
GrimScythe2001

1. What are the different types of loops present in C language?

→ There are three types of loops:

i) for loop

ii) while loop

iii) do...while loop

2. Why loops are required in C programming? Explain with an example

→ In order to perform or execute same set of statements while a given condition is true, loops are required and used.

for example:

```
for (i=0; i<=100; ++i)
        printf("%d ", i);
//prints numbers 0 to 100.
```

3. Describe how the syntax of various loops used in C programming are different from each other

→ The for loop enables programmers to perform numbers of steps together in one line as it contains initialisation expression, test expression and update expression all in one line. In for loop, number of iterations is known beforehand.

while loop is used in situations where we do not know the exact number of iterations of loop beforehand.

The main difference between do while and while loop is that in do while loop, the condition is tested at the end of the body, ie. even if the condition is false, the body executes at least once.

4. Ex

Syntax of for loop;

```
for( initialisation ; test condition; update)
{
    // body
}
```

Syntax of while loop;

```
// initialisation_expression;
while ( test_expression)
{
    // body
    // update_expression;
}
```

Syntax of do... while loop;

```
// initialisation_expression;
do
{
    // body
    // update_expression;
} while (test_expression);
```

4) Explain the basic difference between while and do...while loop.

→ The only difference between while and do...while is that the test condition is tested at the end of the body in do...while loop, whereas it is tested at the beginning in case of while loop.

5) How 'for' loop is different from while or do...while loop.

→ In case of 'for' loop, the number of iterations should be known beforehand, which is not the case with while or do...while loop.

6) State some of the applications of loop statements in C programming

→ Some of the applications of loop statements in C programming are:

i) To print set of numbers in sequence or progression.

ii) To input information into the database system (with structures / classes) of number of people.

iii) To sort an array or database information based on some conditions.

iv) To calculate factorial.

v) To work with dimensional arrays like matrix.

7. What do you understand by the terms while(1) and while(0)? Is while(2) also a valid statement? Explain.

→ while(1) means that the loop is always true and hence the loop will execute infinitely. while(0) means that the loop is false and hence the loop will terminate immediately. Yes, while(2) is a valid statement. Any number other than '0' is true and hence while(2) will also execute infinitely.

8. Can while(1) or while(0) be represented in 'for' loop also? Explain your answer.

→ Yes, every loop can be expressed or represented in another form. while(0) & while(1) can be represented in for loop in following manner:

```
for( ; 0 ; );
for( ; 1 ; );
```

If there is no initialisation expression or updated expression, the space is left blank and the test expression is written as it is.

9. Generally, loop statements are called iterative statements. If a program uses iterative statements, is there any other way without using iterative manner to solve the problem? If yes, how, and if no, why?

→ Yes, loops or iterative statements can be solved using an alternative method of recursion. Every iterative program can be performed using recursion and vice-versa.

10. What do you mean by nested loop? Explain with example

→ The loop under another loop, i.e. using loop as a body of a main loop is known as nested loop. for example:

```
for (i=0; i<5; ++i)
    for (j=i; j<5; ++j)
        printf("Nested loop");
```

11. How many times 'Hello' will be printed when the following statement is executed:

```
while(printf("Hello"));
```

How the above statement can be changed to print 'Hello' exactly once?

→ while (printf("Hello")); will be printed infinitely. In order to print it exactly once, the following changes should be made:

```
while(!printf("Hello"));
```

12. What is the difference in output of the following two statements?

(i) for(; printf("Hello");); (ii) for(; ;) printf("Hello");

→ In (i), printf("Hello") has been used as the test condition while in (ii), printf("Hello") has been used as the body of the for loop. There is, however, no difference in the output. ~~Both the~~ Both the statements will print 'Hello' infinite time as output.

13) Explain break and continue in loops with example

→ break; statement ends the loop and exits out of that loop, however, continue; statement skips the remaining statements within that loop to continue with the same loop again ~~with~~ after performing increment/decrement and testing test condition.

For example:

```
for(i=0; i<5; ++i)          //break statement
{
    if (i=3)
        break;
    printf("%d",i) }
```

Output: 0 1 2

```
for (i= 0; i<5; ++i)        //continue statement
{
    if (i=3)
        continue;
    printf("%d", i);
}
```

Output 0 1 2 4

14) The statement for(;;); can be broken by which statement?

→ The statement for(;;); can be broken by break; statement.

15) Which statement is the correct syntax to run two 'for' loop simultaneously?

(a) for(i=0, i<n; ++i) for(j=0; j<n; j+=5)

(b) for (i=0; j=0; i<n; j<n; i++; j+=5)

(c) for(i=0; i<n; i++) {} for(j=0; j<n; j+=5) {}

(d) None of the mentioned

→ (d) None of the mentioned

16) ~~for~~ Which 'for' loop has range of similar indexes of 'i' used in for(i=0; i<n; ++i)?

(a) for(i=n; i>0; --i)

(b) for(i=n; i>=0; --i)

(c) for(i=n-1; i>0; i--)

(d) for(i=n-1; i>-1, i--)

→ (d) for(i=n-1; i>-1; i--)

17) Which of the following cannot be used as LHS of the expression in "for(exp1; exp2; exp3)"

(a) Variable (c) typedef

(b) Function (d) macros

→ (d) macros

18) Which loop is most suitable to first perform the operation and then test the condition?

(a) for loop (b) while loop

(c) do..while loop (d) none of the mentioned.

→ (c) do..while loop.

19) Write a short note on Arrays

→ An array is a container in the memory location that holds a fixed number of values of a single type. It is basically a collection of items stored at contiguous memory location and the main idea is to store multiple items of same type together.

20) What will happen if in a C program, you assign a value to an array element whose subscript exceeds the size of array?

→ If the index of the array is exceeded, the program will crash.

21) What is the difference between Strings and Arrays?

→ The main difference between the two is that arrays can store values of any data type of any length, while strings are usually ASCII characters that are terminated with a null character '\0'.

22) Write the advantages and disadvantages of Array.

→ ADVANTAGES OF ARRAYS:

1. Arrays represent multiple data items of the same type using a single name.
2. Elements in array can be accessed randomly using the index number.
3. Arrays allocate memory in contiguous memory locations for all its elements.
4. Using arrays, other data structures like linked lists, stacks, queues, trees, graphs etc can be implemented.

LIMITATIONS OF ARRAYS:

1. The number of elements to be stored in an array should be known beforehand.
2. Insertion and deletion are quite difficult in an array as the elements are stored in ~~consequitive~~ consecutive memory locations, and the shifting operation is costly.
3. Allocating more memory than the requirement leads to wastage of memory space and less allocation of memory also leads to a problem.

23) Explain one dimension array with an example.

→ A one-dimensional array is a type of linear array. Accessing its elements involves a single subscript which can either represent a row or a column index.

For example:

```
int array-name[10];
```

24) Explain two dimensional array with an example

→ An array of arrays is known as two dimensional array, which is also commonly known as matrix due to its large use for this specific task. A two dimensional array can be represented as a table of rows and columns.

For e.g: `int array-name[3][4];`

where,

int is datatype
array-name is variable
[3] is number of rows
[4] is number of columns

25) Explain applications of array.

→ Some of the applications of array are:

i) Array stores data elements of the same data type

ii) Arrays can be used for CPU scheduling.

iii) Arrays are used to implement other data structures like Stacks, Heaps, Queues, etc.

iv) Arrays can be used to sort data elements.

v) Maintains multiple variables using a single name, which is used to maintain large data records.

vi) Arrays can be used to perform matrix operations.

26) Consider the following declaration of a 'two-dimensional' array in C: char a[100][100], Assuming that the main memory is byte-addressable and that the array is stored starting from memory address 0, the address of a[40][50] is?

→ Here,

a[0] has taken 100 byte space

So, till a[39], memory size taken is,

$100 \times 40 = 4000$

and, a[40], till index value 50 is 51 spaces

So,

Address of a[40][50] is $4000 + 51$

$= 4051$

Since the address starts from 0,

Address of a[40][50] $= 4051 - 1$

$= 4050$

ALTERNATIVELY:

Row-major form:

Here, $R = 100$; $C = 100$ // in a[100][100]

size, $s = 1$

base address, $B = 0$

So,

$a[40][50] = B + s(C * r + c)$

$= 0 + 1(100 * 40 + 50)$

$= 4000 + 50$

∴ Address of $a[40][50] = 4050$

27) Which of the following statements mentioning the name of the array begins DOES NOT yield the base address?

(a) When array name is used with sizeof operator.

(b) When array name is operand of the & operator.

(c) When array name is passed to scanf() function.

(d) When array name is passed to printf() function.

→ (a) When array name is used with sizeof operator.

(b) When array name is operand of the & operator.

* ERROR: Predict the error (syntactical as well as logical (if any) of the following C codes and also find the output after correction (if any).

28) →
```
# include <stdio.h>
int main() {
    int i;
    for (i=1; i<=5; ++i) {
        printf ("%d", i);
        if (i == 3)
            break;
    }
    return 0; }
```
// Output: 1 2 3

29) →
```
# include <stdio.h>
int main() {
    short i;
    for ( i = 1; i >0 ; --i)
        printf ("%d\n", i);
    return 0;
}
```
// Output: 1

30) →
```
# include <stdio.h>
void main() {
    int i=0, j=0;
    for ( i ; i<5 ; ++i) {
        printf ("C Programming \n");
    }
}
```
// Second for loop was not logical as it's not required

// Output: C Programming
C Programming
C Programming
C Programming
C Programming

31)→
```
# include <stdio.h>
void main()
{
    double k;
    for (k=6.0; k>3.0; --k);
    printf("%lf", k);
}
```
// Output: 3.000000

32)→
```
# include <stdio.h>
int main() {
    for (int i=5; i>=2; --i)
        printf("Hello\n");

 return 0;
}
```
// Output: Hello
Hello
Hello
Hello

33)→
```
# include <stdio.h>
int main() {
    static int i=0;
    for(i++; ++i; i++) {
        printf("%d", i);
        if(i==6)
            break;
    }
    return 0;
}
// Output: 2 4 6
```

34)→
```
# include <stdio.h>
int main() {
    int i = 1024;
    for(i; i>>=1; )
        printf("C Programming");
    return 0;
}
// Output: C ProgrammingC ProgrammingC programing
C ProgrammingC ProgrammingC programming
C ProgrammingC ProgrammingC programming
C Programming
```

35)→
```
# include <stdio.h>
int main() {
    for(int n = 9; n!=0 && n>0; n--)
        printf("n = %d", n--);
    return 0;
// Output: n = 9 n = 7 n = 5 n = 3 n = 1
```

36) →
```
#include <stdio.h>
int main() {
    int i = -5;
    while (i<=0) {
        ++i;
        printf("C Programming");
    }
    return 0;
}
// Output: C Programming C Programming C Programming
           C Programming C Programming C Programming
```

37) →
```
#include <stdio.h>
int main() {
    int i = 0;
    for(i=0 ; i<20 ; ++i) {
        switch(i) {
            case 0: i+=5; break;
            case 1: i+=2; break;
            case 5: i+=5; break;
            default: i+=4; break;
        }
    printf("%d", i);
    }
    return 0;
}
// Output: 5 10 15 20
```

38)→
```
# include <stdio.h>
int main() {
        int i;
        for(i=1 ; i<=10 ; i+=2)
                printf(" C Programming ");
        return 0;
    }
// Output : C Programming C Programming C Progra
            mming C Programming C Programming
```

39)→
```
# include <stdio.h>
int main()
 {
     int c=5, no=10;
     do {
        no/=c;
        c--;
     }while(c);
     printf("%dn",no);
     return0;
  }
// Output: 0n
```

40)→
```
# include <stdio.h>
int main(){
     int  i=3;
     while(i--)
      {
        int i=100;
        i--;
        printf("%d",i);
      }
```

```
    return 0;
}
// Output:  gg gg gg
```

41) →

```
#include<stdio.h>
int main(){
    int i=1, j;
    for( ; ; )
    {
        if(i)
            j=--i;
        if(j<10)
            printf("C Programming", j++);
        else
            break;
    }
    return 0;
}
```

// Output: C Programming C Programming C Programming C Programming C Programming C Programming C Programming

42)→
```
#include <stdio.h>
int main()
{
    int j=0;
    for( ; j<10; ){
            printf("C ",j++);
            printf("Programming");
    }
    return 0;
}
```
// Output: CProgrammingCProgrammingCProgrammingCProgra
mmingCProgrammingCProgrammingCProgrammingCProgramming
CProgrammingCProgrammingCProgramming.

43)→
```
#include <stdio.h>
int main(){
    int i;
    for(i=0; i<10; i++){
        printf("C Programming");
        break;
    }
    return 0;
}
```
// Output: C Programming.

44)→
```
#include<stdio.h>
int main() {
    unsigned int i=65000;
    printf("%d",i);
    return 0;
}
//Output: 65000;
```

45)→
```
#include<stdio.h>
int main() {
    for(int i=0; i<5; ++i) {
        int i=10;
        printf("%d",i);
        i++;
    }
    return 0;
}
//Output 10101010101
```

* OUTPUT: Find the output of the following C Code

46)
```
#include <stdio.h>
int main()
{
    static int i;
    for(i++ ; ++i ; i++) {
        printf("%d", i);
        if (i==6)
            break;
    }
    return 0;
}
```
→ 2 4 6

47)
```
#include <stdio.h>
int main() {
    for(5; 2; 2)
        printf("Hello");
    return 0;
}
```
→ HelloHelloHelloHelloHello ... prints infinite time.

48)
```
#include <stdio.h>
int main() {
    int i, j;
    for (i=1, j=1 ; i<=3, j<=3 ; i++, j++)
        printf("%d %d", i, j);
    return 0;
}
```
→ 1 1 2 2 3 3

49)
```
#include <stdio.h>
int main() {
    int i = 1, j = 1;
    for( ; j; printf("%d %d ", i, j))
        j = i++ <= 1;
    return 0;
}
```

→ 2 1 3 0

50)
```
#include <stdio.h>
int i;
int main() {
    int i;
    for( t=4; scanf("%d", &i)-t; printf("%d\n", i))
        printf("%d--", t--);
    return 0;
}
```

→
```
10
4--10
12
3--12
8
2--8
14
```

Q1)

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

→ -128

Q2)

```
#include <stdio.h>
int main() {
    unsigned char i = 0;
    for ( ; i <= 0 ; i++),
        printf ("%d\n", i);
    return 0;
}
```

→ 1

Q3)

```
#include <stdio.h>
int main () {
    int i;
    for (i=0; i>9 ; i+=3) {
        printf("for ");
    }
    return 0;
}
```

→ // No output.

54) #include <stdio.h>
int main() {
int a[5] = {5, 1, 15, 20, 25};
int i, j, m;
i = ++a[1];
j = a[1]++;
m = a[i++];
printf("%d, %d, %d", i, j, m);
return 0;
}

→ 2, 2, 15

55) #include <stdio.h>
int main() {
int arr[5], i=0;
while (i<5)
arr[i] = ++i;
for (i=0; i<5; i++)
printf("%d, ", arr[i]);
return 0;
}

→ 1, 2, 3, 4, 5,

56) #include <stdio.h>
int main()
{ int arr[1] = {10};
printf("%d\n", 0[arr]);
return 0;
}

→ 10

57.
```
#include <stdio.h>
int main() {
    float arr[] = {12.4, 2.3, 4.5, 6.7};
    printf("%d\n", sizeof(arr)/sizeof(arr[0]));
    return 0;
}
```
→ 4

58.
```
#include <stdio.h>
int main() {
// if the array begins at address 65486
int arr[] = {12, 14, 15, 23, 45};
printf("%u, %u\n", arr, &arr);
 return 0;
}
```
→ 65486, 65486

59.
```
#include <stdio.h>
int main() {
    float arr[] = {12.4, 2.3, 4.5, 6.7};
    printf("%d\n", sizeof(arr)/sizeof(arr[0]));
    return 0;
}
```
→ 4

60.
```
#include <stdio.h>
int main() {
    int a[1][2][3] = {0};
    a[0][1][2] = 5;
    printf("%d", *(*(*(a+0)+1)+2));
    return 0; }
```
→ 5

61.
```
#include <stdio.h>
int main() {
    int arr[5][5][5] = {0};
    printf("%d", (&arr+1 - &arr));
    return 0;
}
```
→ 1

62.
```
#include <stdio.h>
int () {
    int rows = 3, columns = 4, i, j, k;
    int a[3][4] = {1, 2, 3, 5, 7};
    i = j = k = 0;
    for(i = 0; i < rows; i++)
        for(j=0; j<columns; j++)
            if( a[k][j] < k)
                k = a[i][j];
    printf("%d\n", k);
    return 0;
}
```
→ 0

63.
```
#include <stdio.h>
int main() {
int  i = 0;
printf ("Hello");
char s[4] = {'\b', '\t', '\r', '\n'};
for (i = 0 ; i<4 ; ++i) {
    printf("%c", s[i];
}
return 0;
```
→ Hell

64.
```
#include <stdio.h>
int main() {
    int a, b, c;
    int arr[5] = {1, 2, 3, 25, 7};
    a = ++arr[1];
    b = arr[1]++;
    c = arr[2++];
    printf("%d--%d--%d", a, b, c);
    return 0;
}
```
→ 4--3--25

65.
```
#include <stdio.h>
int main() {
    int rows = 3, columns = 4, i, j, k;
    int a[3][4] = {23, 46, 69, 102, 99, 109};
    i = j = k = 99;
    for(i = 0; i > rows; ++i)
        for(j = 0; j > columns; j++)
            if (a[k][j] > k)
                k = a[i][j];
    printf("%d\n", k);
    return 0;
}
```
→ 99

66.
```
# include <stdio.h>
int main() {
    int x = 1;
    short int i = 2;
    float f = 3;
    if (sizeof((x==2)?f:i) == sizeof(float))
        printf("float\n");
    else if (sizeof((x==2)?f:i) == sizeof(short int))
        printf("short int\n");
}
```
→ float

67.
```
# include <stdio.h>
void main() {
    int x=2;
    if (x--, --x, x)
        printf("Tom");
    else
        printf("Jerry");
}
```
→ Jerry

68.
```
# include <stdio.h>
int main() {
 char s[4] = {'\0', '\0', '\0', '\0'};
 for (i=0; i<4; i++) {
  printf("%c ", s[i]);
 }
  return 0;
}
```
→

// No output

69.
```
#include <stdio.h>
int main() {
    static int var[5];
    int count = 0;
    var[++count] = ++count;
    for(count = 0; count < 5; count++)
        printf("%d", var[count]);
    return 0;
}
```

→ 0 0 2 0 0

* PROGRAMS: Write all programs using loops.

70. Write a C Program to print the following pattern:

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

// continues

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

→

```
#include <stdio.h>
void main()
{
    int i, j;
    for(i=0; i<9 ; ++i)
    {
        if (i<9/2)
        {
            for(j=0; j<(9/2)-i ; ++j)
                printf(
```

69.
```
#include <stdio.h>
int main() {
    static int var[5];
    int char = 0;
    var[++count] = ++count;
    for( count = 0; count < 5; count++)
        printf("%d", var[count]);
    return 0;
}
```

→ 0 0 2 0 0

* PROGRAMS: Write all programs using loops

70. Write a C program to print the following pattern.

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

→
```
#include <stdio.h>
int main()
{
    int i, j;
    for(i=0; i<9; ++i)
    {
        if(i<9/2)
            for(j=0; j<(2*i+1); ++j)
                printf("* ");
        else
            for(j=0; j<((9-i)*2-1); ++j)
                printf("* ");
        printf("\n");
    }
    return 0;
}
```

71) Write a program to print the following pattern.

```
1
2 3 2
3 4 5 4 3
4 5 6 7 6 5 4
3 4 5 4 3
2 3 2
1
```

→

```
#include <stdio.h>
void main()
{
    int i, j, a;
    for(i=0; i<7; ++i)
    {
        if (i<7/2)
        {
            for(j=0, a=i+1; j<2*i+1; ++j)
                if(j<=(2*i+1)/2)
                {
                    printf("%d ", a++);
                    if (j==(2*i+1)/2)
                        --a;
                }
                else
                    printf("%d", --a);
        }
        else
            for(j=0, a=7-i; j<(7-i)*2-1; ++j)
            {
                if(j<=((7-i)*2-1)/2)
                {
                    printf("%d", a++);
                    if(j==((7-i)*2-1)/2)
                        --a;
                }
                else
                    printf("%d", --a);
            }
        printf("\n");
```

```
    }
}
```

72. Write a C program to print all the alphabet from a to z (using while loop).

→
```
#include <stdio.h>
void main()
{
    char i = 'a';
    while ( i != 'z' + 1)
        printf("%c ", i++);
}
```

73. Write a C program to check if a number is perfect number or not.

→
```
#include <stdio.h>
void main()
{
    int num, sum=0, i;
    printf(" Enter number : ");
    scanf("%d ", &num);
    for (i=1; i<=num/2)
        if ( num % i == 0)
            sum = sum + i;
    if ( sum == num)
        printf(" Perfect Number ");
```

```
    else
        printf(" Not a perfect number ");
}
```

74. Write a C program to print all Armstrong numbers between 1 to n.

→

```
#include <stdio.h>
#include <math.h>
int main()
{
    int i, num, temp, exp, mod, sum;
    printf(" Enter limit: ");
    scanf(" %d ", &num);
    printf(" Armstrong Numbers are: ");
    for(i=1; i<=num; ++i)
    {
        temp = i; exp = 0; sum = 0;
        while (temp != 0)
        {
            temp = temp/10;
            ++exp;
        }
        temp = i;
        while (temp != 0)
        {
            mod = temp % 10;
            sum = sum + pow(mod, exp);
            temp = temp/10;
        }
        if (sum == i)
            printf(" %d ", i);
```

```
    }
    return 0;
}
```

7. Write a C program to separate odd and even integers in separate arrays.

→

```
#include <stdio.h>
void main()
{
    int num[], odd[], even[], i=0, j, k,
    printf(" Enter elements in your array... ");
    printf(" Enter 0 to terminate... ");
    do
    {
        scanf(" %d ", &num[i++])
    } while ( num[i-1] != 0)
    for ( i=0, j=0, k=0; num[i] != 0, ++i)
    {
        if ( num[i] % 2 == 0)
            even[j++] = num[i];
        else
            odd[k++] = num[i];
    }
    even[j] = 0 , odd[j] = 0;
    printf(" Odd and Even numbers separated ");
    printf("\n Odd Numbers and Even Numbers :\n");
    for( j=0; even[j] != 0 ; ++j)
        printf(" %d, ", even[j]);
    for( k=0, odd[k] != 0 ; ++k)
        printf(" %d, ", odd[k]);
}
```

76. Write a C program to delete an element at desired position from an Array.

→

```
#include <stdio.h>
void main()
{
    int arr[' ' ], size, pos, i,
    printf(" Enter size of array: "),
    scanf("%d", &size),
    printf(" Enter elements to your array\n"
    for(i=0, i<size ; ++i)
        scanf("%d ", &arr[i]);
    printf(" Enter location you wish to delete: "
    scanf("%d ", &pos);
    printf(" You chose to delete %d from location
        %d ", arr[pos-1], pos),
    for(i = pos-1 ; i<size-1, ++i)
        arr[i] = arr[i+1],
    arr[i] = NULL,
    printf("\nNew array: "),
    for (i=0 ; arr[i]' = NULL ; ++i)
        printf("%d ", arr[i]);
}
```

77. Write a C program to find sum of rows of a Matrix.

→

```
#include <stdio.h>
void main()
{
    int mat[][], row, col, sum;
    printf("Enter row: ");
    scanf("%d", &row);
    printf("Enter column: ");
    scanf("%d", &column);
    printf("Enter elements to your Matrix: ");
    for(i=0; i<row; ++i)
        for(j=0; j<column; ++j)
            scanf("%d", &mat[i][j]);
    for(i=0; i<row; ++i)
    {
        sum = 0;
        for(j=0; j<col; ++j)
            sum += mat[i][j];
        printf("Sum of row %d is %d", i+1, sum);
    }
}
```

78. Write a program to enter / accept a matrix and determine whether it is a sparse matrix

→

```
#include <stdio.h>
void main()
{
    int mat[][], row, col, count=0
    Printf(" Enter row and column: "),
    scanf("%d %d", &row, &col),
    printf(" Enter elements in your matrix"
    for(i=0; i<row; ++i)
        for(j=0; j<col; ++j)
            scanf("%d", &mat[i][j]
    for(i=0; i<row; ++i)
        for(j=0; j<col; ++j)
        {
            if(mat[i][j]==0)
                ++count;
        }
    if( count > (row*col)/2)
        printf(" It is sparse matrix "),
    else
        printf(" It is not a sparse matrix"),
}
```

79. Write a C program to accept two matrices and check whether they are equal.

→
```
#include <stdio.h>
void main()
{
    int mat1[3][3], mat2[3][3], brk = 1
    printf("Enter elements in matrix first: \n");
    for(i=0; i<3; ++i)
        for(j=0, j<3; ++j)
            scanf("%d", &mat1[i][j]);
    printf("Enter elements in matrix second: \n");
    for(i=0, i<3; ++i)
        for(j=0; j<3; ++j)
            scanf("%d", &mat2[i][j]);
    for(i=0; i<3; ++i)
    {
        for(j=0; j<3; ++j)
        {
            if(mat1[i][j] != mat2[i][j])
            {
                brk = 0;
                break;
            }
        }
        if('brk)
            break;
    }
    if(brk)
        printf("Matrices are equal");
    else printf("Matrices are not equal");
}
```