

第 5 章

循环结构程序设计

为什么需要循环控制

- 要向计算机输入全班50个学生的成绩;
(重复50次相同的输入操作)
- 分别统计全班50个学生的平均成绩;
(重复50次相同的计算操作)

解决方法

```
scanf(" %f,%f,%f,%f,%f" ,&score1,&score2,&score3,&score4,&score5);  
//输入一个学生5门课的成绩  
aver=(score1+score2+score3+score4+score5)/5;  
//求该学生平均成绩  
printf(" aver=%7.2f" ,aver);  
//输出该学生平均成绩
```

重复写49个同样的程序段



```
i=1;                //设整型变量i初值为1  
while( i<=50 )      //当i的值小于或等于50时执行花括号内的语句  
{  
    scanf("%f,%f,%f,%f,%f",&score1,&score2,&score3,&score4,&score5);  
    aver=(score1+score2+score3+score4+score5)/5;  
    printf("aver=%7.2f",aver);  
    i++;            //每执行完一次循环使i的值加1  
}
```

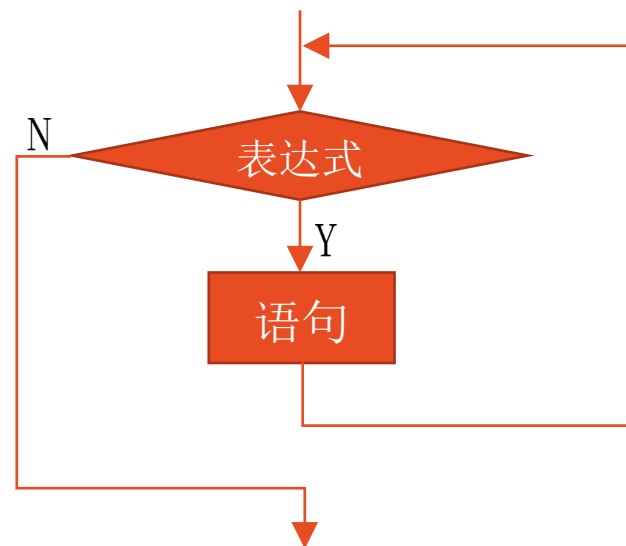
用while语句实现循环

while(表达式) 语句

while语句可简单地记为：只要当循环条件表达式为真(即给定的条件成立)，就执行**循环体语句**。

“语句”就是循环体。循环体可以是一个简单的语句，可以是**复合语句**(用花括号括起来的若干语句{ })。

执行循环体的次数是由循环条件控制的，这个循环条件就是上面一般形式中的“**表达式**”，它也称为循环条件表达式。当此表达式的值为“**真**” (以非0值表示)时，就执行循环体语句；为“**假**” (以0表示)时，就不执行循环体语句。



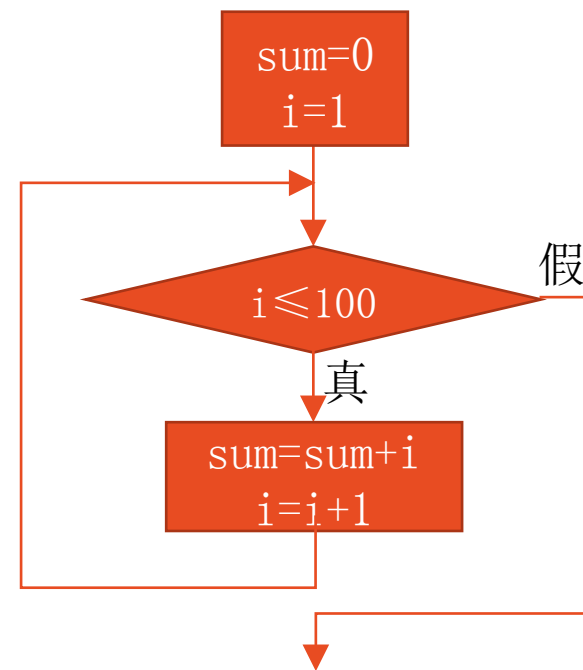
注意

while循环的特点是
先判断条件表达式，
后执行循环体语句。

while语句实现循环

【例5.1】求 $1+2+3+\dots+100$ ，即 $\sum_{n=1}^{100} n$

```
#include<stdio.h>
int main()
{
    int i=1, sum=0;           //定义变量i的初值为1, sum的初值为0
    while(i <= 100)           //当i>100, 条件表达式i<=100的值为假, 不执行循环体
    {                           //循环体开始
        sum=sum+i;             //第1次累加后, sum的值为1
        i++;                   //加完后, i的值加1, 为下次累加做准备
    }                           //循环体结束
    printf("sum=%d\n", sum);   //输出1+2+3...+100的累加和
    return 0;
}
```

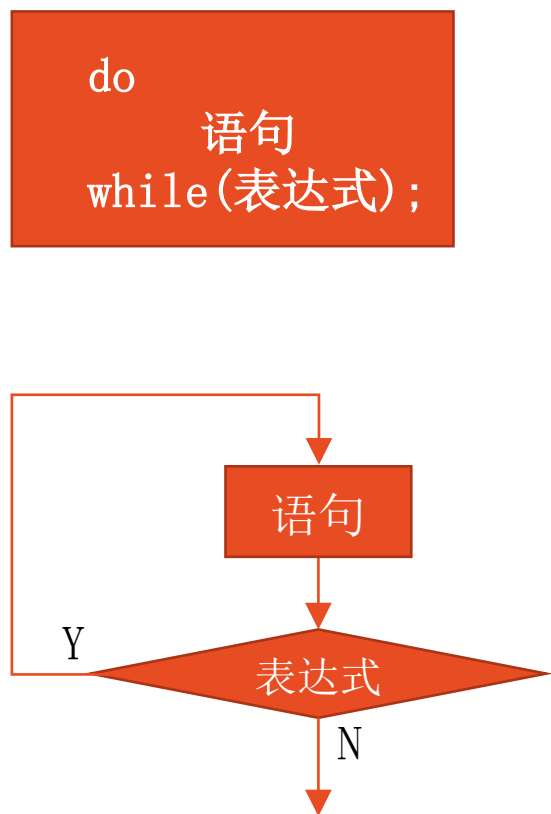


The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The output displayed is:
sum=5050
请按任意键继续. . .



- (1) 循环体如果包含一个以上的语句，应该用**花括号**括起来，作为复合语句出现。
- (2) 不要忽略给*i*和sum**赋初值**，否则它们的值是不可预测的，结果显然不正确。
- (3) 在循环体中应有使循环趋向于结束的语句。如本例中的“`i++;`”语句。如果无此语句，则*i*的值始终不改变，循环永远不结束。

用do...while语句实现循环



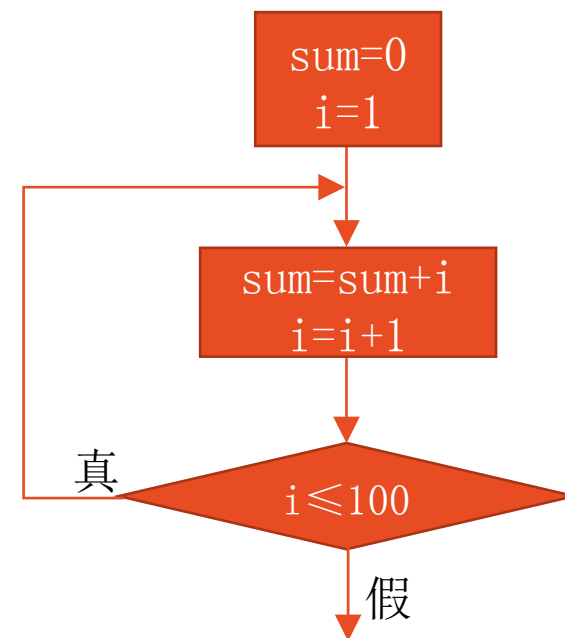
注意

do...while语句的特点是，先无条件地执行循环体，然后判断循环条件是否成立。

用do...while语句实现循环

【例5.2】用do...while语句求 $1+2+3+\dots+100$ ，即 $\sum_{n=1}^{100} n$

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



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the output of the program: `sum=5050` followed by the prompt `请按任意键继续. . .` (Press any key to continue. . .).

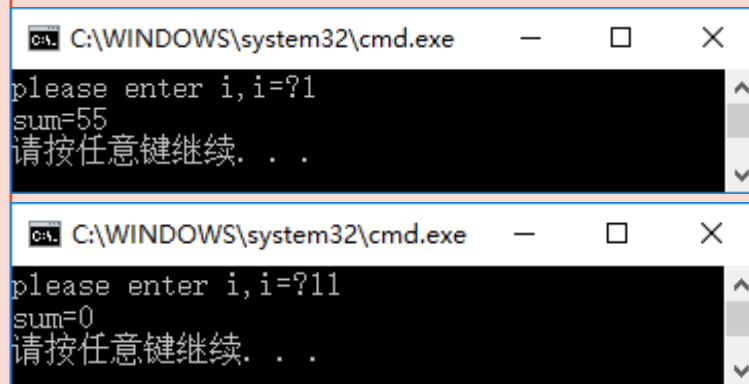


在一般情况下，用while语句和用do...while语句处理同一问题时，若二者的循环体部分是一样的，那么结果也一样。
但是如果while后面的表达式一开始就为假(0值)时，两种循环的结果是不同的。

用do...while语句实现循环

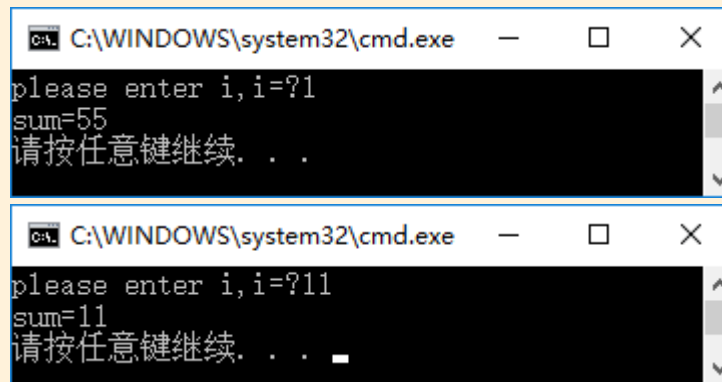
【例5.3】 while和do...while循环的比较。

```
#include <stdio.h>
int main()
{
    int i, sum=0;
    printf("please enter i, i=?");
    scanf("%d", &i);
    while(i <= 10)
    {
        sum=sum+i;
        i++;
    }
    printf("sum=%d\n", sum);
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
please enter i, i=?1
sum=55
请按任意键继续. . .

C:\WINDOWS\system32\cmd.exe
please enter i, i=?11
sum=0
请按任意键继续. . .
```



```
C:\WINDOWS\system32\cmd.exe
please enter i, i=?1
sum=55
请按任意键继续. . .

C:\WINDOWS\system32\cmd.exe
please enter i, i=?11
sum=11
请按任意键继续. . .
```

```
#include <stdio.h>
int main()
{
    int i, sum=0;
    printf("please enter i, i=?");
    scanf("%d", &i);
    do
    {
        sum=sum+i;
        i++;
    }while(i <= 10);
    printf("sum=%d\n", sum);
    return 0;
}
```

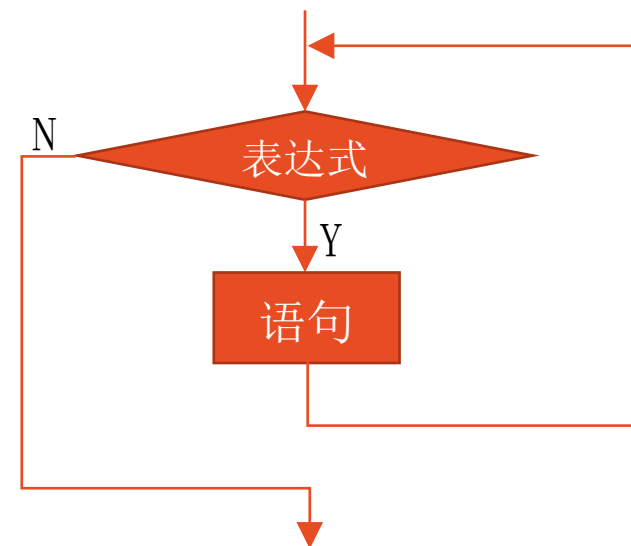
用for语句实现循环

for(表达式1; 表达式2; 表达式3)
语句

表达式1：设置初始条件，只执行一次。可以为零个、一个或多个变量设置初值。

表达式2：是循环条件表达式，用来判定是否继续循环。在每次执行循环体前先执行此表达式，决定是否继续执行循环。

表达式3：作为循环的调整，例如使循环变量增值，它是在执行完循环体后才进行的。



注意

for与while循环的特点是先判断条件表达式，后执行循环体语句。

用for语句实现循环

for(表达式1; 表达式2; 表达式3)
语句



for(循环变量赋值; 表达式2; 表达式3)
语句

for语句更为灵活，不仅可以用于循环次数已经确定的情况，还可以用于循环次数不确定而只给出循环结束条件的情况，它完全可以代替while语句。

表达式1：设置初始条件，只执行一次。可以为零个、一个或多个变量设置初值。

表达式2：是循环条件表达式，用来判定是否继续循环。在每次执行循环体前先执行此表达式，决定是否继续执行循环。

表达式3：作为循环的调整，例如使循环变量增值，它是在执行完循环体后才进行的。

用for语句实现循环

```
for(表达式1; 表达式2; 表达式3)  
    语句
```

≡

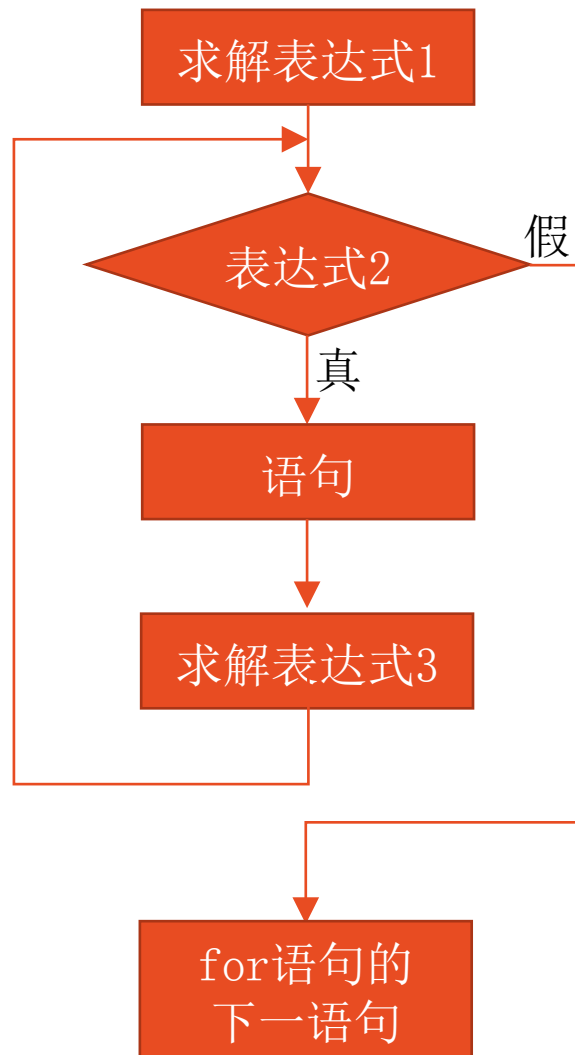
```
表达式1;  
while 表达式2  
{  
    语句  
    表达式3  
}
```

for语句的执行过程如下：

- (1) 求解表达式1。
- (2) 求解表达式2，若此条件表达式的值为真(非0)，则执行for语句中的循环体，然后执行第(3)步。若为假(0)，则结束循环，转到第(5)步。
- (3) 求解表达式3。
- (4) 转回步骤(2)继续执行。

注意：在执行完循环体后，循环变量的值“超过”循环终值，循环结束。

- (5) 循环结束，执行for语句下面的一个语句。



用for语句实现循环

for(表达式1; 表达式2; 表达式3)
语句

```
int i = 0;
```

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

```
for(i = 0; ;i++) { ...; if(表达式 ) break;... }
```

```
for(i = 0; i <= 100;) { ...; i++; ... }
```

注意

- “表达式1”可以省略，即不设置初值，但表达式1后的分号不能省略。例如：for(; i<=100; i++)。应当注意：由于省略了表达式1，没有对循环变量赋初值，因此，为了能正常执行循环，应在for语句之前给循环变量赋以初值。
- 表达式2也可以省略，即不用表达式2来作为循环条件表达式，不设置和检查循环的条件。此时循环无终止地进行下去，也就是认为表达式2始终为真。
- 表达式3也可以省略，但此时程序设计者应另外设法保证循环能正常结束。
- 甚至可以将3个表达式都可省略，即不设初值，不判断条件(认为表达式2为真值)，循环变量也不增值，无终止地执行循环体语句，显然这是没有实用价值的。
- 表达式1可以是设置循环变量初值的赋值表达式，也可以是与循环变量无关的其他表达式。表达式3也可以是与循环控制无关的任意表达式。但不论怎样写for语句，都必须使循环能正常执行。
- 表达式1和表达式3可以是一个简单的表达式，也可以是逗号表达式，即包含一个以上的简单表达式，中间用逗号间隔。
- 表达式2一般是关系表达式或逻辑表达式，但也可以是数值表达式或字符表达式，只要其值为非零，就执行循环体。
- for语句的循环体可为空语句，把本来要在循环体内处理的内容放在表达式3中，作用是一样的。可见for语句功能强，可以在表达式中完成本来应在循环体内完成的操作。
- C99允许在for语句的“表达式1”中定义变量并赋初值。

循环的嵌套

01

```
while()  
{  
    :  
    while()  
    {...}  
}
```

} 内层
循环

02

```
do  
{  
    :  
    do  
    {...}  
    while();  
}while();
```

} 内层
循环

03

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

} 内层
循环

04

```
while()  
{  
    :  
    do  
    {...}  
    while();  
    :  
}
```

} 内层
循环

05

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

} 内层
循环

06

```
do  
{  
    :  
    for( ; ; )  
    {...}  
}while();
```

} 内层
循环



几种循环的比较

(1) 3种循环都可以用来处理同一问题，一般情况下它们可以互相代替。

(2) 在while循环和do...while循环中，只在while后面的括号内指定循环条件，因此为了使循环能正常结束，应在循环体中包含使循环趋于结束的语句(如i++，或i=i+1等)。

for循环可以在表达式3中包含使循环趋于结束的操作，甚至可以将循环体中的操作全部放到表达式3中。因此for语句的功能更强，凡用while循环能完成的，用for循环都能实现。

(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。

(4) while循环、do...while循环和for循环都可以用break语句跳出循环，用continue语句结束本次循环。

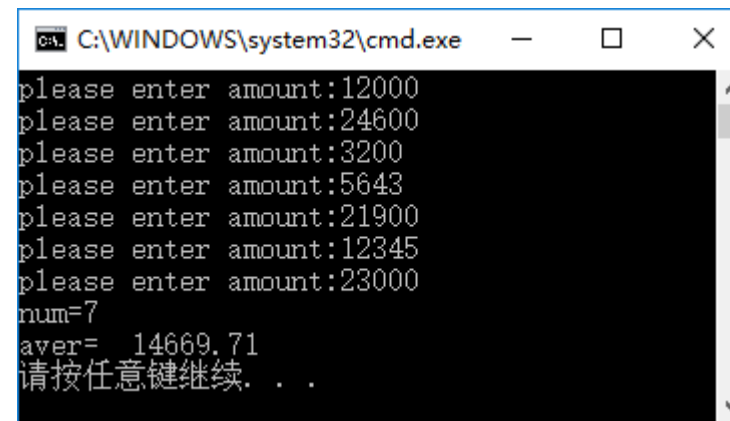




用break语句提前终止循环

【例5.4】在全系1000名学生中举行慈善募捐，当总数达到10万元时就结束，统计此时捐款的人数以及平均每人捐款的数目。

```
#include <stdio.h>
#define SUM 100000 //指定符号常量SUM代表10万
int main()
{
    float amount, aver, total;
    int i;
    for (i=1, total=0; i<=1000; i++)
    {
        printf("please enter amount:");
        scanf("%f", &amount);
        total = total + amount;
        if (total >= SUM) break;
    }
    aver=total/i;
    printf("num=%d\naver=%10.2f\n", i, aver);
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
please enter amount:12000
please enter amount:24600
please enter amount:3200
please enter amount:5643
please enter amount:21900
please enter amount:12345
please enter amount:23000
num=7
aver= 14669.71
请按任意键继续. . .
```



for语句指定执行循环体1000次。每次循环中，输入一个捐款人的捐款数，并累加到total中。设置了if语句，在每一次累加捐款数amount后，立即检查累加和total是否达到或超过SUM(即100 000)，若超过就执行break语句，流程跳转到循环体的花括号外，提前结束循环。

用break语句提前终止循环

```
break;
```

作用：使流程跳到循环体之外，接着执行循环体下面的语句。

注意：break语句只能用于循环语句和switch语句之中，而不能单独使用。

用continue语句提前结束本次循环

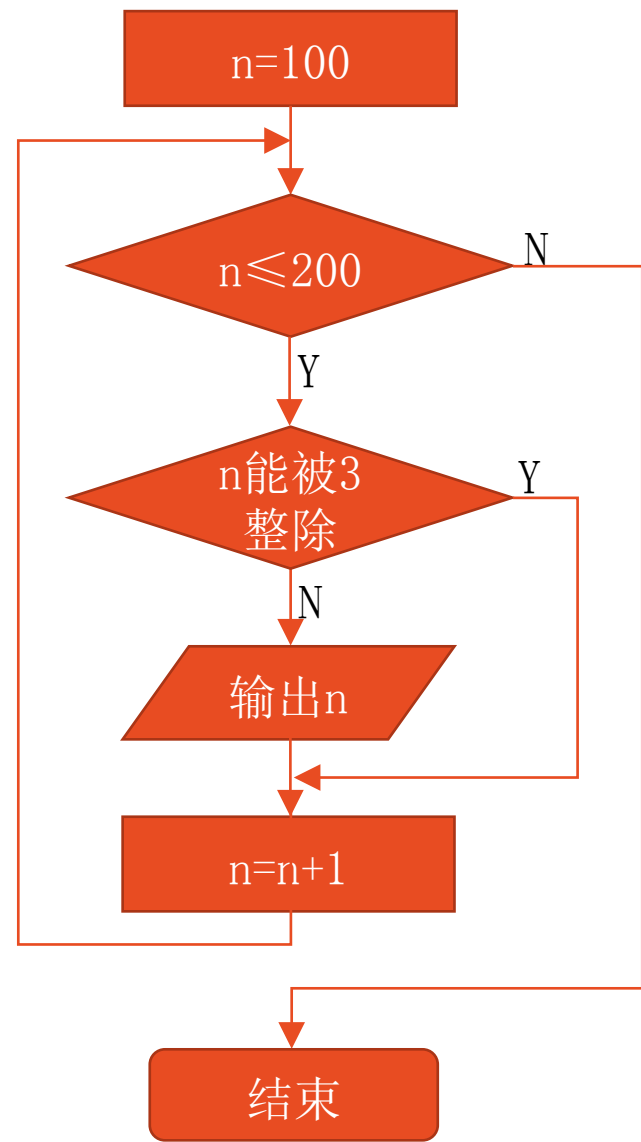
【例5.5】要求输出100 ~ 200之间的不能被3整除的数。

```
#include <stdio.h>
int main()
{   int n;
    for (n = 100; n <= 200; n++)
    {   if (n%3==0)
        continue;
        printf("%d ", n);
    }
    printf("\n");
    return 0;
}
```



当n能被3整除时，执行continue语句，流程跳转到表示循环体结束的右花括号的前面(注意不是右花括号的后面)，从而跳过printf函数语句，结束本次循环，然后进行循环变量的增值(n++)，只要n<=200，就会接着执行下一次循环。

```
C:\WINDOWS\system32\cmd.exe
100 101 103 104 106 107 109 110 112 113 115 116 118 119 121 122
124 125 127 128 130 131 133 134 136 137 139 140 142 143 145 146
148 149 151 152 154 155 157 158 160 161 163 164 166 167 169 170
172 173 175 176 178 179 181 182 184 185 187 188 190 191 193 194
196 197 199 200
请按任意键继续. . .
```



用continue语句提前结束本次循环

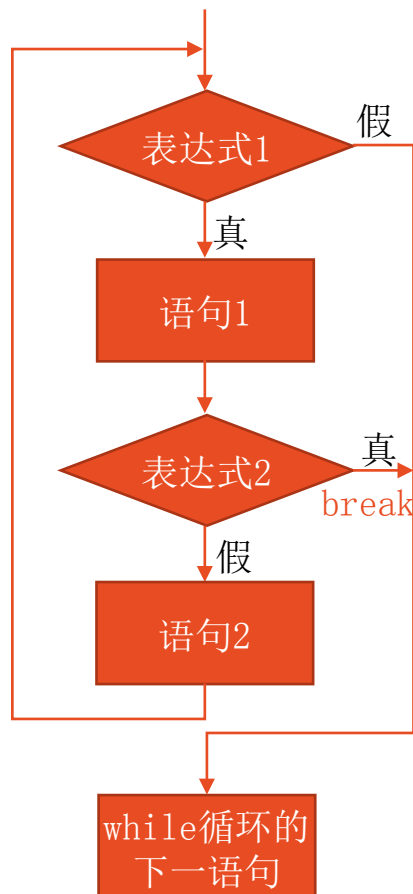
```
continue;
```

作用：结束本次循环，即跳过循环体中下面尚未执行的语句，转到循环体结束点之前，接着执行for语句中的“表达式3”，然后进行下一次是否执行循环的判定。

break语句和continue语句的区别

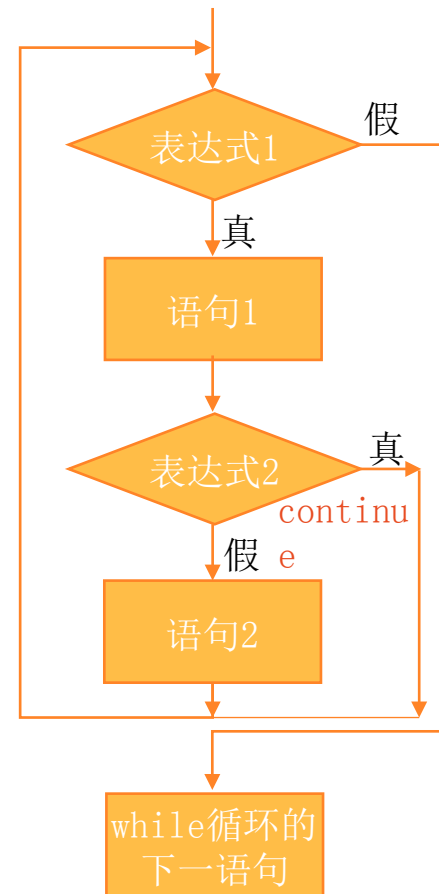
break;

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



continue;

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



`continue`语句只结束本次循环，而非终止整个循环。`break`语句结束整个循环，不再判断执行循环的条件是否成立。

break语句和continue语句的区别

【例5.6】输出以下4×5的矩阵。

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20

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

```
int main()
```

```
{
```

```
    int i, j, n=0;
```

```
    for(i=1; i<=4; i++)
```

```
    {    for(j=1; j<=5; j++, n++)
```

```
        {    if(n%5==0) printf("\n"); //n用来累计输出数据的个数
```

```
            printf("%d\t", i*j);
```

```
        }
```

```
    printf("\n");
```

```
    return 0;
```

```
}
```



本程序包括一个双重循环，是for循环的嵌套。外循环变量i由1变到4，用来控制输出4行数据；内循环变量j由1变到5，用来控制输出每行中的5个数据。

if (i==3 && j==1) break;

```
C:\WINDOWS\system32\cmd.exe

1      2      3      4      5
2      4      6      8      10
4      8      12     16     20
请按任意键继续. . .
```

if (i==3 && j==1) continue;

```
C:\WINDOWS\system32\cmd.exe

1      2      3      4      5
2      4      6      8      10
6      9      12     15
4      8      12     16     20
请按任意键继续. . .
```

```
C:\WINDOWS\system32\cmd.exe

1      2      3      4      5
2      4      6      8      10
3      6      9      12     15
4      8      12     16     20
请按任意键继续. . .
```

循环程序举例

【例5.7】用公式 $\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$ 求 π 的近似值，直到发现某一项的绝对值小于 10^{-6} 为止(该项不累加)。

解题思路：找规律：

- (1) 每项的分子都是1。
 - (2) 后一项的分母是前一项的分母加2。
 - (3) 第1项的符号为正，从第2项起，每一项的符号与前一项的符号相反。
- 在每求出一项后，检查它的绝对值是否大于或等于 10^{-6} 。

```
#include <stdio.h>
#include <math.h>           //程序中用到数学函数fabs，应包含头文件math.h
int main()
{   int sign=1;              //sign用来表示数值的符号
    double pi=0.0,n=1.0,term=1.0; //pi开始代表多项式的值，最后代表 $\pi$ 的值，n代表分母，term代表
    当前项的值
    while(fabs(term)>=1e-6)   //检查当前项term的绝对值是否大于或等于 $10^{-6}$ 
    {   pi=pi+term;           //把当前项term累加到pi中
        n=n+2;               //n+2是下一项的分母
        sign=-sign;          //sign代表符号，下一项的符号与上一项符号相反
        term=sign/n;         //求出下一项的值term
    }
    pi=pi*4;                 //多项式的和pi乘以4，才是 $\pi$ 的近似值
    printf("pi=%10.8f\n",pi); //输出 $\pi$ 的近似值
    return 0;
} // 对于double类型的输出，“%f”和“%lf”是一致的，但是scanf中必须使用%lf
```

sign=1, pi=0, n=1, term=1

当 |term| $\geq 10^{-6}$

pi=pi+term

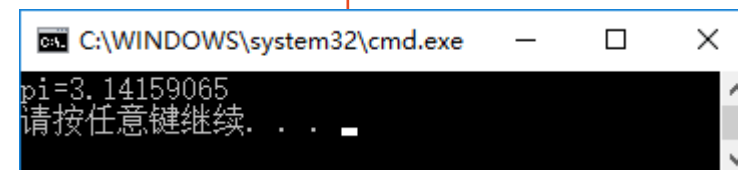
n=n+2

sign=-sign

term=sign/n

pi=pi*4

输出pi



```
C:\WINDOWS\system32\cmd.exe
pi=3.14159065
请按任意键继续. . .
```

循环程序举例

【例5.8】求Fibonacci(斐波那契)数列的前40个数。这个数列有如下特点: 第1, 2两个数为1, 1。从第3个数开始, 该数是其前面两个数之和。即该数列为1,1,2,3,5,8,13,...,用数学方式表示为:

$$\begin{cases} F_1 = 1 & (n = 1) \\ F_2 = 1 & (n = 2) \\ F_n = F_{n-1} + F_{n-2} & (n \geq 3) \end{cases}$$

这是一个有趣的古典数学问题：有一对兔子，从出生后第3个月起每个月都生一对兔子。小兔子长到第3个月后每个月又生一对兔子。假设所有兔子都不死，问每个月的兔子总数为多少？

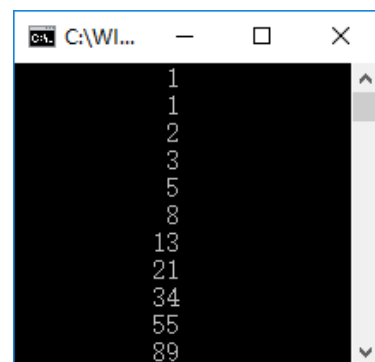
兔子繁殖的规律	月数	小兔子对数	中兔子对数	老兔子对数	兔子总对数
	1	1	0	0	1
	2	0	1	0	1
	3	1	0	1	2
	4	1	1	1	3
	5	2	1	2	5
	6	3	2	3	8
	7	5	3	5	13
	⋮	⋮	⋮	⋮	⋮

注：假设不满1个月的为小兔子，满1个月不满2个月的为中兔子，满2个月以上的为老兔子。

循环程序举例

【例5.8】求Fibonacci(斐波那契)数列的前40个数。

f1=1, f2=1
输出f1, f2
for i=1 to 38
f3=f1+f2
输出f3
f1=f2
f2=f3



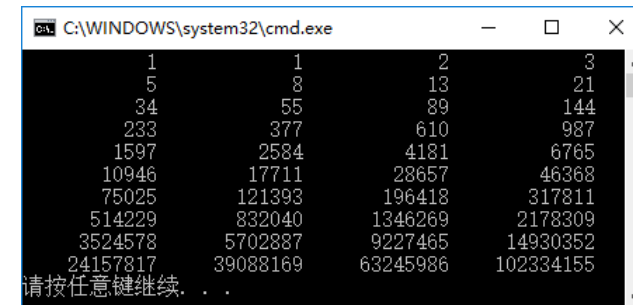
```
C:\WI... 1
1
2
3
5
8
13
21
34
55
89
```

⋮

```
#include <stdio.h>
int main()
{
    int f1=1, f2=1, f3;
    int i;
    printf("%12d\n%12d\n", f1, f2);
    for(i=1; i<=38; i++)
    {
        f3=f1+f2;
        printf("%12d\n", f3);
        f1=f2;
        f2=f3;
    }
    return 0;
}
```



f1=1, f2=1
for i=1 to 20
输出f1, f2
f1=f1+f2
f2=f2+f1



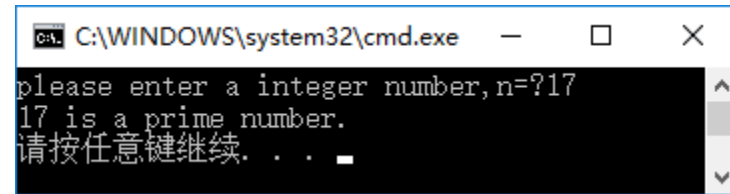
```
C:\WINDOWS\system32\cmd.exe
1      1      2      3
5      8      13     21
34     55     89     144
233    377    610    987
1597   2584   4181   6765
10946  17711  28657  46368
75025  121393 196418 317811
514229 832040 1346269 2178309
3524578 5702887 9227465 14930352
24157817 39088169 63245986 102334155
请按任意键继续. . .
```

```
#include <stdio.h>
int main()
{
    int f1=1, f2=1;
    int i;
    for(i=1; i<=20; i++) //每个循环输出2个月的数据，故只需循环20次
    {
        printf("%12d %12d ", f1, f2); //输出已知的两个月的兔子数
        if(i%2==0) printf("\n");
        f1=f1+f2; //计算出下一个月的兔子数，并存放在f1中
        f2=f2+f1; //计算出下两个月的兔子数，并存放在f2中
    }
    return 0;
}
```

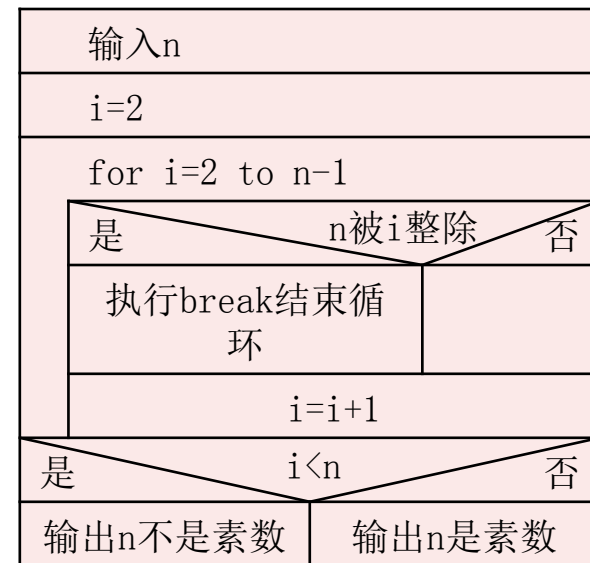
循环程序举例

【例5.9】输入一个大于3的整数 n ，判定它是否为素数(prime，又称质数)。

```
#include <stdio.h>
int main()
{ int n,i;
  printf("please enter a integer number,n=?");
  scanf("%d",&n);
  for (i=2;i<n;i++)
    if(n%i==0) break;
  if(i<n) printf("%d is not a prime number.\n",n);
  else printf("%d is a prime number.\n",n);
  return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
please enter a integer number,n=?17
17 is a prime number.
请按任意键继续. . .
```



若 n 能被 $2 \sim (n-1)$ 之间的一个整数整除，则执行break语句，提前结束循环，流程跳转到循环体之外。此时 $i < n$ 。如果 n 不能被 $2 \sim (n-1)$ 之间任何一个整数整除，则不会执行break语句，循环变量 i 一直变化到等于 n ，然后由第1个判断框判定“ $i < n$ ”条件不成立，从而结束循环。这种正常结束的循环，其循环变量的值必然大于事先指定的循环变量终值(本例中循环变量终值为 $n-1$)。因此，只要在循环结束后检查循环变量 i 的值，就能判定循环是提前结束还是正常结束的。从而判定 n 是否为素数。希望读者理解和掌握这一方法，以后会常用到。

循环程序举例

【例5.9】输入一个大于3的整数 n ，判定它是否为素数(prime，又称质数)。

```
#include <stdio.h>

int main()
{
    int n, i;
    printf("please enter a integer number, n=?");
    scanf("%d", &n);
    for (i=2; i<n; i++)
        if(n%i==0) break;
    if(i<n) printf("%d is not a prime number.\n", n);
    else printf("%d is a prime number.\n", n);
    return 0;
}
```



程序改进:

其实 n 不必被 $2 \sim (n-1)$ 范围内的各整数去除，只须将 n 被 $2 \sim \sqrt{n}$ 之间的整数除即可。因为 n 的每一对因子，必然有一个小于 n ，另一个大于 n 。

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

循环程序举例

【例5.9】输入一个大于3的整数n，判定它是否为素数(prime，又称质数)。

其他求素数方法

```
int i, t = 1; // t: 标志变量
for(t=1, i=2; i<=(int) sqrt(n); i++)
    if(n%i==0)
        t=0; //t=0表示n能被i整除, n不是素数
if(t) //如果t=1表示n是素数
    printf("%d is prime.\n", n);
```



```
for(t=1, i=2; i<=(int) sqrt(n); i++)
    if(n%i==0) {
        t=0;
        break;
    }
if(t)
    printf("%d is prime.\n", n);
```

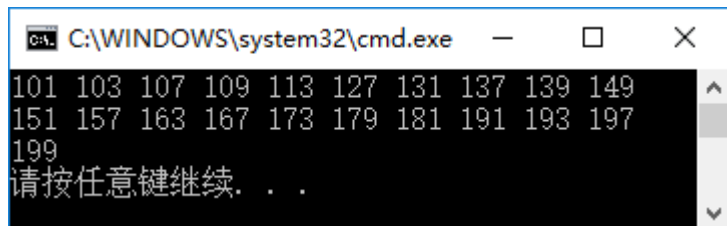


```
for(t=1, i=2; i<=sqrt(n) && t; i++)
    if(n%i==0)
        t=0;
if(t)
    printf("%d is prime.\n", n);
```

循环程序举例

【例5.10】求100~200间的全部素数。

```
#include<stdio.h>
#include<math.h>
int main()
{   int n,k,i,m=0;
    for(n=101;n<=200;n=n+2)           //n从100变化到200，对每个奇数n进行判定
    {   k=sqrt(n);
        for(i=2;i<=k;i++)
            if(n%i==0) break;        //如果n被i整除，终止内循环，此时i<k+1
        if(i>=k+1)                    //若i>=k+1，表示n未曾被整除
        {   printf("%d ",n); //应确定n是素数
            m=m+1;              //m用来控制换行，一行内输出10个素数
        }
        if(m%10==0) printf( "\n" ); //m累计到10的倍数，换行
    }
    printf( "\n" );
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
101 103 107 109 113 127 131 137 139 149
151 157 163 167 173 179 181 191 193 197
199
请按任意键继续. . .
```

循环程序举例

【例5.11】译密码。为使电文保密，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。例如，可以按以下规律将电文变成密码:将字母A变成字母E，a变成e，即变成其后的第4个字母，W变成A，X变成B，Y变成C，Z变成D。

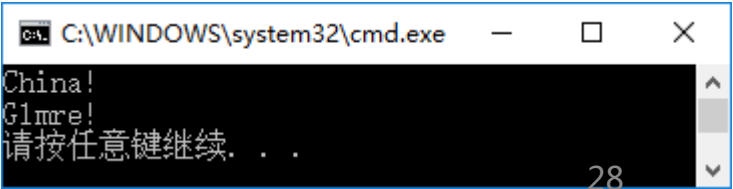
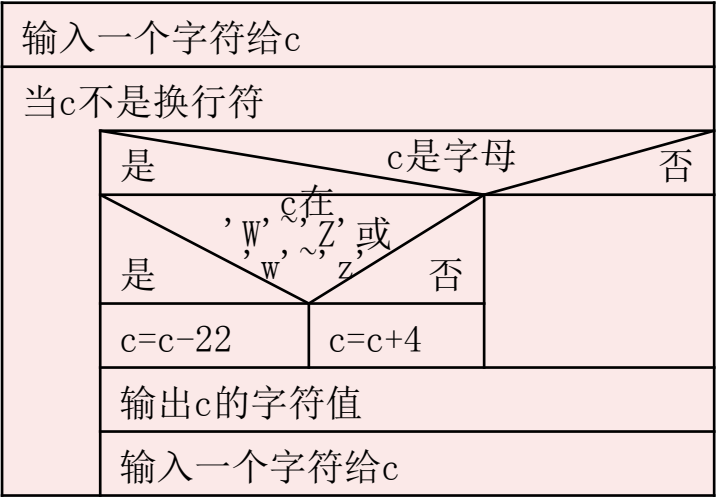
解题思路:

(1) 判断哪些字符不需要改变，哪些字符需要改变。(2) 通过改变字符c的ASCII值的方式将其变为指定的字母。

‘A’ 至 ‘V’ 或 ‘a’ 至 ‘v’ : $c = c + 4$; ‘W’ 至 ‘Z’ 或 ‘w’ 至 ‘z’ : $c = c - 22$ 。

```
#include <stdio.h>
int main()
{
    char c;
    c=getchar();           //输入一个字符给字符变量c
    while(c!='\n')         //检查c的值是否为换行符'\n'
    {
        if((c>='a' && c<='z') || (c>='A' && c<='Z')) //c如果是字母
        {
            if(c>='W' && c<='Z' || c>='w' && c<='z') c = c -22;
            //如果是26个字母中最后4个字母之一就使c - 22
            else c =c + 4; //如果是前面22个字母之一，就使c + 4
        }
        printf("%c", c);           //输出已改变的字符
        c=getchar();              //再输入下一个字符给字符变量c
    }
    printf("\n");
    return 0;
}
```

26个字母的ASCII码顺序排列
 $22 = 26(\text{个字母}) - 4$



例题1：

求 $s=a+aa+aaa+\dots aa...a$ 的值，其中 a 是一个数字。例如 $2+22+222+2222$ (此时 $n=4$), n 和 a 均由键盘输入。

```
int i, s, n, term = 0;
for(i = 1, s=0; i<=n; i++)
{
    term = term*10 + a;
    s += term;
}
```

例题2：

韩信点兵。韩信有一队兵，他想知道有多少人，便让士兵排队报数：

按从1至5报数，最末一个士兵报的数为1；

按从1至6报数，最末一个士兵报的数为5；

按从1至7报数，最末一个士兵报的数为4；

按从1至11报数，最末一个士兵报的数为10；

计算韩信至少有多少兵。

```
int x;  
for(x=1;;x++)  
    if(x%5==1 && x%6==5 && x%7==4 && x%11==10)  
        { 输出x;  break;}
```

例题3：

求水仙花数。如果一个三位数的个位数、十位数和百位数的立方和等于该数自身，则称该数为水仙花数。

编程求出所有的水仙花数。

```
int i, j, k; // 百、十、个位
for(i=1; i<=9; i++) // 百位
    for(j=0; j<=9; j++) // 拾位
        for(k=0; k<=9; k++) // 个位
            if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
                { // 水仙花数 }
```

例题4：

百钱百鸡,已知公鸡5个钱1只,母鸡3个钱1只,小鸡1个钱3只,用100个钱买了100只鸡,问公鸡、母鸡、小鸡各几只?

```
int x, y, z; // 公鸡、母鸡、小鸡个数
for (x=0; x<=100; x++)
    for (y=0; y<=100; y++)
        for (z=0; z<=100; z++)
            if (5*x+3*y+z/3 == 100 && x+y+z == 100 && z%3 == 0)
                {输出x, y, z }
```


注意事项小结

1. `while() { }; do {} while; for(;;) { }` 执行顺序;
2. 循环变量的开始和结束条件;
3. 循环体是复合语句时, 必须用 `{ }` 扩起来;
4. 必要时, 用 `break` 结束整个循环, 用 `continue` 结束本次循环;
5. 关键是找出循环规律, 必要时设计流程图, 指导代码实现。