计算机导论与程序设计 [CS006001]

段江涛

机电工程学院

西安电子科技大学 XIDIAN UNIVERSITY

2022 年 10 月

Outlines

Part I

循环结构程序设计

Outlines

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)

```
float score1,score2,score3,score4,score5,aver; // 5门课成绩及平均成绩
// 输入第1个学生5门课的成绩
scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
// 求第1个学生平均成绩
aver=(score1+score2+score3+score4+score5)/5;
printf("aver=%7.2f",aver); // 输出第1个学生平均成绩
// 输入第2个学生5门课的成绩
scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
// 求第2个学生平均成绩
aver=(score1+score2+score3+score4+score5)/5;
printf("aver=%7.2f",aver);// 输出第2个学生平均成绩
...
```

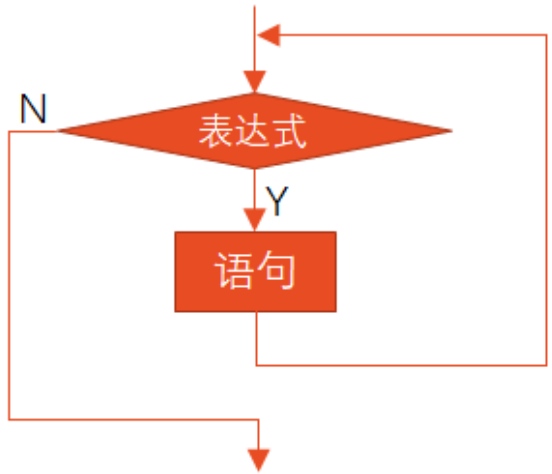
用循环控制处理重复操作

- 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)

```
float score1,score2,score3,score4,score5,aver; // 5门课成绩及平均成绩
int 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
}
```

while(表达式){···}

```
while(表达式)
{
    // 循环体
    执行多条语句;
}
```



while 循环特点

每轮循环: 首先判断表达式的值, 若“真” (表达式值为非 0) 时, 就执行循环体语句; 为“假” (表达式值为 0) 时, 就不执行循环体语句。

常见错误

```
while(表达式);
{
    // 循环体
    执行多条语句;
}
```

变体

```
while(1)
{
  if(表达式) break; // 退出循环
  执行多条语句;
}
```

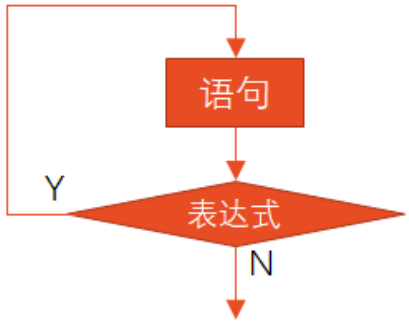
[例 5.1] 求 $1+2+3+\cdots+100$, 即 $\sum_{i=1}^{100} i$

```
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, 条件表达式的值为假(0), 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, 为下轮累加做准备
} //循环体结束
printf("sum=%d\n",sum); //输出1+2+3…+100的累加和
```

1. 循环体如果包含一个以上的语句,必须用花括号括起来,作为复合语句出现。
2. 不要忘记给 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。

do{···}while(表达式);

```
do
{
  // 循环体
  执行多条语句;
} while(表达式);
```



循环特点

首先无条件地执行循环体,然后判断循环条件是否成立。

易犯错误

```
do
{
  // 循环体
  执行多条语句;
} while( 表达式 )
```

变体

```
do
{
  执行多条语句;
  if(表达式) break; // 退出循环
} while(1);
```

[例 5.2] 求 $1+2+3+\cdots+100$, 即 $\sum_{i=1}^{100} i$

```
int i=1,sum=0; //定义变量i的初值为1,sum的初值为0
do
{ //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
}while(i <= 100); //当i>100，条件表达式i<=100的值为假，不执行循环体
printf("sum=%d\n",sum); //输出1+2+3…+100的累加和
```

1. 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。
2. 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。

while(表达式){···} 与 do{···}while(表达式);

[例 5.3] 求 $\sum_{i=n}^{100} i$。考虑输入 $n > 100$ 时的情况，以下程序的不同。

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

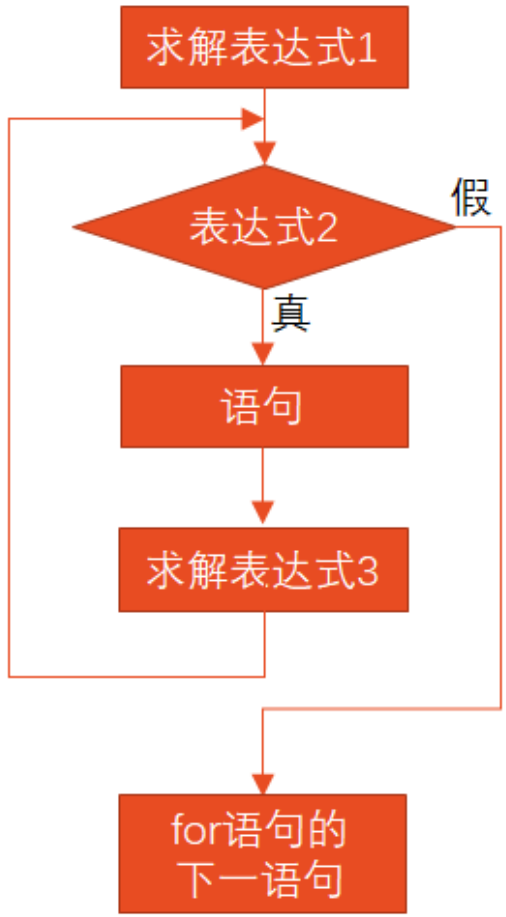
```
int i,sum=0;
scanf("%d",&i);
do
{
  sum=sum+i;
  i++;
}while(i <= 100);
printf("sum=%d\n",sum);
```

for(表达式 1; 表达式 2; 表达式 3){···}

```
for(表达式1;表达式2;表达式3)
{
   // 循环体
   执行多条语句;
}
```

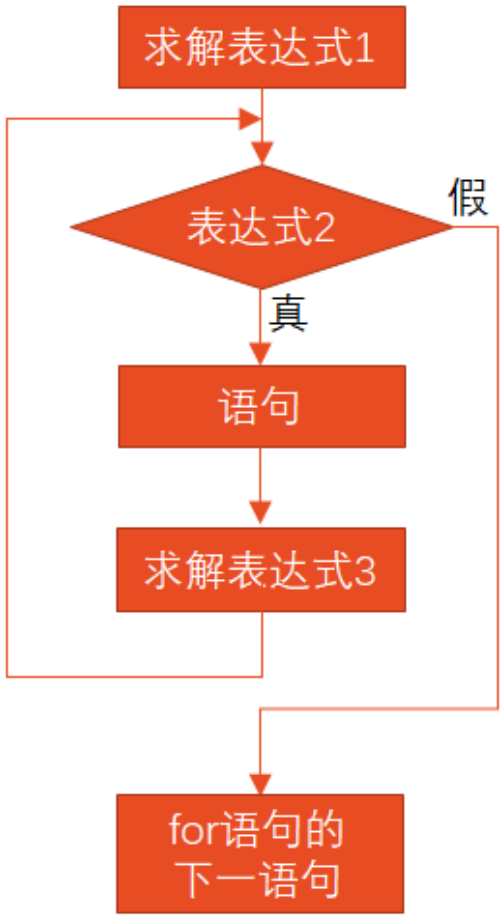
≡

```
表达式1;
while(表达式2)
{
   // 循环体
   执行多条语句;
   表达式3;
}
```



for(表达式 1; 表达式 2; 表达式 3){···}

```
for(表达式1;表达式2
      ;表达式3)
{
   // 循环体
   执行多条语句;
}
```



- 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 (逗号隔开) 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 (包括第 1 次循环),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。

for(表达式 1; 表达式 2; 表达式 3){···}, 省略表达式

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

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

```
int i=0;
for(i=0;;i++)
{
  if(i>100) break;//退出循环
  printf("%d\n",i);
}
```

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

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

```
int i=0;
for(;;)
{
  if(i>100) break;//退出循环
  printf("%d\n",i);
  i++
}
```

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++ 等)。
3. 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 (逗号隔开)。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。
4. 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。
5. 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。

break,continue 改变循环执行的状态


```c
while(表达式)
{
    printf("语句1\n");
    //提前终止循环
    if(条件表达式) break;
    printf("语句2\n");
}
```

```c
while(表达式)
{
    printf("语句1\n");
    //结束本次循环，进入下轮循环
    if(条件表达式) continue;
    printf("语句2\n");
}
```

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。

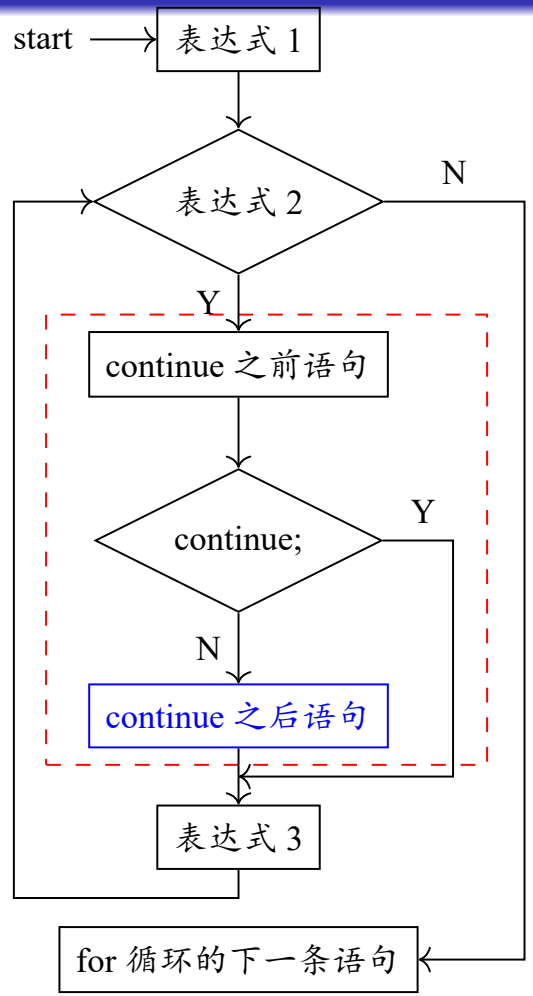
```
#define SUM 100000 //指定符号常量SUM代表10万
float amount,aver,total;
int i;
for (i=1,total=0; i<=1000; i++)// 表达式1给多个变量赋初值, 用逗号隔开。
{
  printf("please enter amount:");
  scanf("%f",&amount);
  total = total + amount;
  if(total >= SUM) break; // 终止整个循环,也不会执行for语句的表达式3 (i++)
}
i=(i<=1000)?i:(i-1); // 技巧: 判断循环是否提前终止
aver=total/i;
printf("num=%d\naver=%10.2f\n",i,aver);
```

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 语句终止本次循环，开始下轮循环。

[例 5.4] 要求输出 100～200 之间的不能被 3 整除的数。

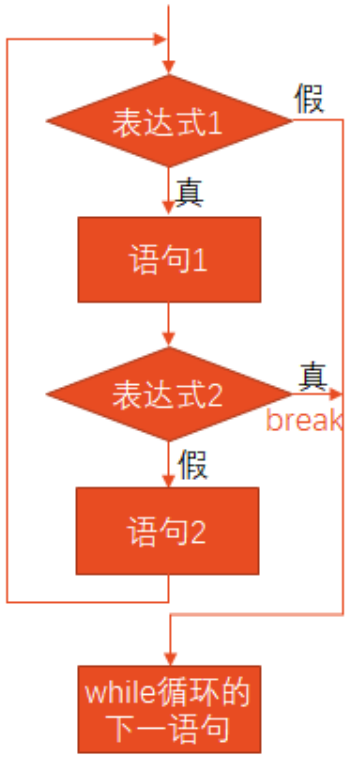
```
#include <stdio.h>
int main()
{
    int n;
    for (n=100;n<=200;n++)//(表达式1;表达式2;表达式3)
    {
        //终止本轮循环,但会执行for语句的表达式3(n++),开始下
            轮循环
        if (n%3==0) continue;
        printf("%d ",n);
    }
    printf("\n");
    return 0;
}
```



break 语句和 continue 语句的区别

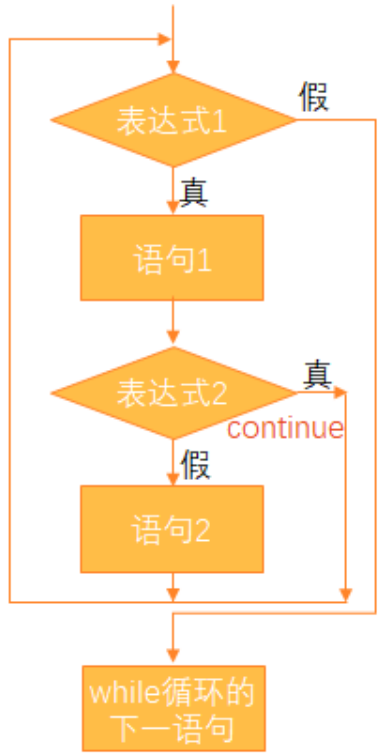
break;

```
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;
}
```



注意: 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 5.5] 输出以下 4 × 5 的矩阵。

```
int i,j,n=0; //n: 累计输出数据个数
for(i=1;i<=4;i++) // 行
{
  for(j=1;j<=5;j++,n++) //列
  {
    if(n%5==0) printf("\n");//5数换行
    if (i==3 && j==1) break;
    printf("%d\t",i*j);
  }
}
printf("\n");
```

```
int i,j,n=0; //n: 累计输出数据个数
for(i=1;i<=4;i++) // 行
{
   for(j=1;j<=5;j++,n++) // 列
   {
     if(n%5==0) printf("\n");//5数换行
     if (i==3 && j==1) continue;
     printf("%d\t",i*j); // \t就是Tab
          键，是特殊字符，表示多个空格
   }
}
printf("\n");
```

分析下列程序片段, 体会 break 与 continue 的不同

```
int i=0;
while(i<4)
{
  if(i==2)
  {
    i++;
    continue;//终止本轮循环,开始下轮循环
  }
  printf("i=%d,",i);//i=0,i=1,i=3,
  i++;
}
printf("\nend i=%d\n",i);//end i=4
```

```
int i=0;
while(i<4)
{
  if(i==2)
  {
    i++;
    break; // 终止整个循环
  }
  printf("i=%d,",i);//i=0, i=1,
  i++;
}
printf("\nend i=%d\n",i);//end i=3
```

分析下列程序片段, 体会 break 与 continue 的不同

```
int i=0;
for(i=0;i<4;i++) //对于continue语句,
    表达式3是本轮循环的一部分
{
  if(i==2) continue; //终止本轮循环,但会
      执行for语句的表达式3 (i++), 开始下
      轮循环
  printf("i=%d,",i); //i=0,i=1,i=3,
}
printf("\nend i=%d\n",i);
//end i=4
```

```
int i=0;
for(i=0;i<4;i++) //对于break语句, 终
    止整个循环, 表达式3也不会执行
{
  if(i==2) break; //终止整个循环,也不
      会执行for语句的表达式3 (i++)
  printf("i=%d,",i); // i=0,i=1,
}
printf("\nend i=%d\n",i);
//end i=2, 此处根据循环变量i的值可以判
    断上述循环是否正常结束, if(i==4)
    正常结束。
```

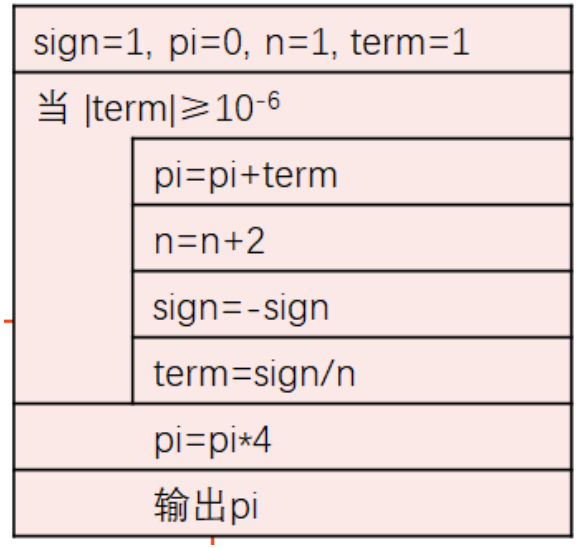
注意事项小结

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 结束本轮循环;
5. 关键是找出循环规律,必要时设计流程图,指导代码实现。

[例 5.7] 用公式 $\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$ 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}。



```c
#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π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π的近似值
  printf("pi=%10.8f\n",pi); //输出π的近似值
  return 0;
}
```

[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?

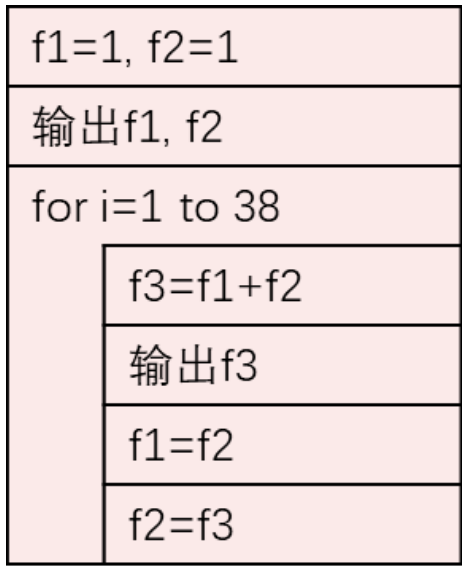
兔子繁殖的规律

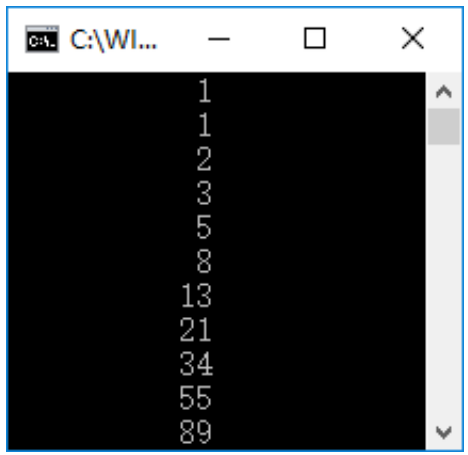
月数	小兔子对数	中兔子对数	老兔子对数	兔子总对数
1	1	0	0	1
2	0	1	0	1
3	1	0	1	2
4	1	1	1	3
5	2	1	2	5
6	3	2	3	8
7	5	3	5	13
⋮	⋮	⋮	⋮	⋮

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。

解法一: 利用递推 (迭代) 公式: $F_1 = F_2 = 1; F_3 = F_1 + F_2; F_1 = F_2; F_2 = F_3;$

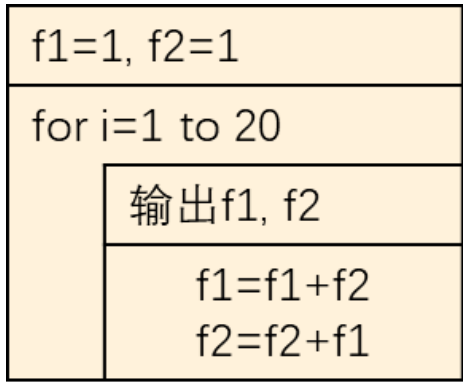
```
#include <stdio.h>
int main()
{
    int f1=1,f2=1,f3;
    int i;
    printf("%12d\n%12d\n",f1,f2);
    for(i=1; i<=38; i++)
    {
        f3=f1+f2;
        printf("%12d\n",f3);
        f1=f2;
        f2=f3;
    }
    return 0;
}
```

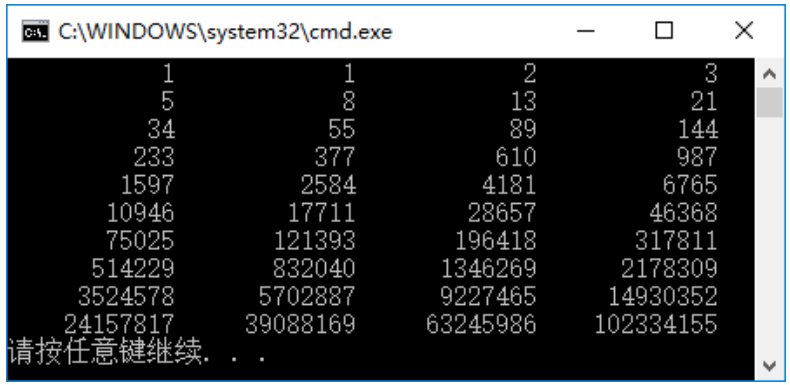




解法二: 利用递推 (迭代) 公式: $F_1 = F_2 = 1; F_1 = F_1 + F_2; F_2 = F_1 + F_2;$

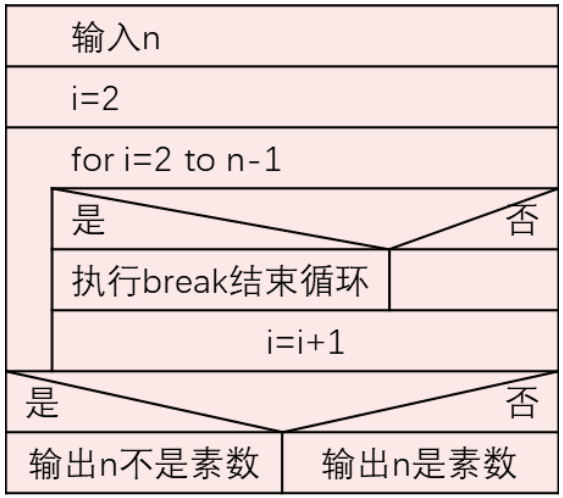
```
#include <stdio.h>
int main()
{
  int f1=1,f2=1;
  int i;
  for(i=1; i<=20; i++)
  {
    printf("%12d%12d",f1,f2);
    if(i%2==0) // 等效 if(!(i%2))
       printf("\n");
    f1=f1+f2;
    f2=f2+f1;
  }
  return 0;
}
```





[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) // for提前结束
    printf("%d is not a prime number.\n",n);
  else // for正常结束
    printf("%d is a prime number.\n",n);
  return 0;
}
```



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。这种判断循环结束的方法以后会常用到。

优化: n 不必被 $2 \sim (n-1)$ 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); // 自动转换为整数(不会四舍五入), 相当于k=(int)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;
}
```

使用标志变量,判断循环结束条件。

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k;i++)
  if(n%i==0) { flag=0; break; }
if(!flag) // for提前结束
  printf("%d is not a prime number.\n",n);
else // for正常结束
  printf("%d is a prime number.\n",n);
```

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k && flag;i++) // 比较与上面for的不同
  if(n%i==0) { flag=0; }
if(!flag)
  printf("%d is not a prime number.\n",n);
else
  printf("%d is a prime number.\n",n);
```

使用标志变量,判断循环结束条件。

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k;i++)
  if(n%i==0) { flag=0; break; }
if(!flag) // for提前结束
  printf("%d is not a prime number.\n",n);
else // for正常结束
  printf("%d is a prime number.\n",n);
```

```
int n,i,k,flag=1; // flag: 标志变量
k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
for (i=2;i<=k && flag;i++) // 比较与上面for的不同
  if(n%i==0) { flag=0; }
if(!flag)
  printf("%d is not a prime number.\n",n);
else
  printf("%d is a prime number.\n",n);
```

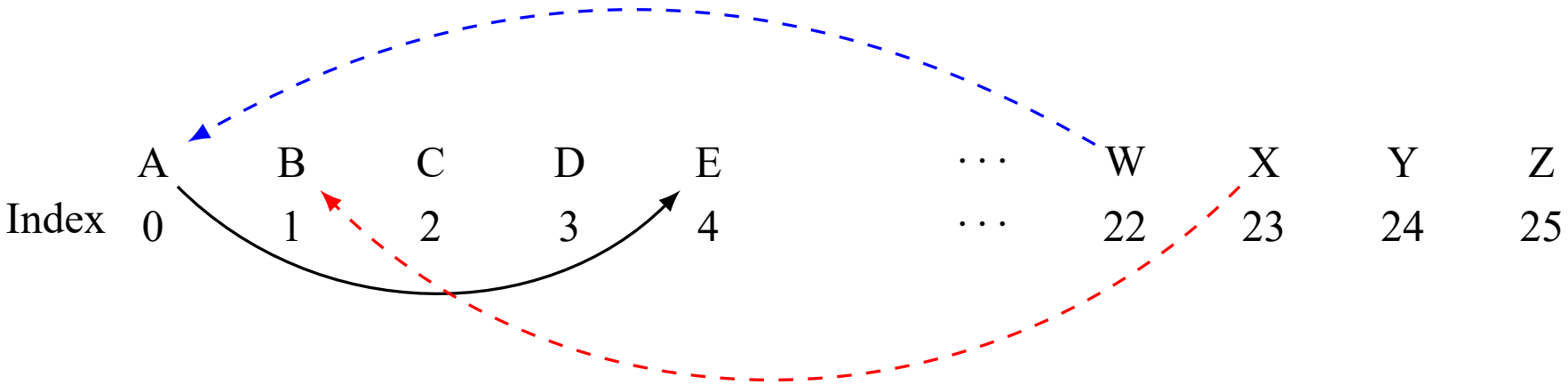
[例 5.10] 求 100 ～ 200 间的全部素数。

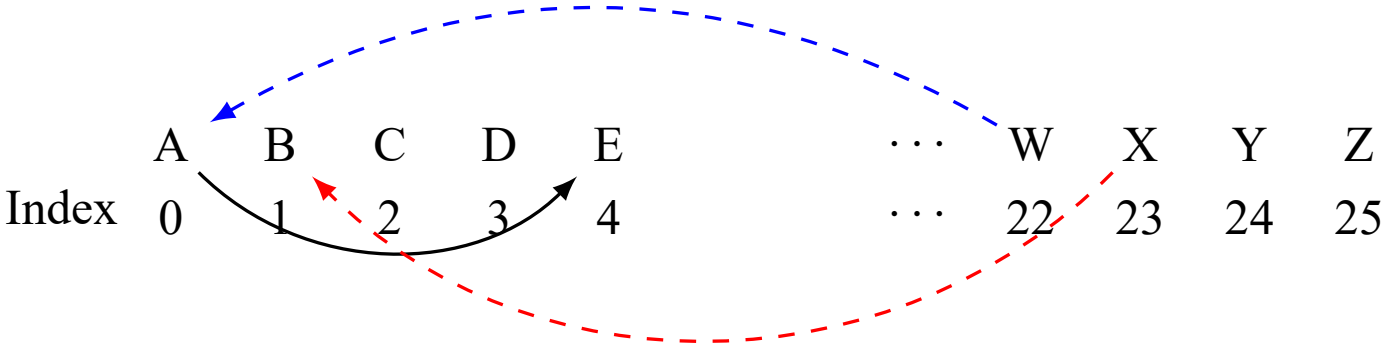
```
#include <stdio.h>
#include <math.h>
int main()
{
  int n,i,k;
  for (n=101;n<=200;n+=2) //n从101变化到200, 对每个奇数n进行判定
  {
    k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
    for(i=2;i<=k;i++)
      if(n%i==0) break;
    if(i>k)
      printf("%d ",n);
  }
  printf("\n");
  return 0;
}
```

[例 5.10] 求 100 ~ 200 间的全部素数。

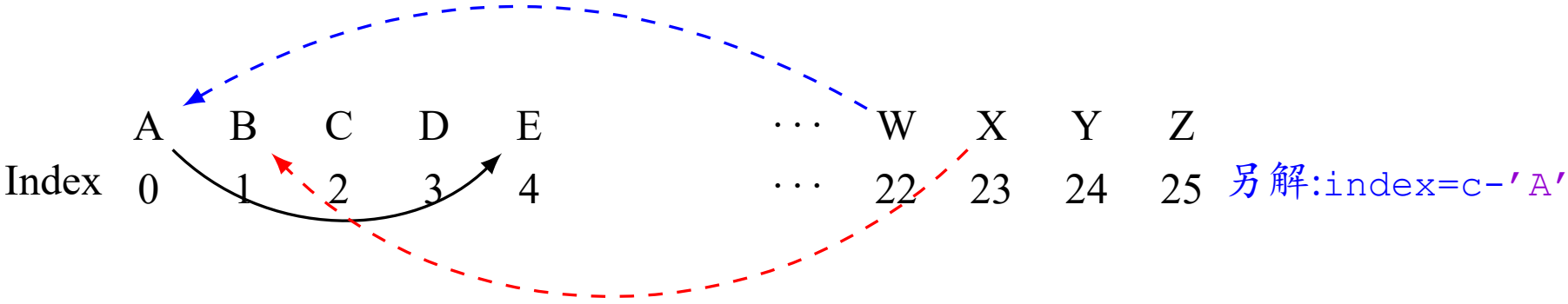
```
#include <stdio.h>
#include <math.h>
int main()
{
  int n,i,k;
  for (n=101;n<=200;n+=2) //n从101变化到200, 对每个奇数n进行判定
  {
     k=sqrt(n); // 自动转换为整数(不会四舍五入), 相当于k=(int)sqrt(n);
     for(i=2;i<=k;i++)
        if(n%i==0) break;
     if(i>k)
        printf("%d ",n);
  }
  printf("\n");
  return 0;
}
```

[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。






```
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");
```




```
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如果是大写字母
        c='A'+(c-'A'+4)%26;
    else if((c>='a' && c<='z')) //c如果是小写字母
        c='a'+(c-'a'+4)%26;
    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");
```

在循环条件中接收输入的字符是一种常见技巧

```
char c;
while((c=getchar())!='\n') //首先从键盘接收一个字符c，再检查c的值是否为换行
    符'\n'
{
   if((c>='A' && c<='Z')) //c如果是大写字母
      c='A'+(c-'A'+4)%26;
   else if((c>='a' && c<='z')) //c如果是小写字母
      c='a'+(c-'a'+4)%26;
   printf("%c",c); //输出已改变的字符
}
printf("\n");
```

附加题 1: 求 $s = a + aa + aaa + \cdots + a\cdots a$, 其中 a 是一个 1 ~ 9 的数字。例如 $a = 2, n = 4$ 时, $s = 2 + 22 + 222 + 2222$, a 和 n 由键盘输入。

```
int i,s,n,term = 0;
for(i=1,s=0; i<=n; i++) // 初始化循环变量用逗号隔开
{
  term = term*10 + a;
  s += term;
}
```

附加题 1: 求 $s = a + aa + aaa + \cdots + a\cdots a$, 其中 a 是一个 1 ~ 9 的数字。例如 $a = 2, n = 4$ 时, $s = 2 + 22 + 222 + 2222$, a 和 n 由键盘输入。

```
int i,s,n,term = 0;
for(i=1,s=0; i<=n; i++) // 初始化循环变量用逗号隔开
{
   term = term*10 + a;
   s += term;
}
```

附加题 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=1;
for(;;x++) // 循环体仅含if()结构, 看作一条语句, '{}'可省略
   if(x%5==1 && x%6==5 && x%7==4 && x%11==10)
   {
      printf("%d\n",x);
      break;
   }
```

附加题 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=1;
for(;;x++) // 循环体仅含if()结构, 看作一条语句, '{}'可省略
   if(x%5==1 && x%6==5 && x%7==4 && x%11==10)
   {
      printf("%d\n",x);
      break;
   }
```

附加题 3-1: 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。

解法一: 采用三重循环

```
int i,j,k; // 百、十、个位
for(i=1;i<=9;i++) // 百位
  for(j=0;j<=9;j++) // 十位
    for(k=0;k<=9;k++) // 个位
      if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
        printf("%d\n",i*100+j*10+k);
```

附加题 3-1: 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。

解法一: 采用三重循环

```
int i,j,k; // 百、十、个位
for(i=1;i<=9;i++) // 百位
  for(j=0;j<=9;j++) // 十位
    for(k=0;k<=9;k++) // 个位
      if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
        printf("%d\n",i*100+j*10+k);
```

附加题 3-2: 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。

解法二: 采用一重循环

```
int m,i,j,k;
for(m=100;m<=999;m++)
{
   i=m/100; j=m/10%10; k=m%10;
   if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
     printf("%d\n",i*100+j*10+k);
}
```

思考: 输出共有多少个水仙数?

附加题 3-2: 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。

解法二: 采用一重循环

```
int m,i,j,k;
for(m=100;m<=999;m++)
{
   i=m/100; j=m/10%10; k=m%10;
   if(i*100+j*10+k == i*i*i+j*j*j+k*k*k)
      printf("%d\n",i*100+j*10+k);
}
```

思考: 输出共有多少个水仙数?

附加题 3-3: 求整数区间 $[a,b]$ 中水仙花数的个数。

```
int n=0; //计数
int a,b; // a,b 区间
int i,t; // 循环变量, 代表a,b区间的每个数
int sum; // i的各位立方和
scanf("%d%d",&a,&b);
for(i=a;i<=b;i++) // 考察i是否水仙数
{
  sum = 0; t=i; // 临时变量记住i; 易遗漏每次内层循环前sum要归0
  while(t!=0) // 累加各位立方
  { sum=sum+(t%10)*(t%10)*(t%10); // 不好: sum+=pow(t%10,3);
    t=t/10;
  }
  if(sum==i) n++; // i是水仙数
}
printf("%d\n",n);
```

附加题 3-3: 求整数区间 $[a,b]$ 中水仙花数的个数。

```
int n=0; //计数
int a,b; // a,b 区间
int i,t; // 循环变量, 代表a,b区间的每个数
int sum; // i的各位立方和
scanf("%d%d",&a,&b);
for(i=a;i<=b;i++) // 考察i是否水仙数
{
  sum = 0; t=i; // 临时变量记住i; 易遗漏每次内层循环前sum要归0
  while(t!=0) // 累加各位立方
  { sum=sum+(t%10)*(t%10)*(t%10); // 不好: sum+=pow(t%10,3);
    t=t/10;
  }
  if(sum==i) n++; // i是水仙数
}
printf("%d\n",n);
```

附加题 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) //全部条件
        printf("%d,%d,%d\n",x,y,z);
```

如何考虑无解的情况?

附加题 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) //全部条件
         printf("%d,%d,%d\n",x,y,z);
```

如何考虑无解的情况?

附加题 5-1: 求整数 a,b 的最大公约数,当两个数中有一个为 0 时,公约数是不为 0 的那个整数; 当两个整数互质时最大公约数为 1。输入两个整数 a 和 b,求最大公约数。

```
int main() // 暴力循环求解, 效率低
{
   int a,b,t=-1,i; // t给初值是好习惯, 否则下面程序逻辑有可能使t得到随机值。
   scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
   if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
   if(b==0) t=a; // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
   else
   {
      for(i=b;i>0;i--)
      {
         if(a%i==0 && b%i==0)
         { t=i; break; } // 求得最大公约数, a,b互质, 必然t=1
      }
   }
   printf("%d\n",t);
   return 0;
}
```

附加题 5-1: 求整数 a,b 的最大公约数，当两个数中有一个为 0 时，公约数是不为 0 的那个整数；当两个整数互质时最大公约数为 1。输入两个整数 a 和 b，求最大公约数。

```
int main() // 暴力循环求解, 效率低
{
    int a,b,t=-1,i; // t给初值是好习惯, 否则下面程序逻辑有可能使t得到随机值。
    scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
    if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
    if(b==0) t=a; // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
    else
    {
        for(i=b;i>0;i--)
        {
            if(a%i==0 && b%i==0)
            { t=i; break; } // 求得最大公约数, a,b互质, 必然t=1
        }
    }
    printf("%d\n",t);
    return 0;
}
```

求整数 a, b 的最大公约数, 欧几里得算法

古希腊数学家欧几里德在其著作 «The Elements» 中最早描述了这种算法。

定理: 两个整数的最大公约数等于其中较小的那个数和两数相除余数的最大公约数。

令 $a = bq + r, q = \lfloor \frac{a}{b} \rfloor, r = a\%b$, 设 a, b 的公约数 $u, (a = su, b = tu)$, 我们有

$r = a - bq = su - qtu = (s - qt)u$, 表明 a, b 的公约数 u 也整除 r.

类似的, 设 b 和 r 的公约数 $v, (b = s'v, r = t'v)$, 则

$a = bq + r = s'vq + t'v = (s'q + t')v$, 表明 b 和 r 的公约数 v 也整除 a.

(1) 如果 $r = a\%b = 0$, 最大公约数为 b. (2) 如果 $r \neq 0$, 令 $a = b, b = r$, 返回到 (1).

重复迭代 (1),(2) 步, 直至余数为 0, 即求得最大公约数.

注: $\lfloor \frac{a}{b} \rfloor$ 表示 $\frac{a}{b}$ 的结果向下取整, 在 C 语言中, $a/b = \lfloor \frac{a}{b} \rfloor$

欧几里得算法 — 求整数 a, b 最大公约数: 迭代过程

$$
\begin{array}{lll}
q_1 = \lfloor \frac{a}{b} \rfloor & a = bq_1 + r_1 & r_1 = a - bq_1 \\
q_2 = \lfloor \frac{b}{r_1} \rfloor & b = q_2 r_1 + r_2 & r_2 = b - q_2 r_1 \\
q_3 = \lfloor \frac{r_1}{r_2} \rfloor & r_1 = q_3 r_2 + r_3 & r_3 = r_1 - q_3 r_2 \\
q_4 = \lfloor \frac{r_2}{r_3} \rfloor & r_2 = q_4 r_3 + r_4 & r_4 = r_2 - q_4 r_3 \\
\cdots & & \\
q_n = \lfloor \frac{r_{n-2}}{r_{n-1}} \rfloor & r_{n-2} = q_4 r_{n-1} + r_n & r_n = r_{n-2} - q_n r_{n-1} \\
q_{n+1} = \lfloor \frac{r_{n-1}}{r_n} \rfloor & r_{n-1} = q_{n+1} r_n + 0 & r_n = r_{n-1} / q_{n+1}
\end{array}
$$

迭代至 $q_{n+1}, r_{n-1} \% r_n = 0$, 得到 a, b 的最大公约数: $r_n = r_{n-1}/q_{n+1}$. 就是余数为 0 时的分母值.

欧几里得算法 — 求整数 a, b 最大公约数: 迭代过程 (例)

$a = 42, b = 30$

$$q_1 = \lfloor \frac{a}{b} \rfloor = \lfloor \frac{42}{30} \rfloor = 1 \qquad r_1 = 12$$

$$q_2 = \lfloor \frac{b}{r_1} \rfloor = \lfloor \frac{30}{12} \rfloor = 1 \qquad r_2 = 6$$

$$q_3 = \lfloor \frac{r_1}{r_2} \rfloor \lfloor \frac{12}{6} \rfloor = 2 \qquad r_3 = 0$$

Greatest Common Divisor of $(42, 30)$ is

$r_2 = 6$

$a = 144, b = 55$

$$r_1 = a\%b = 34 \qquad r_2 = b\%r_1 = 21$$

$$r_3 = r_1\%r_2 = 13 \qquad r_4 = r_2\%r_3 = 8$$

$$r_5 = r_3\%r_4 = 5 \qquad r_6 = r_4\%r_5 = 3$$

$$r_7 = r_5\%r_6 = 2 \qquad r_8 = r_6\%r_7 = 1$$

$$r_9 = r_7\%r_8 = 0$$

Greatest Common Divisor of $(144, 55)$ is

$r_8 = 1$

欧几里得算法 — 求整数 a, b 的最大公约数: 伪代码

定理: 两个整数的最大公约数等于其中较小的那个数和两数相除余数的最大公约数。

a(大),b(小)的最大公约数: 因为: a=qb+r, q=a/b; r=a%b, ⇒a,b的公约数能整除b和r. r=a%b,r为0, 则b就是最大公约数。否则迭代循环, a=b, b=r, 直到余数为零, 则分母就是最大公约数。

```
while(1)
{
   if(b==0) { gcd=a; break; } // 分母(上轮计算的余数)为0时, a就是最大公约数
   r = a%b; // 注意b为0时, 不能计算余数, a就是最大公约数
   if(r==0) { gcd=a; break; } // 本轮循环的a(上轮循环的b)就是最大公约数
   a=b; b=r; // 准备下一轮迭代
}
```

求整数 a, b 的最大公约数, 欧几里得算法, 参考代码 (1)

```
int a,b,r,t;
scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
while(1)
{
   if(b==0) { t=a; break; } // 分母(上轮计算的余数)为0时, a就是最大公约数
   r = a%b;
   if(r==0) {t=b; break;} // b就是最大公约数
   a=b; b=r; // 准备下一轮迭代
}
printf("%d\n",t);// 输出最大公约数
```

求整数 a, b 的最大公约数, 欧几里得算法, 参考代码 (2)

```
int main()
{
   int a,b,r,t;
   scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
   if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
   if (b==0) // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
   {
      printf("%d\n",a);
      return 0; // 主函数结束
   }
   while((r=a%b)!=0) // 去除了分母为0的情况
   {
      a=b; b=r; // 准备下一轮迭代
   }
   printf("%d\n",b);
   return 0; // 主函数结束
}
```

求整数 a, b 的最大公约数, 欧几里得算法, 参考代码 (3)

```
int main()
{
   int a,b,r,t;
   scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
   if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
   if (b==0) // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
   {
      printf("%d\n",a);
   }
   else
   {
      // 排除了分母为0时不能求余数的情况
      while((r=a%b)!=0) // a/b的余数赋值给r,r不等于0时执行循环体
      { a=b; b=r; }
      printf("%d\n",b);
   }
   return 0; // 主函数结束
}
```

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。