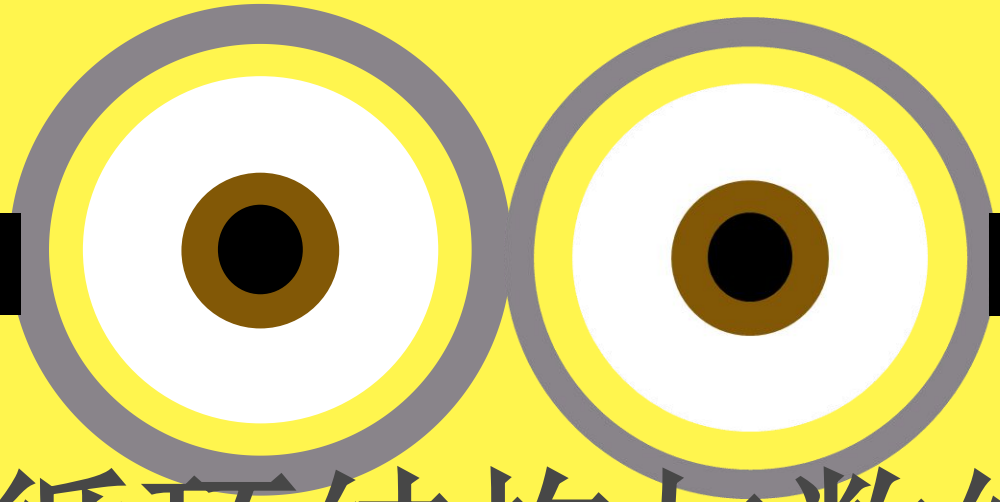


循环结构与数组

C 语言程序设计

循环结构

- *循环*：就是在给定的条件成立时反复执行某一程序段，被反复执行的程序段称为*循环体*。
- 在C语言中可以用以下语句来实现循环：

1、用while语句；

2、用do--while语句；

3、用for语句；

4、用goto语句和if语句构成循环。



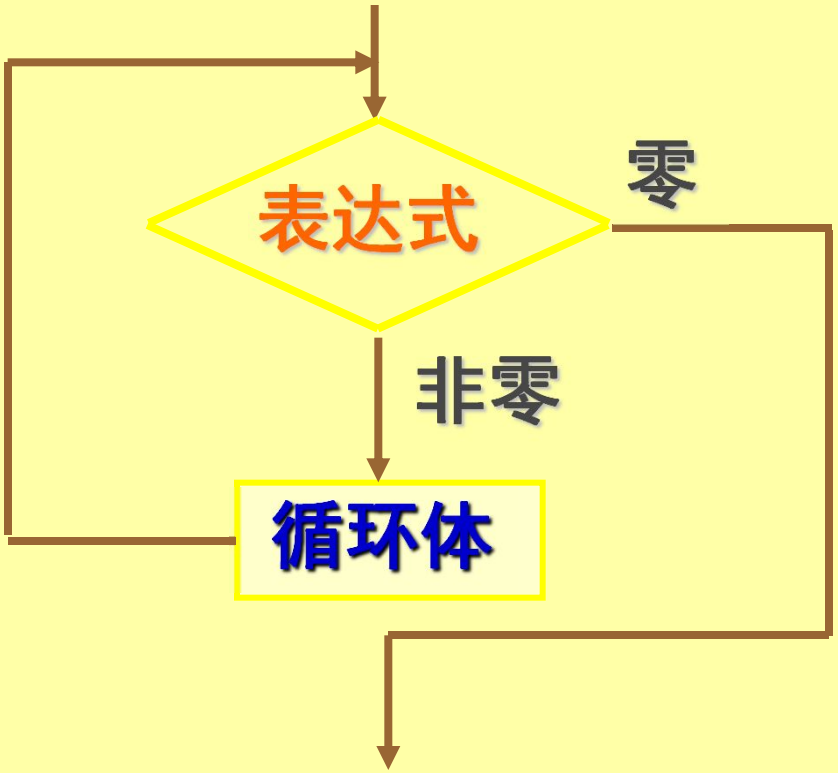


5.1 while语句

1、while 语句 常称为“当型”循环语句。

while(表达式/循环条件)

循环体；



2、while 语句的形式：

while (表达式）

循环体；

↗特点：先判断表达式，后执行语句。

↗说明：

- 循环体有可能一次也不执行
- 循环体可为任意类型语句
- 下列情况，退出while循环
  - 条件表达式不成立（为零）
  - 循环体内遇break,return,goto
- 无限循环：while(1)　　循环体;





例（ch5_01.c）求1+2+3+4+5+・・・+100。

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

循环初值

循环条件

循环变量增值

循环终值

循环体





例（ch5_02.c）显示1~10的平方

```
#include <stdio.h>
main()
{   int i=1;
    while(i<=10)
    {
      printf("%d*%d=%d\n",i,i,i
*i);
      i++;
    }
}
```

运行结果：

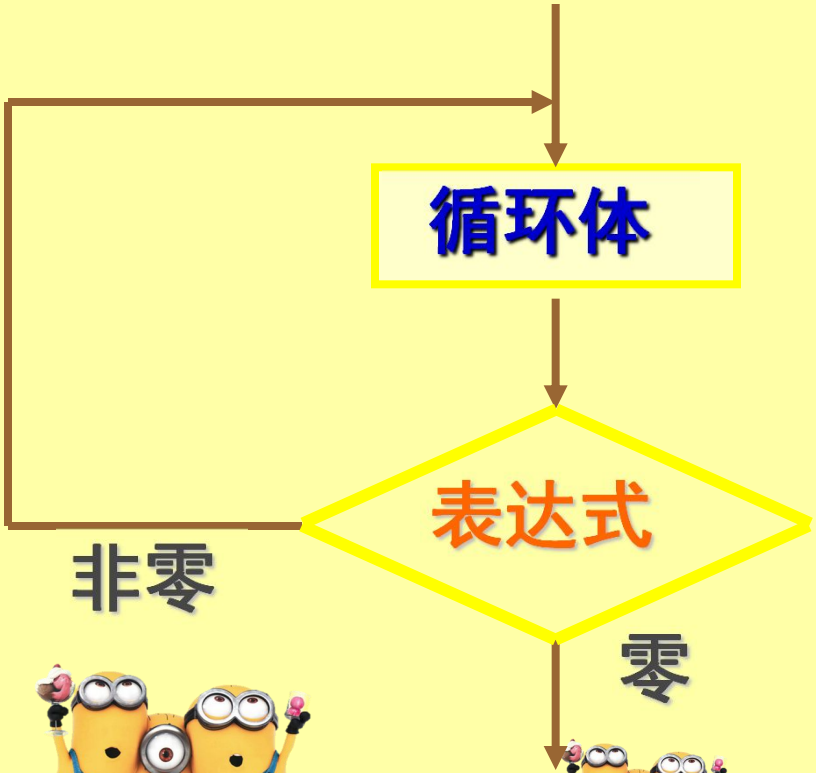
```
1*1=1
2*2=4
3*3=9
4*4=16
5*5=25
6*6=36
7*7=49
8*8=64
9*9=81
10*10=100
```





5.2 do--while语句

1、do--while语句 常称为“直到型”循环语句。



2、do--while的形式：

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

+100。





注:

1. 循环体如果包含一个以上的语句，应该用花括号括起来，以复合语句形式出现。

2. 循环体中应有使循环趋于结束的语句。

例：分析下列三个程序段

```
i=1;
while(i<=100)
putcha('*');
i++;
```

```
i=1;
while(i<=100);
putchar('*'); i++;
```

```
i=1;
while(i<=100)
{ putchar('*');
    i++;    }
```





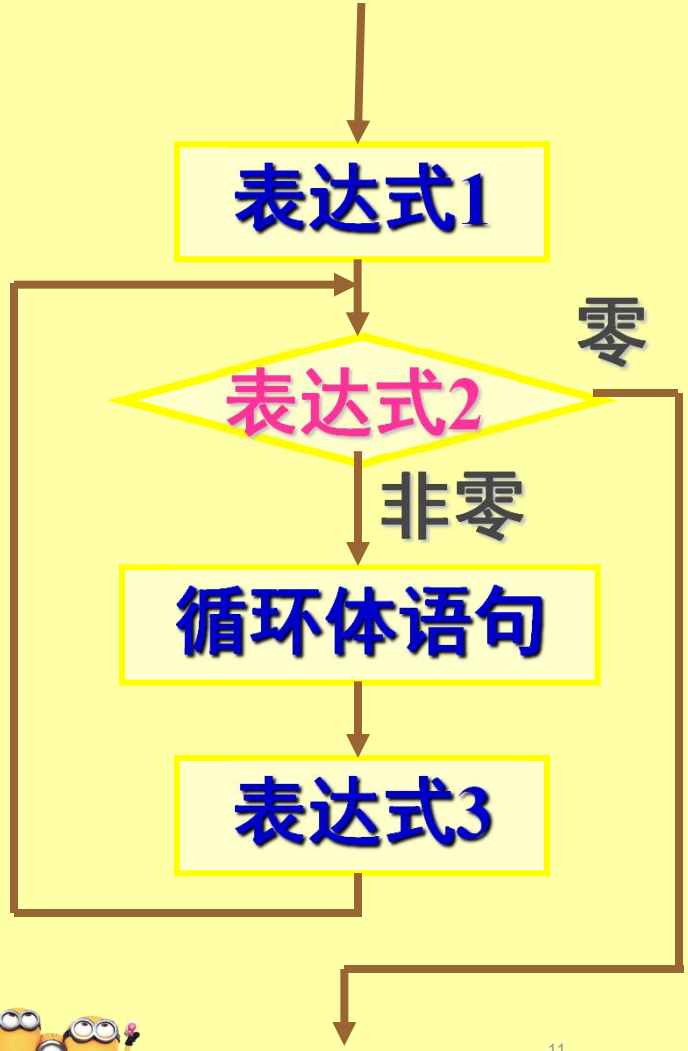
```
main ()
{  int i,sum=0;
   i=1;
   while(i<1)
   {    sum=sum+i;
        i++ ;
   }
   printf("%d\n",sum);
}
```

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

5.3 for语句

1、C语言中最灵活、最复杂的循环语句；

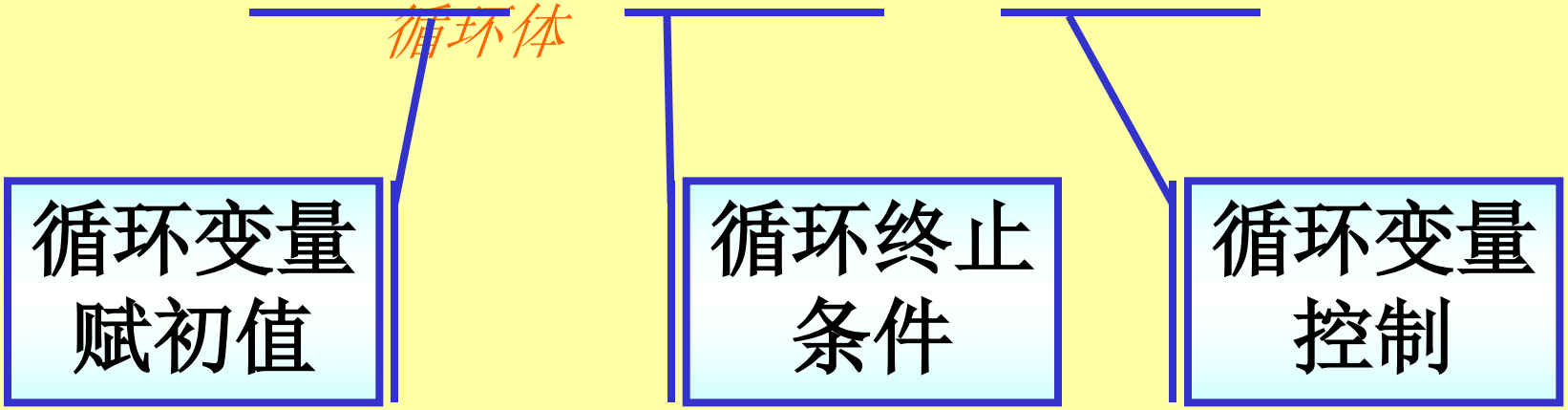
- 可以用于循环次数确定的情况；
- 可以用于循环次数不确定的情况；
- 可实现while和do--while语句的所有功能。



2、for的形式：

for (表达式1；表达式2；表达式3）

循环体



↗说明：

for语句中expr1, expr2 ,expr3 类型任意，都可省略，但分号；不可省

无限循环：for(;;)

for语句可以转换成while结构





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

它相当于以下语句：

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

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





3、for语句中表达式的省略

（1）for语句一般形式中的“表达式1”可以省略；

如： sum=0;i=1;

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

（2）表达式2省略，即不判断循环条件，循环无终止地进行下去；

如： for(sum=0,i=1;;i++)

```
{
    sum=sum+i;
    if(i>=100) break;
}
```





（3）表达式3也可以省略，但此时保证循环能正常结束。

如：for(sum=0,i=1;i<=100;)

```
{ sum=sum+i;
  i++;
}
```

（4） 可以省略表达式1和表达式3，只有表达式2。

如：i=1; sum=0;

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

i=1;sum=0;

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





（5）三个表达式都可省略，

如：　　for (; ;)　循环体;

相当于

　　while (1)　循环体;

即不设初值，不判断条件，循环变量不增值。无终止地执行循环体。

如：　　sum=0,i=1;

　　for(;;)

　　{　if(i>100)　break;

　　sum=sum+i;　i++;

　　}





[例] 统计选票。现有选票如下

3,1,2,1,1,3,3,2,1,2,3,3,2,1,1, 3,2,0,1,4,-1.

-1是结束标志。设1选李, 2选张，3选王，0和4为废票，谁会当选？

解题思路

1.每当我们读入一张选票，只有6种情况，将它们加到相应的人选上。

2.-1结束循环



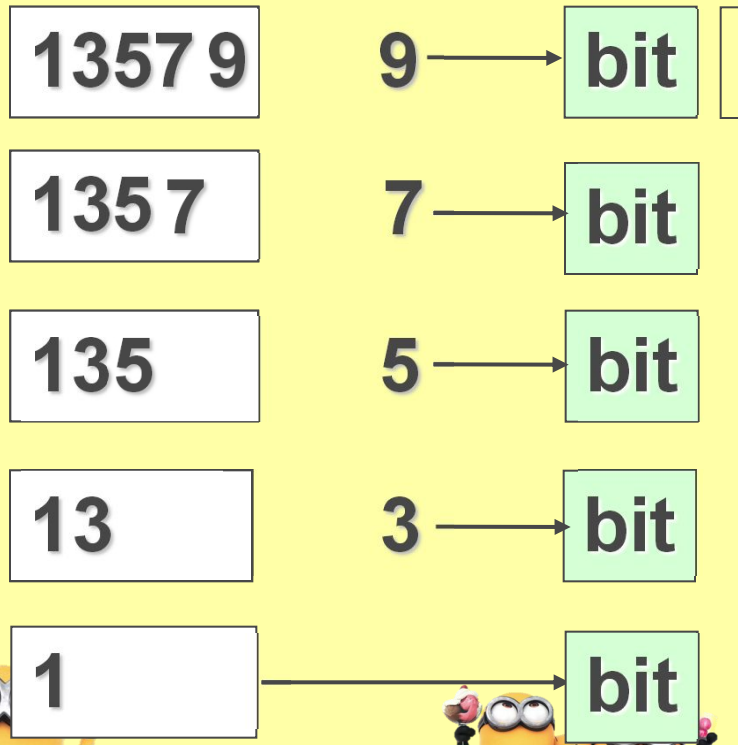


[例] 输入一个整数，计算它的位数.并反向输出.

分析:设一个数13579,一位一位地切下末位

↗循环结束条件: num==0;

↗同时计数 count=count+1;



bit=num%10	num=num/10

count=0;输入num

num!=0

输出num%10

num=num/10

count++

输出count

```
main()
{ long int num;
  int count = 0;
  printf("Please enter an integer:\n");
  scanf("%ld",&num);
  do {
    printf("%d",num%10);
    num = num /10;
    count  + +;
  } while (num !=0 );
  printf(" %d digits.", count);
}
```

Please enter an integer:
3829
9283 4 digits.

5.4 goto语句

1、 无条件转移语句；

形式：goto 标号；

2、有标号的语句称为标号语句；

形式： 标号：语句；

满足标识符的规定

3、 goto语句在使用时只能转移到goto所在的函数内的标号处，不能转移到该函数外；

4、 可以从多重循环的内层转移到最外层，而break只能跳出一层循环。





```
main()
{  int i=1,sum=0;
 loop: if (i<101)
  { sum=sum +i;
     i++;
    goto loop;
  }
 printf("SUM = %f\n",sum);
}
```

```
loop: sum=sum+i;
      i++;
      if(i<101) goto  loop;
```





5.5 break语句和continue语句

break语句

↗一般形式:　　break;

↗功能:

跳出所在的多分支switch语句

跳出所在的while、do-while、for循环语句（提前结束循环）。

continue语句

↗一般形式:　　continue;

↗功能:

提前结束本次循环体的执行，接着进行下一次循环条件的判别。





break语句

```
main()
{  int i;
    for(i=1;i<=5;i++)
    {   printf("\n%5d",i);
        printf("%5d",i);
     }
}
```

```
    1    1
    2    2
    3    3
    4    4
    5    5
```

```
main()
{  int i;
    for(i=1;i<=5;i++)
    {  printf("\n%5d",i);
       if(i==3) break;
        printf("\n%d",i);
     }
}
```

```
    1    1
    2    2
    3
```

当i=3时，结束循环

continue语句

```
main()
{  int i;
    for(i=1;i<=5;i++)
    {   printf("\n%5d",i);
        printf("%5d",i);
     }
}
```

```
1    1
2    2
3    3
4    4
5    5
```

```
main()
{  int i;
    for(i=1;i<=5;i++)
    {        printf("\n%5d",i);
              i f ( i = = 3 )
          continue;
             printf("\n%d",i);
     }
}
```

```
1    1
2    2
3
4    4
5    5
```

当i=3时,结束本次循环体的执行

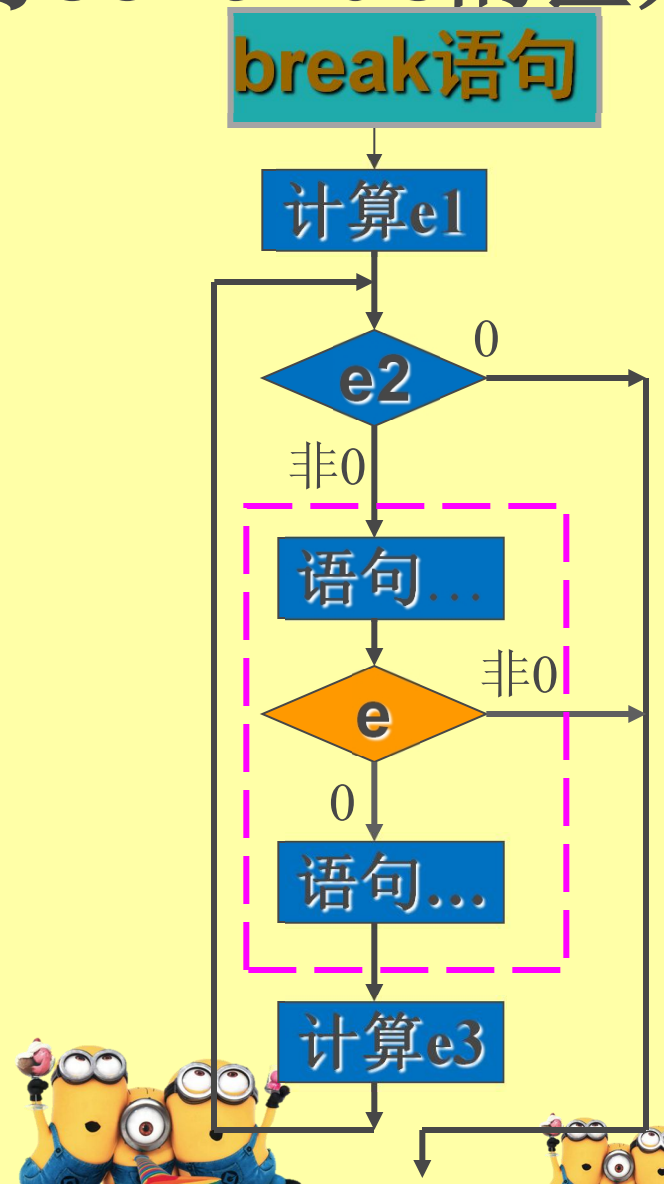


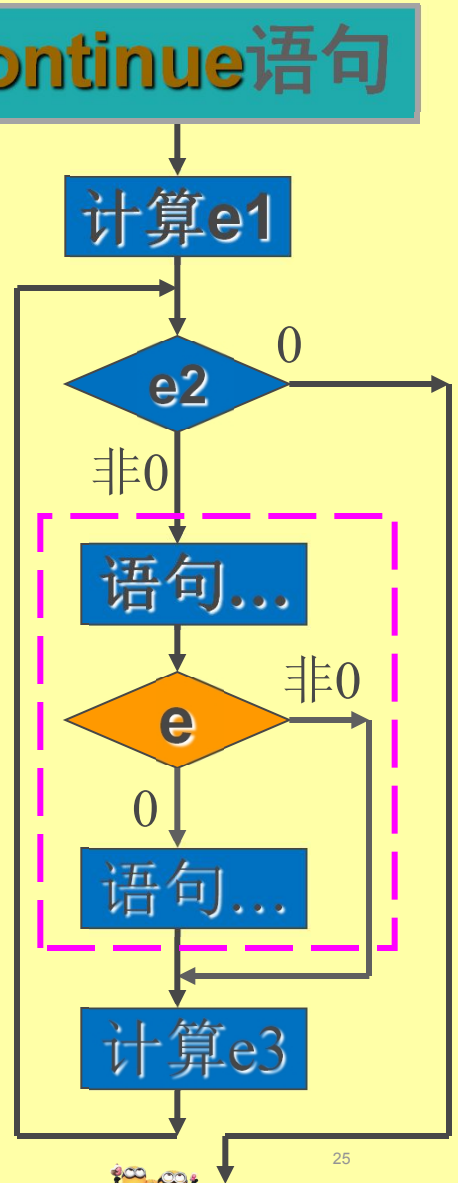


break与continue的区别

```
for( e1;e2;e3)
{  …
    if(e) break;
    …
}
```

```
for( e1;e2;e3)
{  …
    if(e) continue;
    …
}
```





[例]（ch5_05.c）

输出1~10中不是3的倍数的数。

```
main()
  {     int n;
   for (n=1;n<=10;n++)
       {  if (n%3==0)
          break;
          printf("%d,",n);
       }
}
```

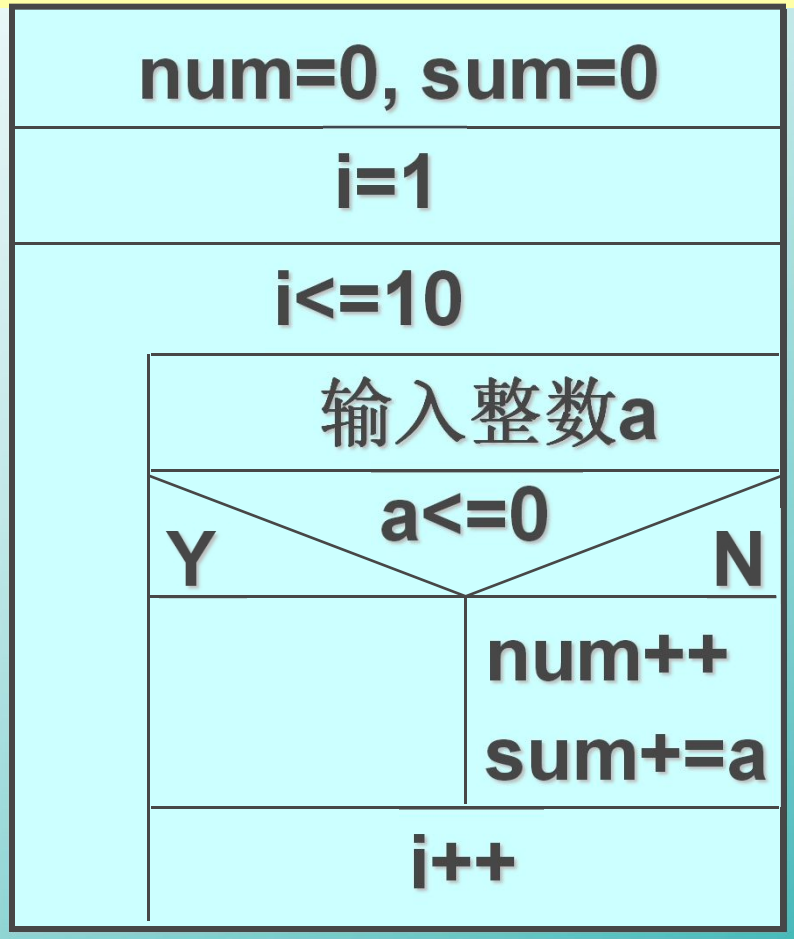
continue；

输出：1,2,4,5,7,8,10,

[例] 求输入的十个整数中正数个数及其平均值。

（ch5_06.c）

```
main()
{    int i,num=0,a;
     float sum=0;
     for(i=1;i<=10;i++)
     {   scanf("%d",&a);
         if(a<=0)
                continue;
         num++;
         sum+=a;
     }
     printf("num=%d，aver=%6.2f\n", num ,sum/num);
}
```



5.6 循环的嵌套

- 三种循环可互相嵌套,层数不限
- 外层循环可包含两个以上内循环
- 嵌套循环的执行流程
- 嵌套循环的跳转

禁止：

从外

跳入

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

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

外循环

内循环

内循环

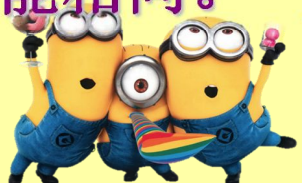
对于嵌套循环结构的几点说明：

（1）当外层循环结构每执行一次循环时，内层循环结构在一般情况下要从循环的开始到循环的正常结束从头到尾执行一遍。 例1

（2）在内层循环结构中使用**break**语句可以提前结束本次内层循环结构的执行，而不影响外层循环结构的继续执行。 例2

（3）如果程序因某种原因需要从内层循环体跳出整个循环结构，此时才可考虑使用 **goto** 语句。 例3

（4）对于并列的循环结构，控制循环执行的变量名字可以相同。在嵌套循环结构中，内、外层控制循环执行的变量名字不能相同。 例4


例1：

```
main()
{  int i,j;
   for(i=0; i<3;i++)
   {
      for(j=1;j<=4;j++)
          printf("  %d",j);
      printf("\n");
    }
}
```

运行后，

输出：

```
1 2 3 4
1 2 3 4
1 2 3 4
```

例2：

```
main()
{  int i,j;
   for(i=0; i<3;i++)
   {   for(j=1;j<=4;j++)
       {  printf("  %d",j);
          if( ! (j%3) ) break;  }
   printf("\n");
   }
}
```

运行后，

输出：

```
1 2 3
1 2 3
1 2 3
```

例3：

```
main()
{  int i, j;
   for( i=0; i<3; i++)
   {   for( j=1;j<=4;j++)
       {  printf("  %d",j);
          if( ! (j%3) ) goto l;  }
       printf("\n");
   }  l:    ;
}
```

运行后，

输出：

1 2 3

例4：

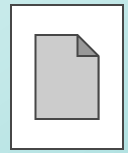
```
main()
{
   int i, j;
   for( i=0; i<3; i++)
        printf("  %d", i );
        printf("\n" ) ;
   for( i=1; i<=4; i++)
         printf("%d", i ) ;
 }
```

运行后，

输出：

0 1 2
1234

[例]（ch5_07.c）求1!+2!+…+n!



分析:

↗求累加和s

s=0

for(k=1;k<=n;k++)

{ 求t_k

s=s+t_k

}

↗求累乘t_k=k!

t_k =1

```
main( )
{  int i,k;
   long  s, t;
   printf("\nInput n:");
   scanf("%d",&n);
   s=0;
   for(k=1;k<=n;k++)
   {   t=1;
       for(i=1;i<=k;i++)
          t=t*i;
       s=s+t;
   }
   printf("\ns=%ld",s);
}
```

```
Input n:5↙
s=153
```

内外层循环控制变量不要同名。

```
main()
{   int i,k;
    long  s, t;
    printf("\nInput n:");
    scanf("%d",&n);
    s=0;
    for(k=1;k<=n;k++)
    {   t=1;
        for(i=1;i<=k;i++)
            t=t*i;
        s=s+t;
    }
    printf("\ns=%ld",s);
}
```

Input n: 3↙
s=9

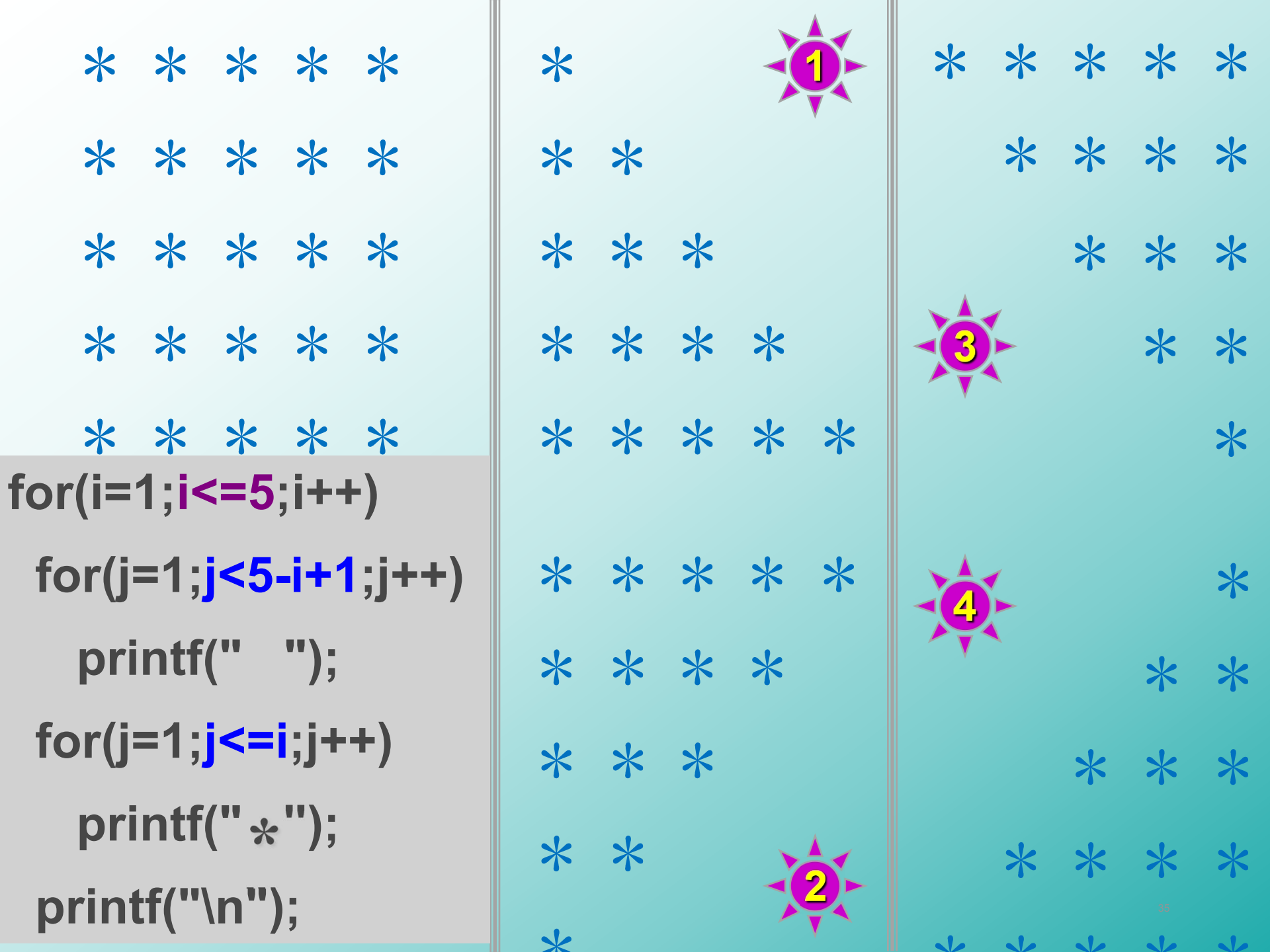
n=3

s	k	k≤3	外循	t	i	i≤k	內循
0	1	1	1	1	1	1	1
				1	2	0	
1	2	1	2	1	1	1	1
				1	2	1	2
				2	3	0	
3	3	1	3	1	1	1	1
				1	2	1	2
				2	3	1	3
				6	4	0	
9	4	0					

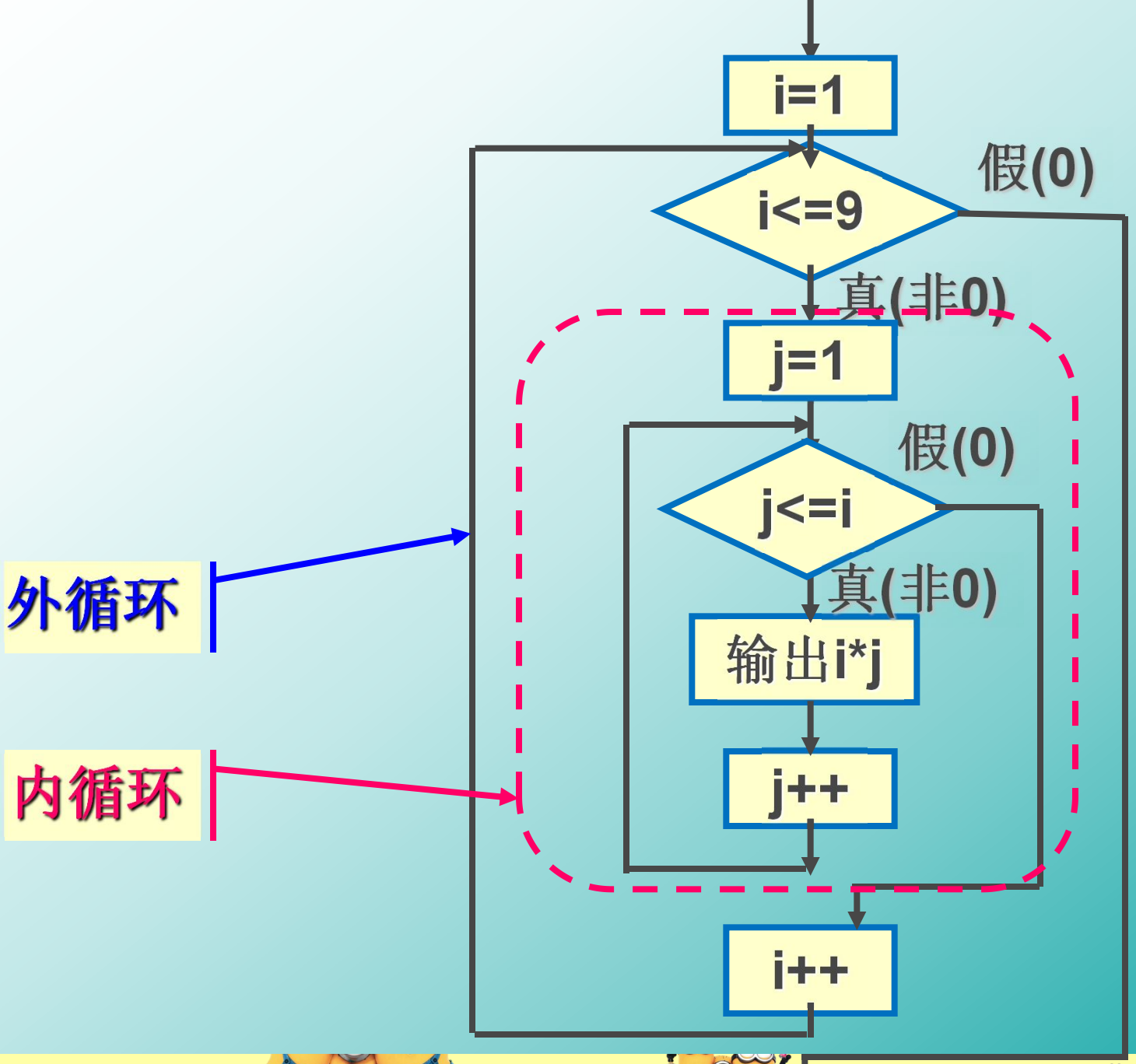
例（ch5_08.c）循环嵌套，输出九九表。








```
for(i=1;i<=5;i++)
  for(j=1;j<5-i+1;j++)
     printf("   ");
  for(j=1;j<=i;j++)
     printf("*");
  printf("\n");
```


i=1
i<=9
假(0)
真(非0)
j=1
j<=i
假(0)
真(非0)
输出i*j
j++
i++
外循环
内循环

分析:

- 行的控制 i:1～9
- ‘*’的个数j与当前行的关系:

  j=2*i-1
- ‘*’前面的空格k与行的关系:

  开始时,第一行有8个空格

  每多一行,少一个空格k=9-i

```
while (i<=9)
{   for (k=1;k<=9-i;k++)
            输出空格;
    for (j=1;j<=2*i-1;j++)
            输出*;
}
```

```
main()
{ int i,j,k;   i=1;
  while (i<=9)
  {
     for(k=1;k<=9-i;k++)
        printf(“ ”);
     for(j=1;j<=2*i-1;j++)
        printf(“*”);
     printf(“\n”);
     i++;
   }
}
```

例(ch5_04.c) 求fibonacci数列0,1,1,2,3,5,8, …的前20项。

fibonacci数列满足下面递归关系：

$F_1=1$ (n=1)

$F_2=1$ (n=2)

$F_n=F_{n-1}+F_{n-2}$ (n≥3)

分析：

1 1 2 3 5 8 13··

a+ b⇒a

b+ a ⇒b

a + b⇒ a

b + a ⇒ b ····

a=1,b=1	
for i=1 to 10	
	输出a,b
	a=a+b b=b+a

```
main()
{     int i,a,b,k=0;
      a=b=1;
            for(i=1;i<=10;i++)
            {       printf("%10d%10d",a,b);
            a=a+b;                  b=a+b;
            k+=2;
            if(k%4==0)        printf("\n");
      }
}
```

1	1	2	3
5	8	13	21
34	55	89	144
233	377	610	987

[例] 编程输出下面的数字金字塔(1到9)。

1

121

12321

……

123456789 87654321

i控制行: i 1~9 空格数: 9-i

j控制列: 第i行 左边: 1~i

右边: i-1~1

（ch5_09.c）

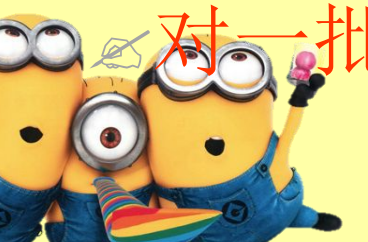
```
main( )
{      int i,j,k,m;
     for (i=1;i<=9;i++)
        { for (j=1;j<=9-i;j++)        /*输出9-i个空格*/
          putchar(‘ ’);
           for (j=1;j<=i;j++)          /*输出1到i*/
                    printf(“%d”, j);
           for (j=i-1;j>0;j--)         /*输出i-1到1*/
                    printf(“%d”, j);
           printf(“\n”); }
}
```

5.7 循环程序设计的问题

写循环，先要发现循环。注意计算中的重复性动作，引进循环可能统一描述和处理。

重复动作的常见例子：

- 累积一批可按规律算出的数据（如累加等）；
- 反复从一个结果算出下一结果
- 对一批数据做同样的加工处理；等。





写循环结构时要考虑和解决的问题：

I. 循环涉及哪些变量，引进什么临时性变量？

II. 这些变量在循环正式开始前应给什么初值？循环如何开始？

III. 每次循环中变量的值应如何改变？

IV. 什么情况下继续循环（什么情况下终止）？

V. 循环终止后如何得到所需结果？





循环中的几种变量

循环中常出现几类变量，了解这些有助于思考和分析。这也是写循环程序的经验总结。

1）循环控制变量(循环变量)：循环前设初值，循环递增/递减，达到/超过界限时循环结束。控制循环的进行/结束。 for中常有这类变量。

for(n = 0; n < 10; n++).....

for(n = 2; n < 52; n += 4) .....





2）累积变量：循环中常用+=或*=等更新。初值常用运算的单位元（加用0；乘用1为初值）。循环结束时变量终值被作为循环计算结果。

3）递推变量：前两类变量的推广形式。复杂循环常用几个协同的变量，每次由一个/几个变量推出一个新值，其余依次更新。

对变量x1、x2、x3，循环体可能有序列：

x1 = x0...;

x2 = x1...;

x3 = ... x1 ... x2 ...;





本章重点

三种循环语句while, do---while和for

建立循环通常有以下情况:

1. 给定次数, for比较适用 for(i=1;i<100;i++)
2. 给定条件, while比较适用 while((x+y)<z)
3. 字符的情况通常以回车做结束符

   while((c=getchar())!='\n')
4. 小经验: 将字符变成数字　i=num-'0';
5. 判断字符范围 (c>'a' &&c <'z')||(c>'A'&& c<'Z')





本章作业

1. 求$2^3+2^4+\ldots+2^{10}$之和。
2. 输入两个正整数a和b，其中a<b,输出a和b组成的闭区间中的所有偶数。
3. 从1开始做自然数的累加，当其累加和超过1000时，共计累加了多少数？当时的累加和是多少？(break 语句)。
4. 请编写程序，其功能是：求出1到1000之内能被7或11整除、但不能同时被7和11整除的所有整数并将它们输出。





5. 编写程序，其功能是：求出能整除x且不是偶数的各整数(x由键盘输入)，并将它们输出。例如，若x 中的值为：30，则有4个数符合要求，它们是1，3，5，15。

6. 有一个数，用3除余2，用5除余3，用7除2，请找出满足条件的最小数。

7. 编程打印出如下矩阵：(5^1~5^{10})

5

25

125

0625

03125

……

0009765625

1.

方法一

```
main()
{    int k,i,s=0,t=4;
  for(k=3;k<=10;k++)
     {
       t=t*2;
          s+=t;
       }
  printf("s=%d\n",s);
}
```

方法二

```
#include "math.h"
main()
{       int k,i,s=0,t=4;
        for(k=3;k<=10;k++)
          {
                t=pow(2,k);
           s+=t;
        }
        printf("s=%d\n",s);
}
```

2 .

```
main()
{ int a,b,t,x;
  scanf("%d%d",&a,&b);
  if(a>b)
  {  t=a;    a=b;   b=t;    }
  for(x=a;x<=b;x+=2)
     if(x%2)
     printf("%4d",x+1);
     else
     printf("%4d",x);
```





3.

方法一

```
main()
{ int a,n,s;
  s=0,n=0;
  for(a=1;;a++)
    {    n++;
     s+=a;
     if(s>1000)   break;
    }
  printf("s=%d,n=%d",s,n);
}
```

方法二

```
main()
{   int a,n,s;
    s=0,n=0;
    for(a=1; s<=1000;a++)
          {
                n++;
                s+=a;
          }
printf("s=%d,n=%d",s,n);
}
```

5.

求全班30名同学的某门功课在平均成绩以上的人数。

```
main()
{ int i,n=0;
  float score[30],aver=0.0;
  for(i=0;i<30;i++)
    {aa: scanf("%f",&score[i]);
     if(score[i]>100||score[i]<0)
       {  printf("Error! ");    goto aa;    }
     aver+=score[i];
    }
  aver=aver/30;
  for(i=0;i<30;i++)
    if(score[i]>aver)  n++;
  printf("aver=%f,n=%d\n",aver,n);
}
```





7.

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





5.8 程序举例

[例] 用π/4≈1-1/3+1/5-1/7+…的公式求π的近似值，直到最后一项的绝对值小于10⁻⁶为止

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

[例] 求 s=1-1/2+1/3-1/4+… +1/99-1/100。

方法一：数列是正、负相间的，在这里可用一个“开关”变量 t 来解决符号的问题。

```
main()
  {   float s=0;
      int t=1,i ;
      for(i=1;i<101;i++)
        {   s+=1.*t / i ;        t= -t ;    }
     printf("s=%f\n",s);
  }
```

在处理循环时应注意初、终值的设定！





方法二：分别求出正项和s1(奇数倒数)与负项和s2(偶数倒数)，则s=s1-s2.

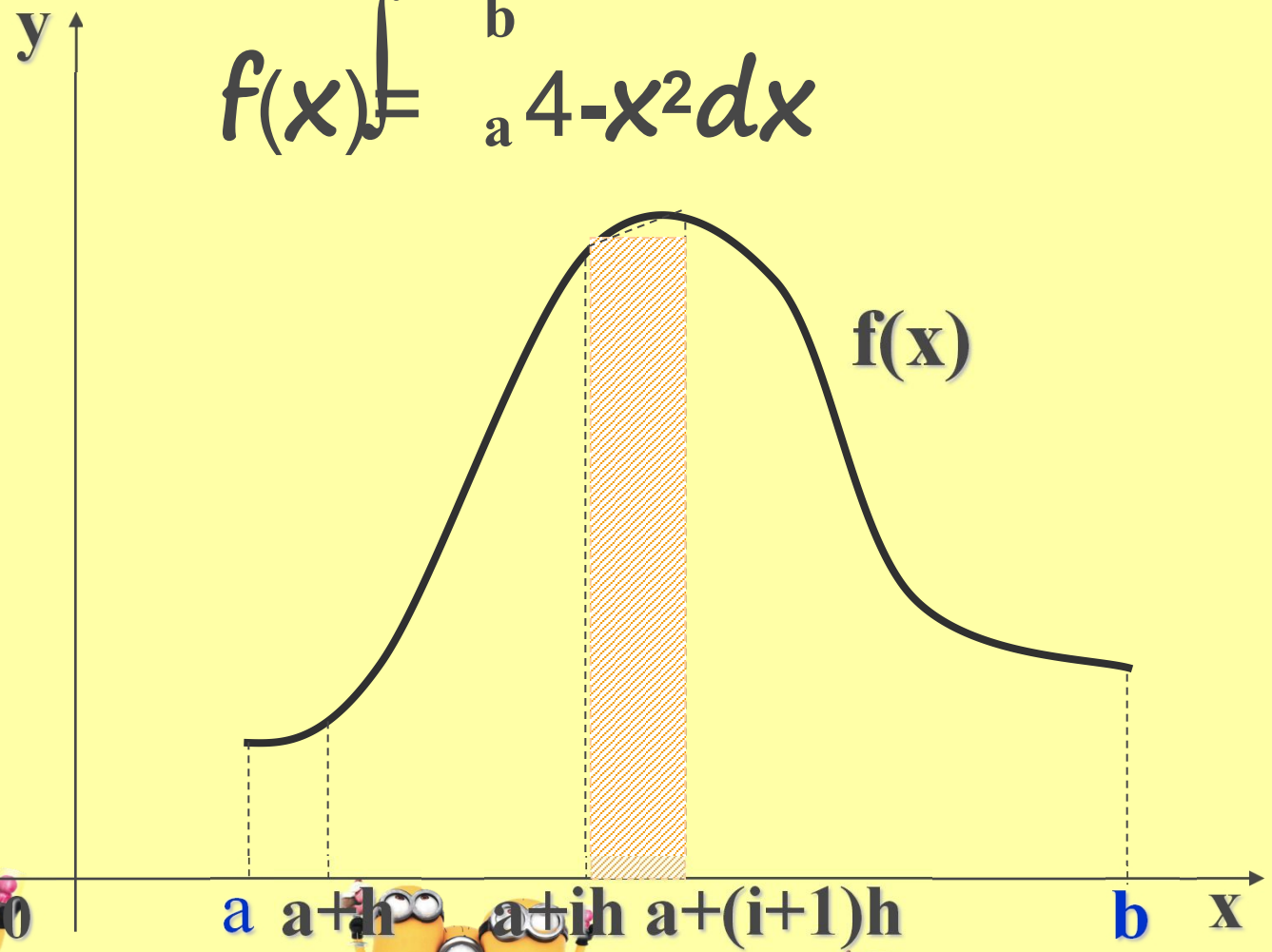
```
main()
{   float s,s1=0,s2=0;
    int i;
    for(i=1;i<101;i+=2)
     { s1+=1.0/ i;
       s2+=1.0/(i+1);   }
    s=s1-s2;
    printf("s=%f\n",s);
}
```



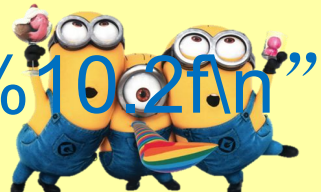


[例] 梯形法求数值积分。

$$f(x)=\int_a^b 4-x^2dx$$



```
main()
{   float a,b,h,n1,n2,s=0;
  int i;
  printf("请输入积分限a和b:");
  scanf("%f%f",&a,&b);
  h=(b-a)/1000;
  for(i=0;i<1000;i++)
      {  n1=4-(a+i*h)*(a+i*h);
         n2=4-(a+(i+1)*h)*(a+(i+1)*h);
         s+=(n1+n2)*h/2;
       }
  printf("s=%10.2f\n",s);
```


[例] 猜数游戏的最简化版。

```
#include<stdio.h>
#define GOAL 39
main()
{ int i;
  printf(“Please input a integer number:\n");
  do
  {  scanf("%d",&i);
     if(i>GOAL)   printf("%d is too big,input again.\n",i);
     else    if(i<GOAL)   printf("%d is too samll,input again.\n",i);
          else     printf("OK!\n");
     } while(i!=GOAL);
}
```

```
#include<stdio.h>
#include<stdlib.h>
main()
{  int i;
   int GOAL,num=0;
   printf("Please input a number from 0 to 99, or -1 to
  exit:\n");
   GOAL=random(100);    /*产生0到99的随机数*/
   do
   {    scanf("%d",&i);
      if(i==-1)
     {     printf("\nExit game!");
           break;    }
```

```
num++;
if(i>GOAL)  printf("%d is too big, input again.\n",i);
else    if(i<GOAL)  printf("%d is too samll, input
  again.\n",i);
     else
     {    printf("OK!\n");
               printf("You take %d times to pass!",num);
          }
} while(i!=GOAL);
}
```





[例] 枚举问题或称为穷举法。一般用于不定方程求非负整数解的问题。它将方程中未知数可以取的到的非负整数逐个进行验证找出所有满足方程的解。

例如：一元人民币兑换成1分、2分、5分共有多少种方法？

若5分、2分、1分的个数分别为x个、y个、z个，则x的取值为0~20，y的取值为0~50，z的取值为0~100。

于是有不定方程：　　5x+2y+z=100。





考虑程序的优化问题？

```
main()
{  int i,j,k,m=0;
    for(i=0;i<21;i++)
      for(j=0;j<51;j++)
                  for(k=0;k<101;k++)
  if(5*i+2*j+k==100)
                m+=1;
    printf("m=%d\n",m);
}
```

运行结果：
m=54130





上面程序的循环次数超过10万次，且大量的循环都不满足方程。可对程序进行优化，由于随着5分个数的增加，2分个数就会减少，因此循环变量j可控制在（100-5i）/2以内,这样使得5i+2j<=100。若不足100则补充1分，将问题转换成为求循环次数的问题了。

```
main()
{ int i,j,m=0;
  for(i=0;i<21;i++)
     for(j=0;j<=(100-5*i)/2;j++)
          m+=1;
  printf("m=%d\n",m);
```

因为该不定方程中，如果有两个未知数确定后，第三个未知数也随之确定。即确定了一种分法。例如：

i=20时，j=0

i=19时，j=0,1,2

i=18时，j=0,1,2,3,4,5





[例] 输入若干字母,将它们变成其后的第四个字母，A-->E, W-->A.非字母字符忽略。

↗思路:

1.建立循环, 循环结束以输入回车符为准

while (c=getchar()!='\n')

2. 判断输入是否是字符,否则忽略

if ((c>='a' && c<='z') || (c>='A' && c<='Z'))

3.变成其后的第四个字母 c=c+4;

4. 若变换后超出z时,要轮回.

If ((c>'Z' && c<'a')||(c>'z')) c=c-26

…A…Z….a….z…..





```
#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<=’a’ )||(c>’z’)) c=c-26;
    }
    printf(“%c”,c);
  }
}
```

abdEgW

efhIkA





[例] 输入若干数字，-1为输入结束标志，计算它们的平均数。

```
main()
{ float  value, total, average; int counter;
  total=0;counter=0; average = 0;
  scanf (“%d”,&value);
  while (value !=-1) {
    total = total + value; counter + +;
    scanf (“%f”,&value); }
  if (counter == 0) printf (“No data entered.\n”);
  else { average = total / counter;
    printf (‘The average of %d values is %f”,
    counter,average) }
}
```

```
23.9  85.68  227E02  0.00863 75
93.44  71  14.7E-05  66  -1
The average of 9 values is 2568.336412
```

[例] 用循环语句显示下面的图案。

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

本例还是要考虑每行的空格数、和星号数问题，但要关注空格数与星号数在增加到一定的时候又要减少的规律。





```
for(i=0;i<4;i++) {
   for(j=0;j<20-i;j++)  printf(“ ”); /*空格递减*/
   for(k=0;k<2*i+1;k++)  printf(“*”); /*星号递增 */
}
```

```
{   int i,j,k;
    for(i= -2;i<=2;i++)
    {  for(j=1;j<=18+fabs(i);j++)
            printf("  ");
       for(k=1;k<=5-2*fabs(i);k++)
            printf("*");
        printf("\n");
    }
```

是:

20,19,18,19,20。

每行的星号数是:
1,2,3,2,1。





[例] 汽车里程表上的读数是95859，7小时之后里程表的读数是一个对称数（最大是5位数），问汽车的速度（是一个整数）。

解题思路:

↗检查所有的在95859到99999之间的对称数，如果它与95859的差能被7整除，则商是速度。

/* 第一种算法*/





```
main()
{ long i,a,b,c,e,d;
  flaot f,g;
  i=95859;
  while (i<=99999 ) {
    a=i/10000;
    b=(i-a*10000)/1000;
    c=(i-a*10000-b*1000)/100;
    d=(i-a*10000-b*1000-c*100)/10;
    e=i-a*10000-b*1000-c*100-d*10;
    if ((a==e) && (b==d))
      if((i-95859)%7)==0)
   printf("The speed is %d",(i-95859)/7);i++;}
 }
```

该程序需要执行循环99999-95859次！





```
/*第二种算法*/
main()
{ long i,distance,a,b,c,d;
  (for i=1;i<=200;i++) {
    distance=95859+i*7;
    a=distance/10000
    b=(distance-a*10000)/1000;
    c=(distance-a*10000-b*1000)/100;
    d=(distance-a*10000-b*1000-c*100)/10;
    e=distance-a*10000-b*1000-c*100-d*10;
    if (a==e) and (b==d)
      printf("The speed is %d",i);
  }
}
```

循环次数为 200-20





该程序需要循环(9-6)*10+1次！

```
/*第三种算法*/
main()
{ long  i,j,distance,speed:real;
  for(i=6;i<=9;i++) {
    for (j=0;j<=9;j++) {
     distance=90000+i*1000+j*100+i*10+9;
     if (distanc%7)==0)
       printf(“The speed is %d”,distance/7);
    }
  }
  distance = distance-95959
  if ((distance%7)==0)
   printf(“The speed is %d”,distance/7);
}
```





分析：

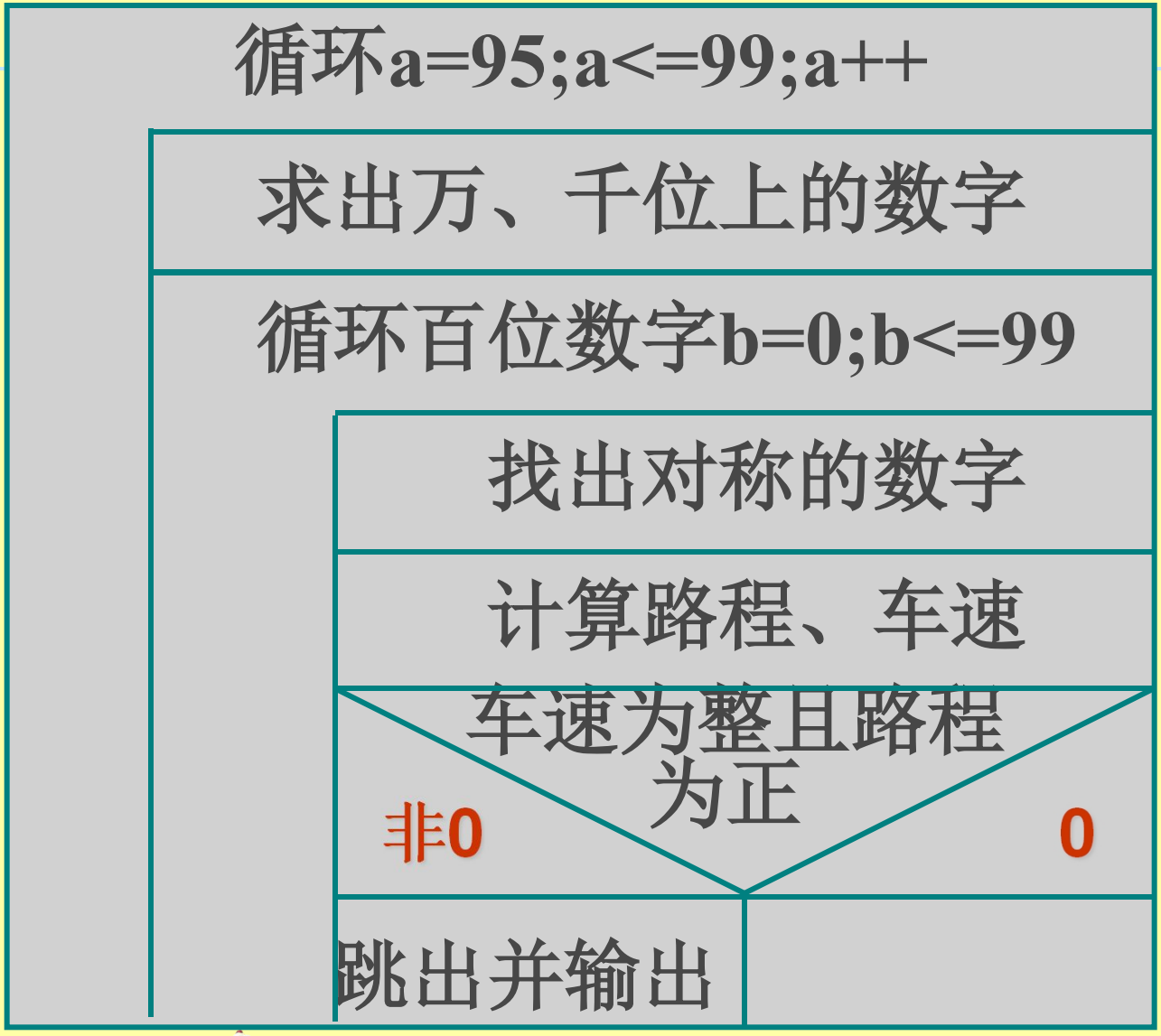
（1）因为新出现的数（dcbcd）是：个位数字（d）与万位数字、十位数字（c）与千位数字相同，而百位数字（b）只能是0~9，表示万位和千位的变量a取值范围为：95~99。所以，要将a分隔出的十位数字和个位数字分别赋予d和c 。

（2）约束条件是：公里数对称且车速为整数。

注意：95859超过了整型量的范围，应使用长整型量。






循环a=95;a<=99;a++
求出万、千位上的数字
循环百位数字b=0;b<=99
找出对称的数字
计算路程、车速
车速为整且路程为正
非0
0
跳出并输出

```
main()
 {  int a,b,c,d,n,v;     long int m;
    for(a=95;a<=99;a++)
    { d=a/10; c=a-d*10; /*分别求出万、千位上的数字 */
      for(b=0;b<=9;b++)      / *百位数从0到9循环 */
        {  m=(long)1000*a+100*b+10*c+d;
           n=m-95859;
           v=n/7;          /*求出车速 */
           if(n%7==0 && n>0) goto loop;
        }
    }
 loop: printf("v=%dkm/h   m=%ldkm\n",v,m);
```

找出对称的数字

车速为整数且路程为正时转出循环并输出结果

在本例这样的情况下，从内循环体直接使用goto语句跳出整个循环结构，收到了事半功倍的效果。但我们再次强调，要限制、最好不使用goto语句。

思考：

（1）若将if语句后goto语句改为break语句，结果如何？

（2）若将printf("v=%d　m=%ld\n",v,m);直接放在if语句后，结果又如何？





5. 9 课堂练习

↗百钱买百鸡:鸡翁1,钱值5; 鸡母1,钱值3;鸡 雏3,钱值1;何以百钱买百鸡?

```
main()
{ int x,y,z;
  for (x=1 to 20)
    for (y=1 to 33)
      if ((x*5+y*3+(100-x-y)/3.0)==100)
          printf("x=%d,y=%d,z=%d",x,y,100-x-y);
}
```





↗打印出100到200之间所有的素数

```
#include “math.h”
main()
{ int n,m, i, k;
  for (n=100;n<=200;n++)
  {   k=sqrt(n);
      for (i=2;i<=k;i++)
          if ((n % i)==0) break;
      if (i=k+1)
          printf(“%d prime number\n”,m);
      else
          printf(“%d not a prime\n”,m);
}
```





课外练习

一、每个苹果0.8元，第一天买两个苹果。从第二天开始，每天买前一天的2倍，当每天购买苹果的数大于100时,则停止. 求平均每天花多少钱?

二、输入一行字符，回车为结束，分别统计出其中的英文字母，空格，数字和其它字符的个数。





求分数数列前 $\frac{2}{1}-\frac{3}{2}+\frac{5}{3}-\frac{8}{5}+\frac{13}{8}-\frac{21}{13}\cdots$ 20项之和。

解法一：

```
main()
{   int i,t,n=20,f=1;
    float a=2,b=1,s=0;
    for(i=1;i<=n;i++)
    {     s=s+a/b;
          t=a;
          a=a+b;
          b=t;
          f=-f;
    }
    printf("sum=%10.6f",s);
}
```

解法二：

```
main()
{   int i,t,n=20,f=1;
    float a=2,b=1,s=0;
    for(i=1;i<=n;i++)
    {     s=s+a/b;
          t=a;
          a=a+b;
          b=t;
          f=-f;
    }
    printf("sum=%10.6f",s);
}
```