Computer programming

Lecture 4

Why do we need loop structure statements

In daily life or in the problems handled by programs, we often encounter problems that need to be addressed repeatedly
inputting the grades of 50 students in a class
calculating the average scores of multiple courses for these 50 students
finding the sum of 30 integers
checking if 30 students have passed their exams.

For example
there are 50 students in the class, and the average grades of each student in three courses are calculated

Input student 1's grades for three courses and calculate the average value before outputting

```
scanf("%f,%f,%f",&s1,&s2,&s3);
aver=(s1+s2+s3)/3;
printf("aver=%7.2f",aver);
```

Input student 2's grades for three courses and calculate the average value before outputting

```
scanf("%f,%f,%f",&s1,&s2,&s3);
aver=(s1+s2+s3)/3;
printf("aver=%7.2f",aver);
```

C language provides three types of loop structures

while loop, also known as a “when” loop, means that when the loop condition is true, an operation is repeated until it terminates when the loop condition is false.

do while loop, also known as a “until” loop, is a loop that first performs an operation and then determines whether the loop condition is true. If it is true, the operation is repeated until it terminates when the loop condition is false.

for loop is the most commonly used loop by programmers. Compared to while and do while, the for loop includes the starting point, loop conditions, and loop variation patterns in the statement structure, which can reduce the probability of omissions.

[Question]: Write a program to write the following statement

```
I am the No. 1 student
I am the No. 2 student
I am the No. 3 student
I am the No. 4 student
I am the No. 5 student
I am the No. 6 student
I am the No. 7 student
I am the No. 8 student
I am the No. 9 student
I am the No. 10 student
```

```
#include “stdio.h”
main()
{  printf(“I am the No. 1 student \n”);
printf(“I am the No. 2 student \n”);
printf(“I am the No. 3 student \n”);
printf(“I am the No. 4 student \n”);
printf(“I am the No. 5 student \n”);
printf(“I am the No. 6 student \n”);
printf(“I am the No. 7 student \n”);
printf(“I am the No. 8 student \n”);
printf(“I am the No. 9 student \n”);
printf(“I am the No. 10 student \n”);
}
```

[Question]: Write a program to write the following statement

```
I am the No. 1 student
I am the No. 2 student
I am the No. 3 student
I am the No. 4 student
I am the No. 5 student
I am the No. 6 student
I am the No. 7 student
I am the No. 8 student
I am the No. 9 student
I am the No. 10 student
```

```
#include “stdio.h”
main()
{
int i;
for(i=1;i<=10;i++)
    printf(“I am the No. %d student
\n”,i);   //Repetitive statements
}
```

[Question]: Write a program to write the following statement (infinite loop)

```
#include “stdio.h”
main()
{
int i;
for(i=1;i<=10;i--)
    printf(“I am the No. %d student
\n”,i);   //Repetitive statements
}
```

The initial value of i is 1, and every time the value of i is subtracted by 1, the loop condition of i<=10 is always satisfied, becoming an endless infinite loop

[Question] Enter the grade of a class and calculate the average grade of the entire class.

Analysis:
1.Confirming is a recurring problem.
2.Set the value of the loop control variable i.
3.Determine the loop body.
4.Finally, calculate the average score.

The steps for solving the problem are as follows:
1.Enter a score.When "score>=0",
2.do the following work.
① Accumulated total score;
② Number of people plus one;
③ Enter the next score.
3. Repeat step (2) until "score<0".

The general form of the while statement is as follows：

while (expression) statement

Loop condition expression

Execute loop body statement when 'true'
Do not execute when 'false'

The characteristics of the while loop are:First judge the conditional expression, then execute the loop body statement

```
#include “stdio.h”
main()
{
    int sum, score, n;
    float ave;
    sum=0; n=0;
    scanf(%d, &score);
    While (score>=0)
    {
          sum+=score;
           n++;
      }
   if (n!=0)
    {
        ave=sum/n;
      printf(“average score=%f”, ave ) ;    }
}
```

[Question]: Write a program to calculate 1+2+3+…+100

Solution ideas:

- **This is an accumulation problem, which requires adding up 100 numbers in sequence**
- **To repeat 100 addition operations, a loop can be used to implement**
- **The latter number is obtained by adding 1 to the previous number**
- **After adding the previous number i, adding 1 to i yields the next number**

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

Cannot be omitted

Compound statement

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

Cannot lose, otherwise the loop will never end

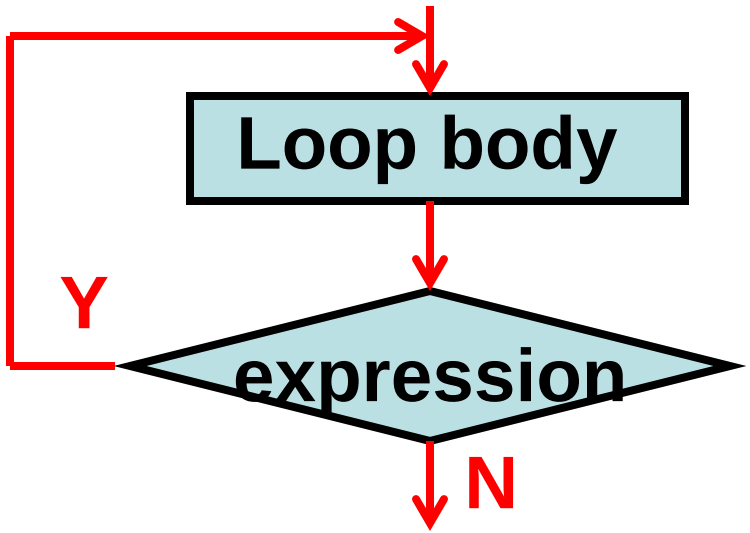
```
sum=5050
```

Using the do -- while statement to implement a loop

The characteristic of the do while statement is to unconditionally execute the loop body first, and then determine whether the loop condition is valid

The general form of the do while statement is:

```
Do
    Statement
While (expression);
```

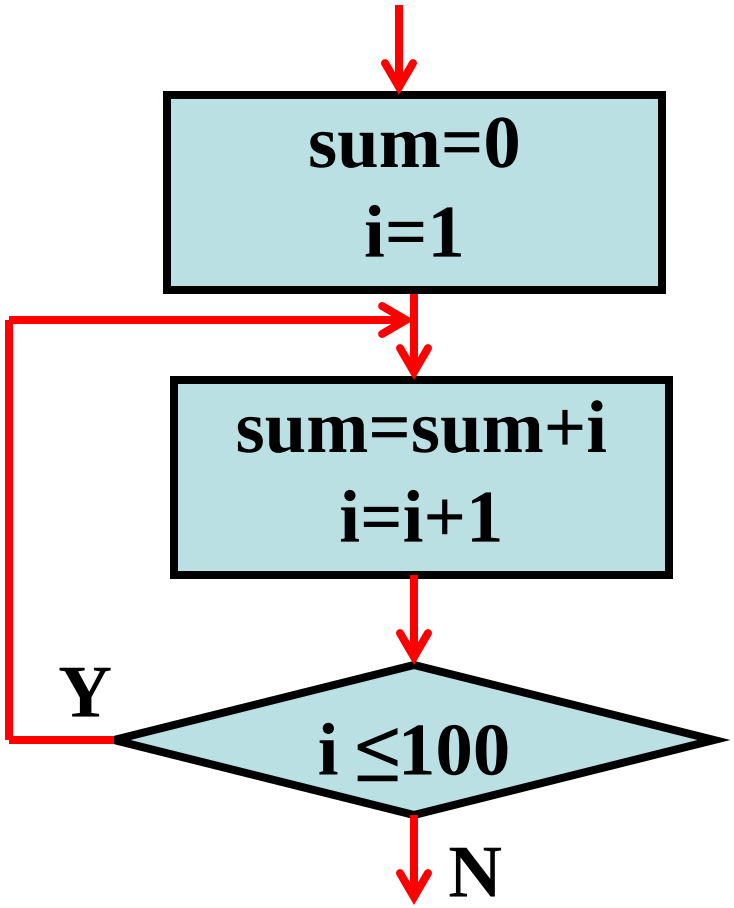


[Question] Enter the grade of a class and calculate the average grade of the entire class.

```
#include “stdio.h”
main()
{
    int sum, score, n;
    float ave;
    sum=0; n=0;
    scanf(%d, &score);
      do
    {
          sum+=score; n++;
     }
     While (score>=0);

   if (n!=0)
    {
        ave=sum/n;
       printf(“average score=%f”, ave ) ;    }
}
```

[Question]: Write a program to calculate 1+2+3+…+100 using do while statement.




```
i=1;   sum=0;
do
 {
    sum=sum+i;
    i++;
 }while(i<=100);
```

Comparison of while and do - while loops

When the first value of the expression after 'while' is' true ', the results obtained by both loops are the same; Otherwise, it's different

```
in
pr
sc
wh
{
  sum=sum+i;
  i++;
}
printf("sum=%d\n",sum);
```

```
i=?1
sum=55
```

```
i=?11
sum=0
```

```
{
   sum=sum+i;
   i++;
}while(i<=10);
printf("sum=%d\n",sum);
```

```
i=?1
sum=55
```

```
i=?11
sum=11
```

Implementing a loop with a for statement

The general form of a for statement is

For (expression 1; expression 2; expression 3)

statement

Set the initial conditions and execute them only once. Can set initial values for zero, one, or more variables for execution

Implementing a loop with a for statement

The general form of a for statement is

For (expression 1; expression 2; expression 3)

statement

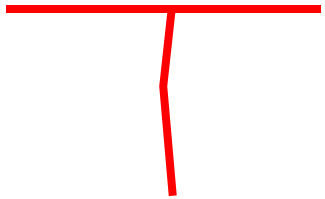
The loop condition expression is used to determine whether to continue the loop. Execute this expression before each execution of the loop body to determine whether to continue executing the loop

Implementing a loop with a for statement

The general form of a for statement is

For (expression 1; expression 2; expression 3)

statement



As a loop regulator, for example, to increment a loop variable, it is done after executing the loop body

Implementing a loop with a for statement

The execution process of the for statement:

(1) Solve expression 1 first

(2) Solve expression 2, if its value is true, execute the loop body, and then execute step (3) below. If false, end the loop and go to step (5)

(3) Solving Expression 3

(4) Return to step (2) above to continue execution

(5) At the end of the loop, execute a statement below the for statement

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

Using the for statement is simpler and more convenient

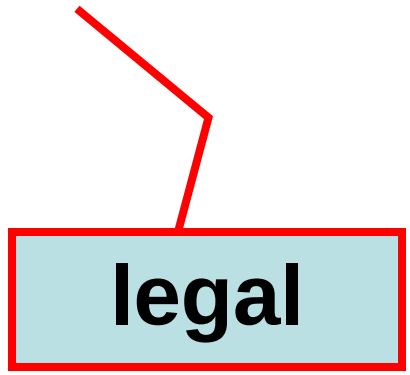
For (expression 1; expression 2; expression 3)

statement

One or two or three expressions can be omitted

```
for(i=0; (c=getchar())!='\n'; i+=c)
        printf("%c",  c)  ;

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



Nested Loop

The nesting of a loop

a loop contains another complete loop structure within its body

Nested loops can also be nested within nested loops, which is called multi-layer loops

Three types of loops (while loop, do... while loop, and for loop) can be nested within each other

Read the following program and write the output result

```
#include<stdio.h>
main()
{ int  i,j;
 for(i=1;i<=3;i++)
 {
      for(j=1;j<=4;j++)

          {   if(i==j)
            printf("i*j=%d\n",i*j);
          }
  }
}
```

output result

```
1*1=1
2*2=4
3*3=9
```

[example]Output n × N characters' * '

Analysis:

The output of n lines' * 'can be controlled by the following loop.

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

(2) The output of n '*' can be implemented using the following loop statements;

```
for (j=1; j<=n; j++)
        putchar('*');
```

```
#include <stdio.h>
 main()
{  int i, j, n;
   scanf("%d",&n);
```

```
 for (i=1; i<=n; i++)
 {    for (j=1; j<=n; j++)
              { putchar('*');}
      putchar('\n');
}
}
```

'break' statement

syntax format： break；

[Question]: Write a program to calculate 1+2+3+…+100

```
#include <stdio.h>
void main()
{  int s=0,i=1;
   for(; ;)
     {s=s+i;
      i++;
      if(i>100)
          break;}
    printf("s=%d\n",s);
}
```

In this program, when i>100, use the break statement to forcibly terminate the for loop and continue executing the next statement after the for statement.

‘continue' statement

General form: `continue;`

[example]Output All odd numbers between 1 and 100.

```
#include <stdio.h>
main()
{   int i,count=0;
    for(i=1;i<=100;i++)
        {   if(i%2==0)
            continue;
             printf("%4d",i);
         }
}
```

The difference between Continue and Break statements

1. **The continue statement only ends the current iteration and controls the transition to the next loop. If the loop conditions are still met, the loop will continue**
2. **Break is to end the entire loop and control the execution of the next statement that jumps to the loop statement**
3. **The continue statement can only be used within a loop statement, meaning it is a statement within a loop. The break statement can be used in a switch statement or loop statement, and usually appears as an embedded statement of an if statement.**
4. **When using a break statement, it is necessary to pay attention to the situation of loop nesting. If the break statement appears in the loop body of a double or multiple nested loop statement, the break statement can only terminate the innermost loop it is in, and cannot terminate the execution of multiple loops.**

[example]Output 4 rows and 5 columns of numbers

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

- **Solution ideas:**
- **You can use nested loops to address this issue**
- **Using an outer loop to output a row of data**
- **Using an inner loop to output a column of data**
- **Output in matrix format (5 data per row)**

```
#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”);
           printf ("%d\t",i*j);
       }
    printf("\n");
    return 0;
}
```

Accumulated number of output data

Control the output of 5 data in a row

```
#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”);
           printf ("%d\t",i*j);
      }
    printf("\n");
    return 0;
}
```

double circulation

```
#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”);
          printf ("%d\t",i*j);
      }
    printf("\n");
    return 0;
}
```

Control
output 4 lines

```
#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”);
            printf ("%d\t",i*j);
        }
     printf("\n");
     return 0;
}
```

Control the output of 5 data in each row

```
#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”);
           printf ("%d\t",i*j);
       }
    printf("\n");
    return 0;
}
```

When i=1

J changed from 1 to 5

The value of i * j is 1,2,3,4,5

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

```
#include
int main
{  int i,j,n=0;
    for (i=1;i<=4;i++)
       for (j=1;j<=5;i++,
       {  if (n%5==
            printf ("%
       }
    printf("\n");
    return 0;
}
```

When i=2

How to modify the program without outputting the first blank line?

J changed from 1 to 5

The value of i * j is 2,4,6,8,10

```
1       2       3       4       5
2       4       6       8       10

4       8       12      16      20
```

```
#include <std
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”);
             if (i==3 && j==1) break;
           printf ("%d\t",i*j);
       }
    printf("\n");
    return 0;
}
```

Encountered row 3, column 1, terminating inner loop

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

The original 3rd row, 1st data 3, was not output

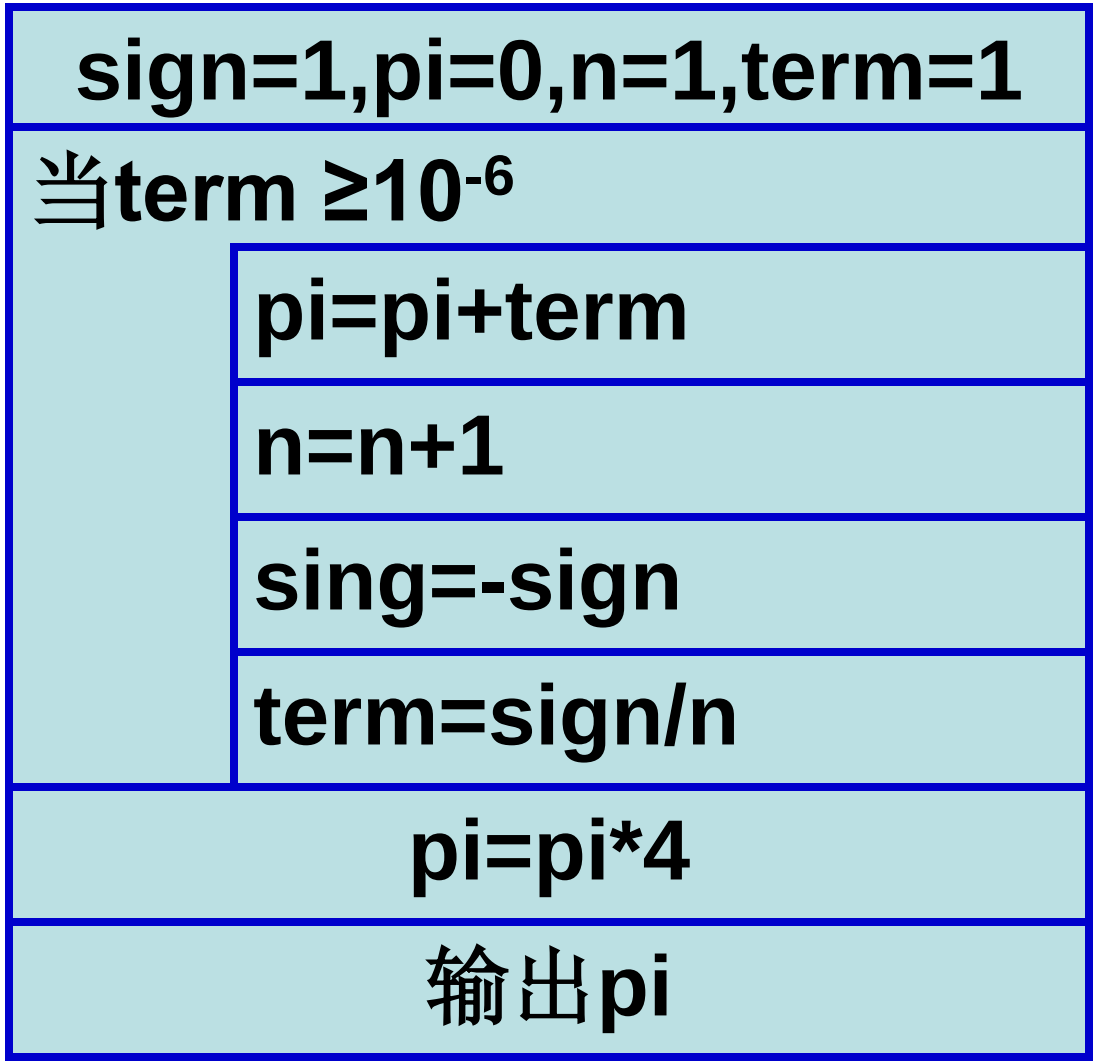
```
#include <s
int main()
{  int i,j,n=0;
        {  if (n%5==0) printf (“\n”);
            if (i==3 && j==1) continue;
           printf ("%d\t",i*j);
        }
    printf("\n");
    return 0;
}
```

[Question]: write a program to approximate the value of pi

$$\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

- **The numerator of each term is 1**
- **The denominator of the latter term is the denominator of the previous term plus 2**
- **The symbol of item 1 is positive, and starting from item 2, the symbol of each item is opposite to the symbol of the previous item**

$$\frac{1}{n} \quad \Rightarrow \quad -\frac{1}{n+2}$$


sign=1,pi=0,n=1,term=1
当term ≥10-6
pi=pi+term
n=n+1
sing=-sign
term=sign/n
pi=pi*4
输出pi

```
#include <stdio.h>
#include <math.h>
int main()
{  int sign=1; double pi=0,n=1,term=1;
   while(fabs(term)>=1e-6)
   {  pi=pi+term;
      n=n+2;
      sign=-sign;
      term=sign/n;
    }
   pi=pi*4;
   printf("pi=%10.8f\n",pi);
   return 0;
}
```

A function for finding absolute values

Only ensure that the first 5 decimal places are accurate

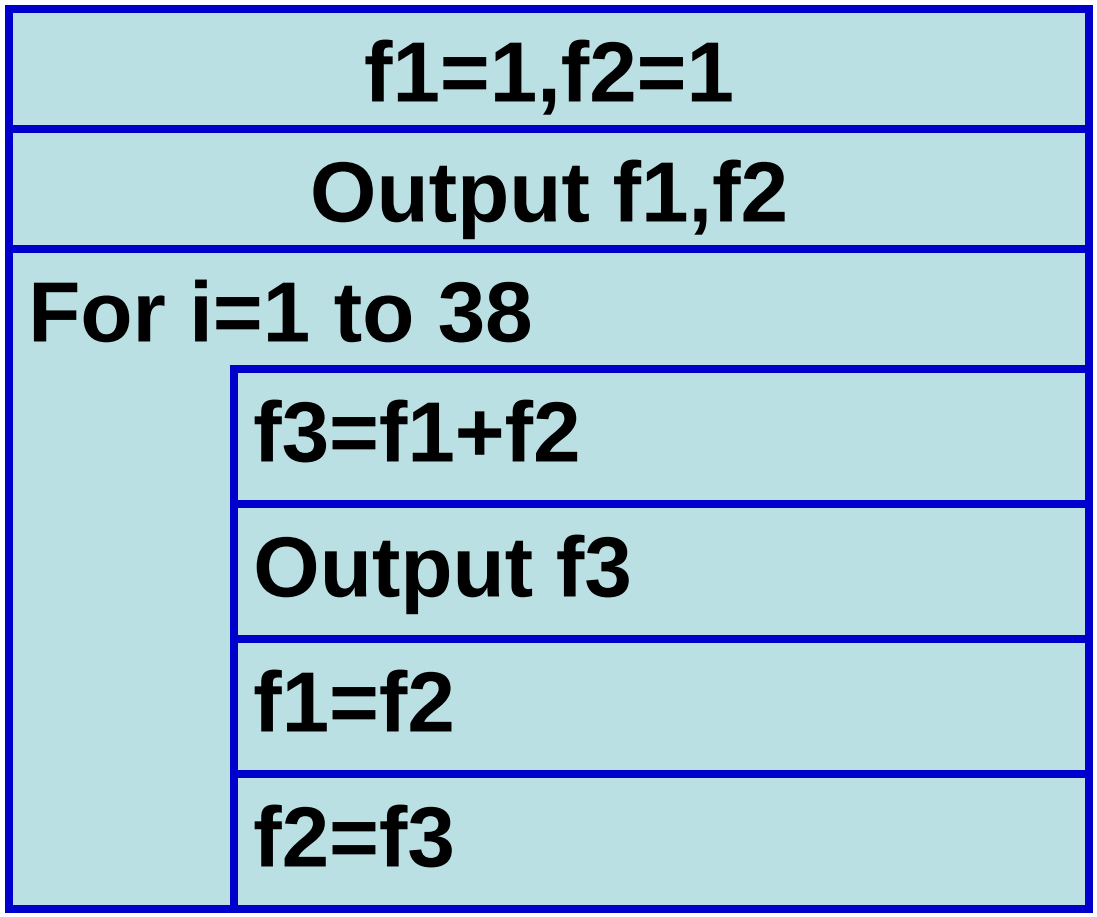
```
#include <stdio.h>
#include <math.h>
int main()
{  int sign=1; double pi=0,n=1,term=1;
   while(fabs(term)>=1e-6)
   {  pi=pi+term;
      n=n+2;
      sign=-sign;
      term=sign/n;
    }
   pi=pi*4;
   printf("pi=%10.8f\n",pi);
   return 0;
}
```

change to 1e-8

```
pi=3.14159065
```

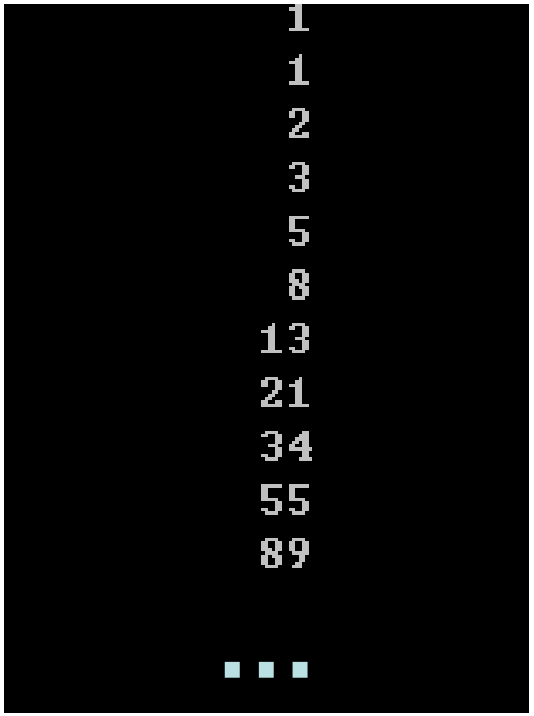
```
pi=3.14159263
```

[Question] Find the first 40 numbers in the Fibonacci sequence. This sequence has the following characteristics: the first and second numbers are 1 and 1. Starting from the third number, this number is the sum of its first two numbers. Namely:


f1=1,f2=1
Output f1,f2
For i=1 to 38
f3=f1+f2
Output f3
f1=f2
f2=f3

```
#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;
}
```

Code can be improved



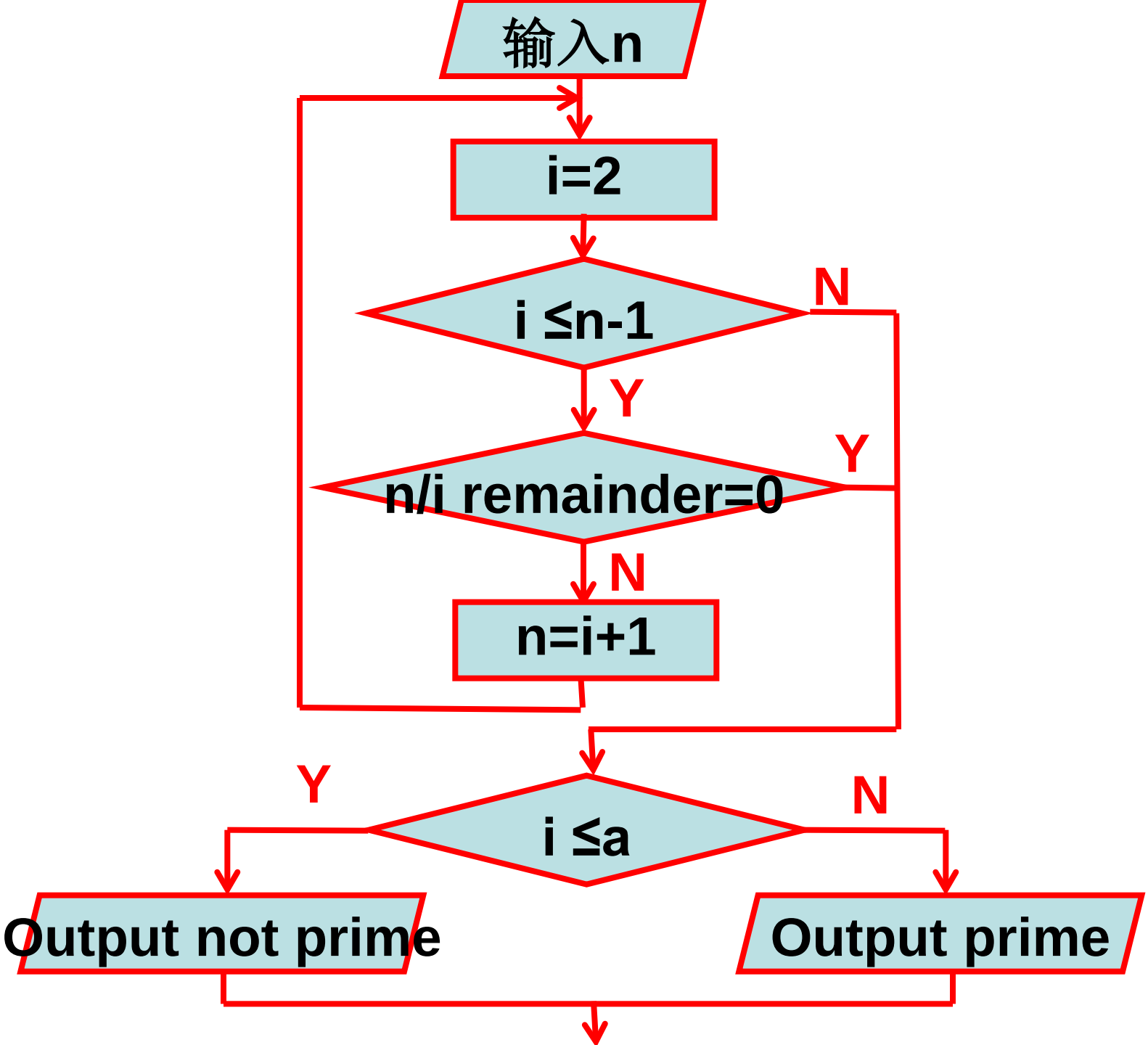
```
#include <stdio.h>
int main()
 {  int f1=1,f2=1;  int i;
     for(i=1; i<=20; i++)
    { printf("%12d %12d ",f1,f2);
       if(i%2==0) printf("\n");
```

```
           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
```

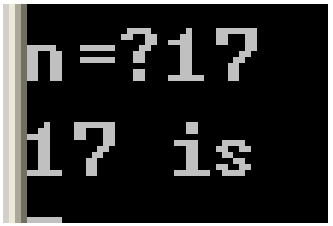
[Question] Enter an integer n greater than 3 to determine whether it is a prime number.

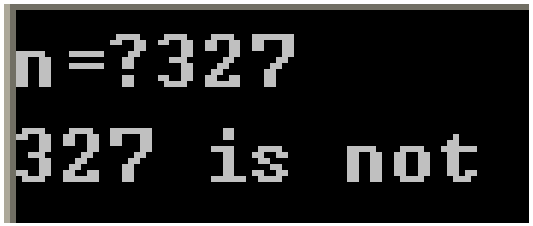
Solution ideas:

- **Dividing n by i (changing the value of i from 2 to n-1)**
- **If n can be divided by any integer from 2 to (n-1), it means that n is definitely not a prime number and there is no need to continue dividing by subsequent integers.**
- **Therefore, the loop can be terminated early**
- **Note: At this point, the value of i must be less than n**


输入n
i=2
i ≤n-1
N
Y
n/i remainder=0
Y
N
n=i+1
Y
i ≤a
N
Output not prime
Output prime

```
#include <stdio.h>
int main()
 { int n,i;
    printf(“n=?");  scanf("%d",&n);
    for (i=2;i<=n-1;i++)
        if(n%i==0) break;
    if(i<n) printf("%d is not\n",n);
    else printf("%d is\n",n);
    return 0;
}
```





```
#include <stdio.h>
int main()
 { int n,i;
    printf("n=?");  scanf("%d",&n);
    for (i=2;i<=n-1;i++)
        if(n%i==0) break;
    if(i<n) printf("%d is not\n",n);
    else printf("%d is\n",n);
    return 0;
}
```

k=sqrt(n);

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

#include <math.h>

k=sqrt(n);

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

#include <math.h>

k=sqrt(n);