N-S 结构流程图表示算法，见图：




```
main()
{
   int i,sum=0;
   i=1;
   while(i<=100)
        {
          sum=sum+i;
          i++;
         }
   printf("%d\n",sum);
}
```

【例 6.3】统计从键盘输入一行字符的个数。

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

本例程序中的循环条件为 getchar()!='\n',其意义是， 只要从键盘输入的字符不是回车就继续循环。循环体 n++完成对输入字符个数计数。从而程序实现了对输入一行字符的字符个数计数。

使用 while 语句应注意以下几点：

1) while 语句中的表达式一般是关系表达或逻辑表达式，只要表达式的值为真(非 0)即可继续循环。

【例 6.4】

```
main(){
    int a=0,n;
    printf("\n input n:    ");
    scanf("%d",&n);
    while (n--)
      printf("%d  ",a++*2);
}
```

本例程序将执行 n 次循环，每执行一次，n 值减 1。循环体输出表达式 a++*2 的值。该表达式等效于(a*2; a++)。

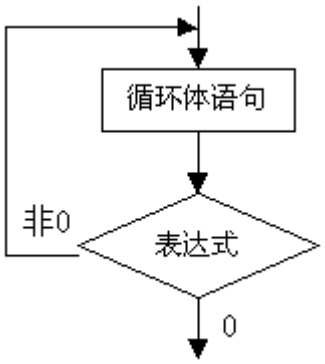
2) 循环体如包括有一个以上的语句，则必须用{}括起来，组成复合语句。

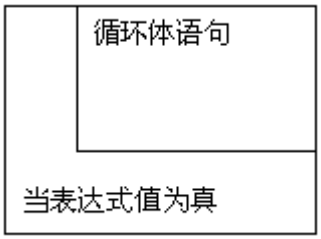
1.1 do-while 语句

do-while 语句的一般形式为:

```
do
    语句
while(表达式);
```

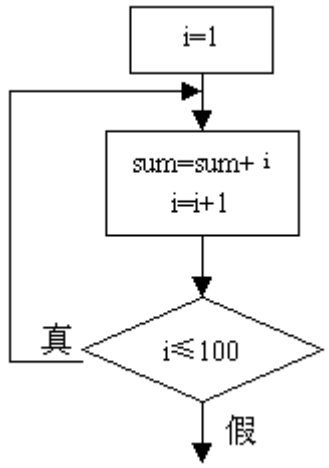
这个循环与 while 循环的不同在于:它先执行循环中的语句,然后再判断表达式是否为真，如果为真则继续循环；如果为假，则终止循环。因此，do-while 循环至少要执行一次循环语句。其执行过程可用下图表示。

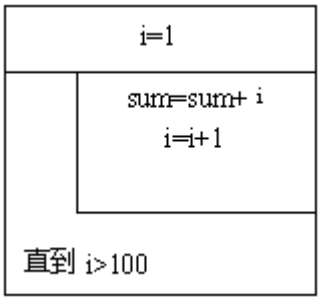




【例 6.5】用 do-while 语句求 $\sum_{n=1}^{100} n$ 。

用传统流程图和 N-S 结构流程图表示算法，见图：






```
main()
{
   int i,sum=0;
   i=1;
   do
        {
          sum=sum+i;
          i++;
         }
    while(i<=100)
    printf("%d\n",sum);
}
```

同样当有许多语句参加循环时，要用"{"和"}"把它们括起来。

【例 6.6】while 和 do-while 循环比较。

（1）main()

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

（2）main()

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

1.1 for 语句

在 C 语言中，for 语句使用最为灵活，它完全可以取代 while 语句。它的一般形式为：

for(表达式 1；表达式 2；表达式 3）语句

它的执行过程如下：

1）　先求解表达式 1。

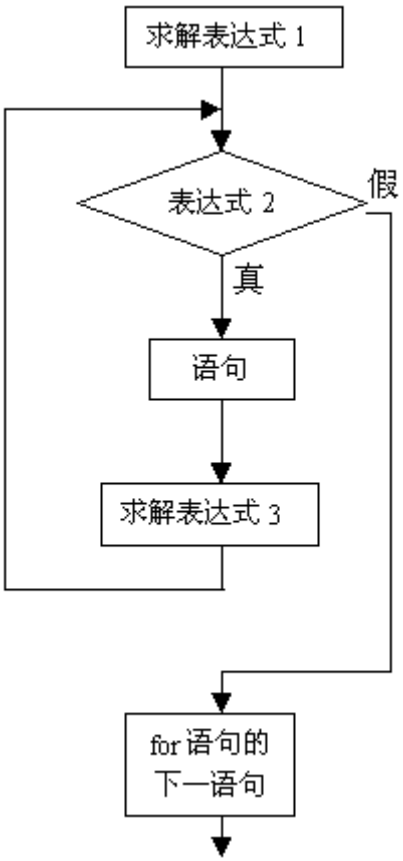
2）　求解表达式 2，若其值为真（非 0），则执行 for 语句中指定的内嵌语句，然后执行下面第 3）步；若其值为假（0），则结束循环，转到第 5）步。

3）　求解表达式 3。

4）　转回上面第 2）步继续执行。

5）　循环结束，执行 for 语句下面的一个语句。

其执行过程可用下图表示。



for 语句最简单的应用形式也是最容易理解的形式如下：

for(循环变量赋初值；循环条件；循环变量增量) 语句

循环变量赋初值总是一个赋值语句，它用来给循环控制变量赋初值；循环条件是一个关系表达式,它决定什么时候退出循环；循环变量增量，定义循环控制变量每循环一次后按什么方式变化。这三个部分之间用"；"分开。

例如:

```
for(i=1; i<=100; i++)sum=sum+i;
```

先给 i 赋初值 1,判断 i 是否小于等于 100，若是则执行语句,之后值增加 1。再重新判断,直到条件为假,即 i>100 时,结束循环。

相当于：

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

对于 for 循环中语句的一般形式，就是如下的 while 循环形式：

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

注意:

1) for 循环中的"表达式 1（循环变量赋初值）"、"表达式 2(循环条件)"和"表达式 3(循

环变量增量)"都是选择项，即可以缺省,但";"不能缺省。

2) 省略了“表达式 1（循环变量赋初值）”，表示不对循环控制变量赋初值。

3) 省略了“表达式 2(循环条件)”，则不做其它处理时便成为死循环。

例如：

```
for(i=1;;i++)sum=sum+i;
```

相当于：

```
i=1;
while(1)
    {sum=sum+i;
     i++;}
```

4) 省略了“表达式 3(循环变量增量)”，则不对循环控制变量进行操作,这时可在语句体中加入修改循环控制变量的语句。

例如：

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

5) 省略了“表达式 1（循环变量赋初值）”和"表达式 3(循环变量增量)"。

例如：

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

相当于：

```
while(i<=100)
    {sum=sum+i;
     i++;}
```

6) 3 个表达式都可以省略。

例如：

```
for(;; )语句
```

相当于：

```
while(1)语句
```

7) 表达式 1 可以是设置循环变量的初值的赋值表达式，也可以是其他表达式。

例如：

```
for(sum=0;i<=100;i++)sum=sum+i;
```

8) 表达式 1 和表达式 3 可以是一个简单表达式也可以是逗号表达式。

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

或：

```
for(i=0,j=100;i<=100;i++,j--)k=i+j;
```

9) 表达式 2 一般是关系表达式或逻辑表达式，但也可是数值表达式或字符表达式，只要其值非零，就执行循环体。

例如：

```
for(i=0;(c=getchar())!='\n';i+=c);
```

又如：

```
for(;(c=getchar())!='\n';)
    printf("%c",c);
```

1.1 循环的嵌套

【例 6.7】

```
main()
{
 int i, j, k;
 printf("i j k\n");
 for (i=0; i<2; i++)
    for(j=0; j<2; j++)
       for(k=0; k<2; k++)
           printf("%d %d %d\n", i, j, k);
 }
```

1.1 几种循环的比较

1) 四种循环都可以用来处理同一个问题，一般可以互相代替。但一般不提倡用 goto 型循环。
2) while 和 do-while 循环，循环体中应包括使循环趋于结束的语句。for 语句功能最强。
3) 用 while 和 do-while 循环时，循环变量初始化的操作应在 while 和 do-while 语句之前完成，而 for 语句可以在表达式 1 中实现循环变量的初始化。

1.1 break 和 continue 语句

1.1.1 break 语句

break 语句通常用在循环语句和开关语句中。当 break 用于开关语句 switch 中时,可使程序跳出 switch 而执行 switch 以后的语句；如果没有 break 语句,则将成为一个死循环而无法退出。break 在 switch 中的用法已在前面介绍开关语句时的例子中碰到,这里不再举例。

当 break 语句用于 do-while、for、while 循环语句中时,可使程序终止循环而执行循环后面的语句, 通常 break 语句总是与 if 语句联在一起。即满足条件时便跳出循环。

【例 6.8】

```
main()
{
   int i=0;
   char c;
   while(1)                    /*设置循环*/
     {
      c='\0';                 /*变量赋初值*/
```

```
    while(c!=13&&c!=27) /*键盘接收字符直到按回车或 Esc 键*/
      {
       c=getch();
       printf("%c\n", c);
      }
    if(c==27)
       break;          /*判断若按 Esc 键则退出循环*/
    i++;
    printf("The No. is %d\n", i);
    }
  printf("The end");
}
```

注意:

1) break 语句对 if-else 的条件语句不起作用。
2) 在多层循环中, 一个 break 语句只向外跳一层。

1.1.1 continue 语句

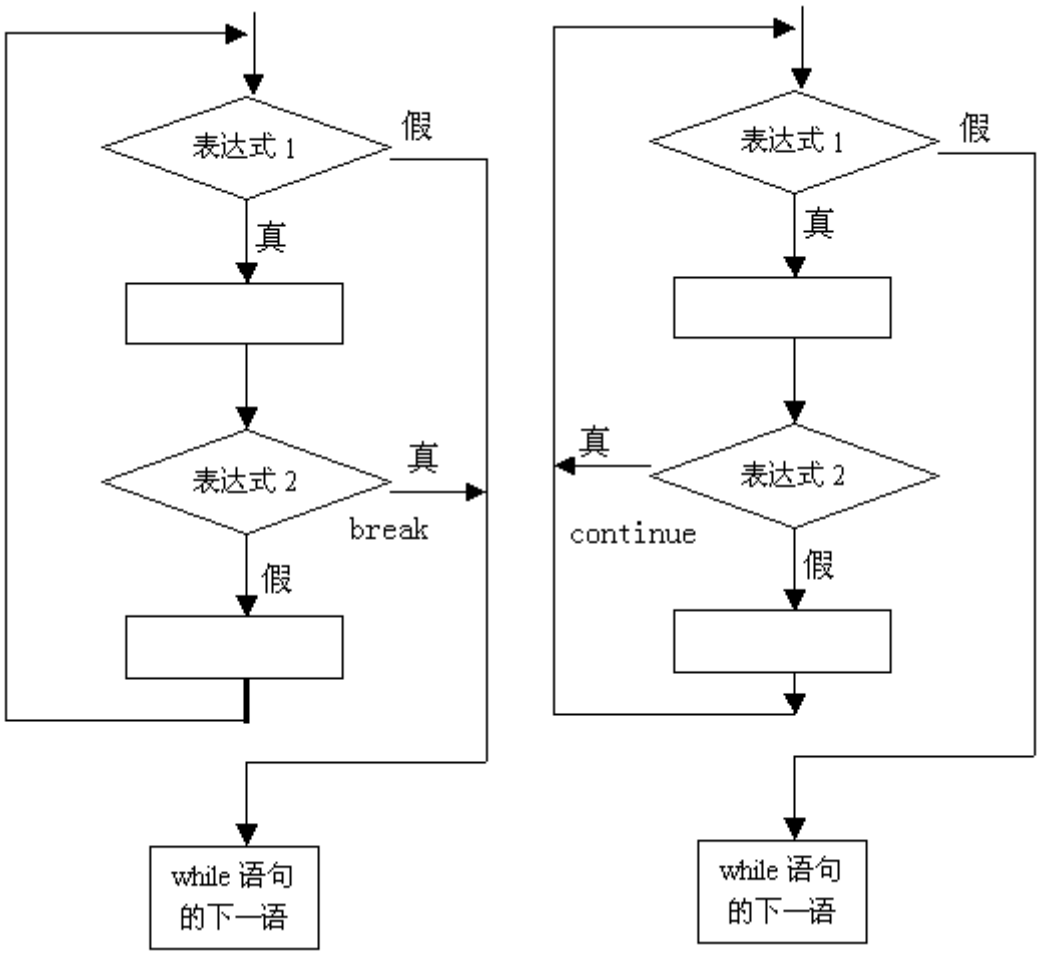
continue 语句的作用是跳过循环本中剩余的语句而强行执行下一次循环。continue 语句只用在 for、while、do-while 等循环体中,常与 if 条件语句一起使用,用来加速循环。其执行过程可用下图表示。

1) while(表达式 1)

```
    { ……
      if(表达式 2)break;
      ……
     }
```

2) while(表达式 1)

```
    { ……
      if(表达式 2)continue;
      ……
     }
```



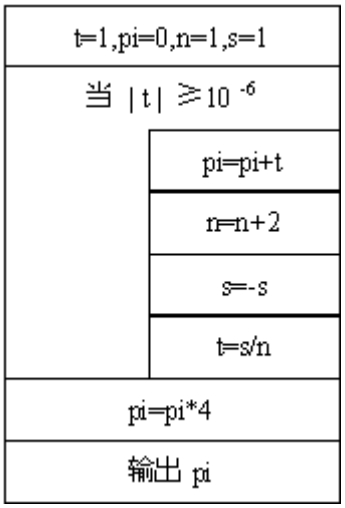
【例 6.9】

```
main()
{
  char c;
  while(c!=13)       /*不是回车符则循环*/
      {
         c=getch();
         if(c==0X1B)
            continue; /*若按 Esc 键不输出便进行下次循环*/
         printf("%c\n", c);
       }
}
```

1.1 程序举例

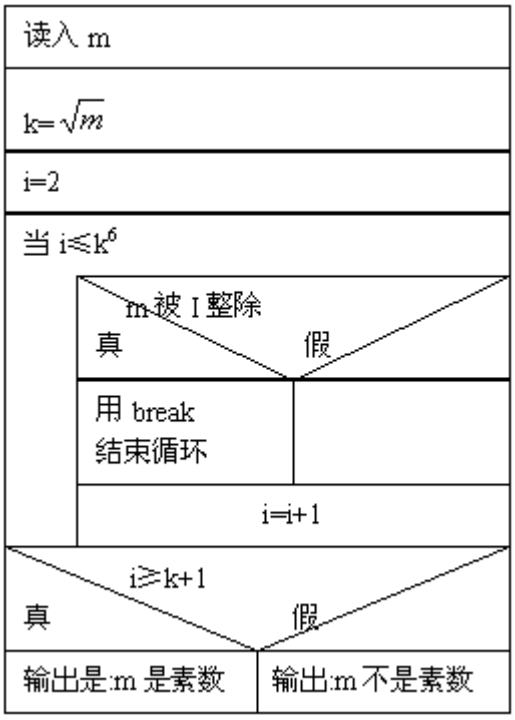
【例 6.10】用 $\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots$ 公式求 π 。

N-S 流程图：




```
#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);
  }
```

【例 6.11】判断 m 是否素数。



N-S 流程图：

```
#include<math.h>
main()
{
int m,i,k;
scanf("%d",&m);
k=sqrt(m);
for(i=2;i<=k;i++)
  if(m%i==0)break;
  if(i>=k+1)
  printf("%d is a prime number\n",m);
  else
  printf("%d is not a prime number\n",m);
}
```

【例 6.12】求 100 至 200 间的全部素数。

```
#include<math.h>
main()
{
  int m,i,k,n=0;
  for(m=101;m<=200;m=m+2)
     {
       k=sqrt(m);
       for(i=2;i<=k;i++)
```

```
        if(m%i==0)break;
        if(i>=k+1)
            {printf("%d",m);
             n=n+1;}
        if(n%n==0)printf("\n");
    }
printf("\n");
}
```