ABES Engineering College, Ghaziabad.

Affiliated to Dr. A.P.J Abdul Kalam Technical University, Lucknow.

Department of CSE-Data Science

Title	**Lecture Notes**
Subject	**Programming for Problem Solving**
Topics	**Loop**
Lecture Date	**May 2024**
Faculty Name	**Mr. Dilip Kr. Bharti**
Section	**First Year Section-B**

UNIT-03.0

3.1 Iteration and Loops

3.1.1 Introduction

Looping is the concept to implement iteration with some specified condition. It is to execute a set of statements (line of codes) repeatedly according to the condition given in the loop. It means it execute the same code multiple times. So, Loop is combination of condition and iteration (Repetition).

3.1.2 Types of looping statement

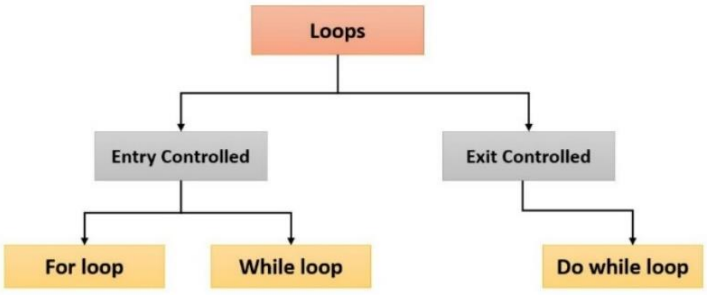


Fig. Types of loops

1. **Entry Controlled loop:** In this type of loop the test condition is tested before entering the loop body. for Loop and while loop are entry-controlled loops.

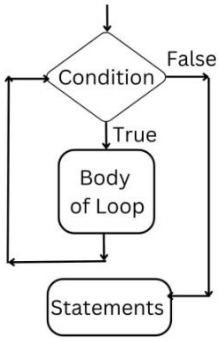


Fig. Entry controlled loop

2. **Exit Controlled Loop:** In this type of loop the test condition is tested or evaluated at the end of loop body.The loop body will execute at least once, irrespective of whether the condition is true or false. do while loop is exit-controlled loop.

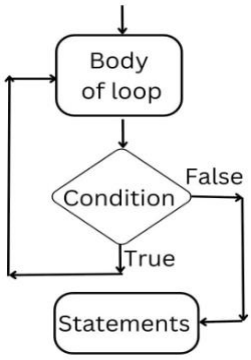


Fig. Exit controlled loop

Entry Controlled loop

for loop:

for loop is usually known as definite loop because the programmer knows exactly how many times the loop will repeat.

for allows us to specify three things about a loop in a single line:

(a) Setting a loop counter to an initial value.

(b) Testing the loop counter to determine whether its value has reached the number of repetitions desired.

(c) Increasing /decreasing the value of loop counter each time the body of the loop has been executed.

Syntax :

```
for (initialization; test expression ; update expression)
 {
//for loop body
```

Statement1;

Statement 2;

}

Flow Chart:

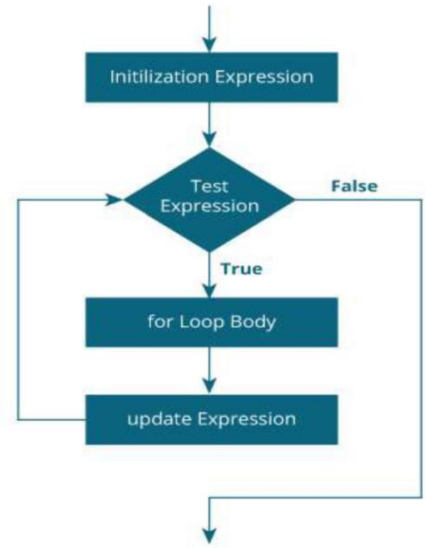


Fig: Flow chart of for loop

Working of for loop:

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of for loop are executed, and the update expression is updated.
- Again, the test expression is evaluated.
- This process goes on until the test expression is false.
- **When the test expression is false, the loop terminates.**

// Program to calculate the sum of first n natural numbers using for loop
// Positive integers 1,2,3...n are known as natural numbers

#include <stdio.h>

```
int main()
{
   int num, count, sum = 0;

printf("Enter a positive integer: ");
scanf("%d", &num);

   // for loop terminates when num is less than count
for(count = 1; count <= num; ++count)
   {
      sum += count;
   }

printf("Sum = %d", sum);

   return 0;
}
```

while loop:

while loop allows a part of the code to be executed multiple times depending upon a given Boolean condition. It can be viewed as a repeating if statement. **The while loop is mostly used in the case where the number of iterations is not known in advance.** It is also called a pre-tested loop.

Syntax :

```
initialisation;

while(testExpression)

{

// statements inside the body of the loop

Increment/decrement;

}
```

Flow Chart:

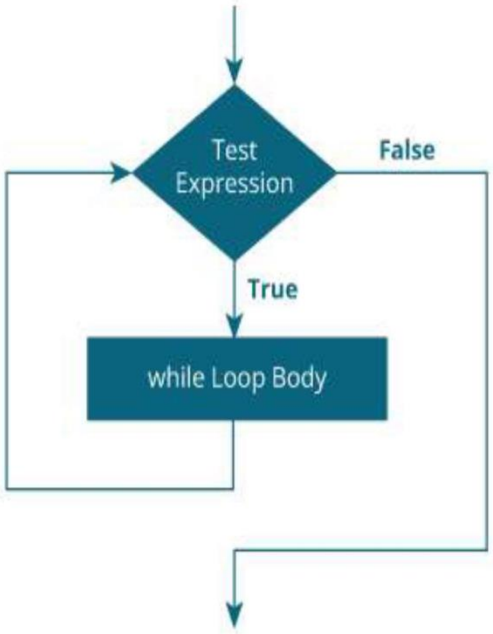


Fig: Flow chart of while loop

Working of while loop:

- The while loop evaluates the test expression inside the parenthesis ().
- If the test expression is true, statements inside the body of while loop are executed. Then, the test expression is evaluated again.
- The process goes on until the test expression is evaluated to false.
- If the test expression is false, the loop terminates (ends).
- **It is also known as entry controlled loops.**

//program to find the sum of digits of a given number using while loop

```
#include <stdio.h>
int main()
{
int n, t, sum = 0, r;
printf("Enter an integer\n");
scanf("%d", &n);
t = n;
while (t != 0)
{
r= t % 10;
```

```
sum = sum + r;
t  = t / 10;
}
printf("Sum of digits of %d = %d\n", n, sum);
  return 0;
}
```

Output:

Enter an integer 12345

Sum of digits of 12345 = 15

Exit Controlled loop

do while loop:

Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop in C programming language checks its condition at the bottom of the loop. A do...while loop is similar to a while loop, **except that a do...while loop is guaranteed to execute at least one time.**

Syntax :

```
do
{
Statement1;
Statement 2;
} while ( condition );
```

Flow Chart:

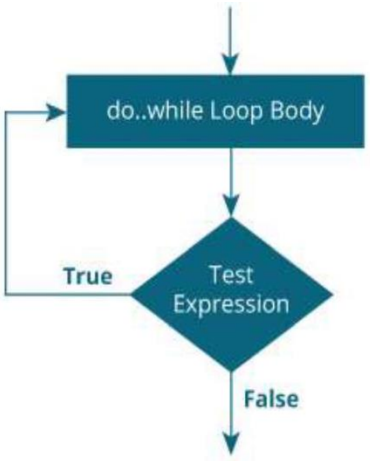


Fig: Flow chart of do-while loop

Working of do while loop:

- The body of do...while loop is executed. After execution of body, the test expression is evaluated.
- If the test expression is true, the body of the loop is executed again and the test expression is evaluated.
- This process goes on until the test expression becomes false.
- If the test expression is false, the loop ends.

// Program to find the factorial of a number using do while loop

```
#include<stdio.h>
void main()
{
   int n,i=1,f=1;
printf("\n Enter The Number:");
scanf("%d",&n);
    do
   {
      f=f*i;
i++;
}while(i<=n);
printf("\n The Factorial of %d is %d",n,f);
   }
```

Differences between while and do while loop

while loop	**do while loop**
In this loop the test condition or criteria is checked and evaluated first before loop starts executing.	In this loop the test condition or criteria is checked after the loop is executed.
while loop is entry-controlled loop.	do-while loop is exit- controlled loop.
Loop conditions are not terminated with semicolon.	Loop conditions are terminated with semicolon.

Body of loop never executed if the condition is false. It means body of the loop executed zero time, if the condition is false.	Body of loop is executed for at least one time even after the condition is false.
If there is single statement in body of loop, brackets are not required.	Brackets are always required.
Variable in condition is initialized before the execution of loop.	variable may be initialized before or within the loop.
Syntax: while (condition) { Block of statements; }	Syntax: do { Block of statements; } while(condition);
//Write a program using while loop	// write a program using do while loop

Programs on Loops

//Program: 1

//Write a program to check the number is palindrome of not.

// An integer is a palindrome if the reverse of that number is equal to the original number.

```
#include <stdio.h>
int main()
 {
  int n, reversed = 0, remainder, original;
printf("Enter an integer: ");
scanf("%d", &n);
   original = n;
// reversed integer is stored in reversed variable
   while (n != 0)
{
      remainder = n % 10;
      reversed = reversed * 10 + remainder;
      n /= 10;
```

```
    }
    // palindrome if original and reversed are equal
    if (original == reversed)
printf("%d is a palindrome.", original);
    else
printf("%d is not a palindrome.", original);
    return 0;
}
```

Output:

```
Enter an integer: 1001
1001 is a palindrome.
```

//Program : 2

// Write a program to print check a number is prime number or not

// A prime number is a positive integer that is divisible only by 1 and itself. For example: 2, 3, 5, 7, 11, 13, 17.

```
#include <stdio.h>
int main()
 {
  int n, i, flag = 0;
printf("Enter a positive integer: ");
scanf("%d", &n);
   if (n == 0 || n == 1)
    flag = 1;
  for (i = 2; i<= n / 2; ++i)
{
      if (n % i == 0) {
     flag = 1;
     break;
```

```
   }
  }
  if (flag == 0)
printf("%d is a prime number.", n);
  else
printf("%d is not a prime number.", n);
  return 0;
}
```

Output:

```
Enter a positive integer: 29
29 is a prime number.
```

OR

// Write a program to print check a number is prime number or not

```
#include <stdio.h>
int main()
 {
  int n, i, count= 0;
printf("Enter a positive integer: ");
scanf("%d", &n);

  for (i = 1; i<= n ; ++i)
{
      if (n % i == 0) {
     count ++;
   }
  }
  if (count == 2)
```

```
printf("%d is a prime number.", n);
  else
printf("%d is not a prime number.", n);
  return 0;
}
```

// Program: 3

// Write a C program to add first seven terms of the following series using for loop.

1/1! +2/2! +3/3! +--------

```
#include<stdio.h>
int main()
{
   int i;
   float fact=1.0, res, n_res=0;
   for (i=1;i<8;i++)
   {
      //find factorial for 1 to 7
      fact = fact * i;
      //find the i/factof(i)
      res = i/fact;
      //add all the results
n_res = n_res + res;
}
printf("%f", n_res);
}
```

Output:

2.718056

// Program: 4

// Write a program to input a floating-point number and find leftmost digit of integral part of a number

```
#include<stdio.h>
#include<math.h>
main()
{
float num;
int ip,rm,count=0,temp;
printf("Enter a floating point number \n");
scanf("%f",&num);
ip=(int)num;
temp=ip;
while(temp>0)
{
temp=temp/10;
count++;
}
rm=ip/pow(10,count-1);
printf("Left most digit of integral part=%d \n",rm);
}
```

OR

// Write a program to input a floating-point number and find leftmost digit of integral part of a number

```
#include<stdio.h>
#include<math.h>
main()
{
float num;
```

```
int n,r;
printf("Enter a floating point number \n");
scanf("%f",&num);
n=(int)num;
printf("%d",n);
while(n>0)
{
r=n%10;
n=n/10;
}
printf("Left most digit of integral part=%d \n",r);
}
```

Nested Loops

A loop inside another loop is called a nested loop. **Loop containing another loop is called outer loop and the loop that is contained inside outer loop is called inner loop.**

Syntax of Nested loop

```
Outer_loop
{
Inner_loop
{
// inner loop statements.
}
// outer loop statements.
}
```

Types of Nested Loops

1. **Nested while loop**
2. **Nested do-while loop**
3. **Nested for loop**

Note: There can be mixed type of nested loop i.e. for loop inside while loop, or while loop inside do-while loop.

1. **Nested while loop**

A while loop inside another while loop is called nested while loop.

Syntax of Nested while loop

```
while (condition1)
{
   statement(s);
   while (condition2)
   {
      statement(s);
      ... ... ...
   }
   ... ... ...
}
```

2. **Nested do-while loop**

A do-while loop inside another do-while loop is called nested do-while loop.

Syntax of Nested do-while loop

```
do
{
   statement(s);
   do
   {
      statement(s);
……
}while (condition2);
……
}while (condition1);
```

3. **Nested for loop**

A for loop inside another for loop is called nested for loop.

Syntax of Nested for loop

```
for (initialization; condition; updation)
{
   statement(s);
   for (initialization; condition; updation)
```

```
   {
      statement(s);
      ... ... ...
   }
   ... ... ...
}
```

Program on nested loops

//Program 1

// Program to print pattern

12345

1234

123

12

1

```
#include <stdio.h>
intmain()
{
   int i, j;
   for(i=5;i>=1;i--)
   {
      for(j=1;j<=i;j++)
      {
printf("%d",j);
      }
printf("\n");
   }
return 0;
}
```

//Program 2

// Write a program to print the pattern:

****

***

**

*

```
#include <stdio.h>
int main() {
  int i, j;
  for (i = 4; i>= 1; --i) {
    for (j = 1; j <= i; ++j) {
     printf("* ");
    }
printf("\n");
  }
  return 0;
}
```

//Program 3

// Write a program to print the pattern:

2 3 4 5 6 7

3 4 5 6 7

4 5 6 7

5 6 7

6 7

7

```
#include <stdio.h>
int main()
```

```
 {
   int i, j;
   for (i = 2; i<=7; i++)
{
     for (j = i; j <= 7; ++j)
{
printf("%d ", j);
     }
printf("\n");
   }
   return 0;
}
```

//Program 4

//Write a program to find the Armstrong number from 1 to 100.

```
#include <stdio.h>
#include <math.h>
int main()
{
   int num, onum, r, result = 0, n = 0;
printf("Armstrong numbers between 1 and 100 are: \n");
   for (num = 1; num<= 100; ++num) {
onum = num;
       // Find the number of digits in the current number
       while (onum != 0)
{
```

```
onum /= 10;
        ++n;
    }
onum = num;
    // Calculate the Armstrong number
    while (onum != 0) {
        r = onum % 10;
        result += pow(r, n);
onum /= 10;
    }
    // Check if the number is Armstrong
    if (result == num) {
printf("%d ", num);
    }
    // Reset variables for next iteration
    n = 0;
    result = 0;
  }
  return 0;
}
```

//Program 5

//Write a program in C to print following pattern with appropriate comments:

```
10
 9 8
7 6 5
4 3 2 1
```

```
#include <stdio.h>
 int main()
```

```
{
  int i,j,k=10;
   // first loop for printing rows
 for (i = 1; i<=4; i++) {
      // second loop for printing numbers in each rows
for (j =1; j <= i; j++)
{
         printf("%d ", k);
                  k--;
         }
         printf("\n");
          }
            return 0;
}
```

//Program 6

//Write a program to print the pattern

```
1
12
123
1234
123
12
1
#include <stdio.h>
int main()
{
 int i, j;
  for(i=1;i<=4;i++)
```

```
 {
for (j=1;j<=i;j++)
{
printf ("%d",j);
}
printf ("\n");
}
 for(i=3;i>=1;i--)
{
 for (j=1;j<=i;j++)
{
printf ("%d",j);
  }
printf ("\n");
 }
}
```

//Program 7

// Write a Program for pattern

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

```
int main()
{
  int i,j,k;
     for (i = 1; i<=5; i++)
 {
  for(k=1;k<i;k++)
printf(" ");
  for (j =5; j >= i; j--)
{
printf("*");
}
          printf("\n");
 }
 for (i = 4; i>=1; i--)
{
          for(k=1;k<i;k++)
                    printf(" ");
           for (j =5; j >= i; j--)
           {
                    printf("*");
            }
          printf("\n");
   }
 return 0;
}
```

//Program 8

//Write a program to print the pattern 7

```
          *
        * * *
```

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

```
#include <stdio.h>
int main() {
  int i, space, j;
   for (i = 1; i<= 5; ++i)
{
    for (space = 5; space >i; space--)
 {
printf("  ");
    }
   for(j=1;j<=2*i-1;j++)
{
printf("* ");
   }
printf("\n");
  }
  return 0;
}
```

//Program 9

//Write a program to print the pattern

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
#include <stdio.h>
```

```
int main() {
  int i, j;

  for (i = 5; i>=1; i--)
  {
      for(j=1;j<=i;j++)
      {
printf("%d ",j);
      }
printf("\n");
  }
  return 0;
}
```

//Program 10

//Write a program to print the pattern

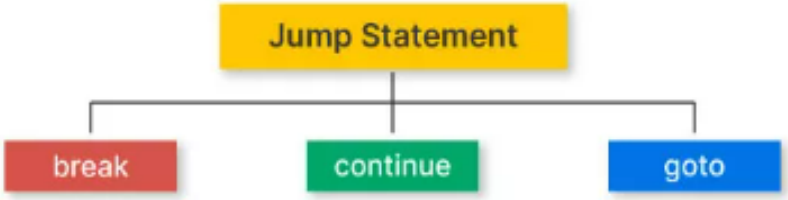
```
*
* *
* * *
* * * *
* * *
* *
*
#include <stdio.h>
int main() {
  int i, j;
  for (i =1; i<=4; i++)
  {
      for(j=1;j<=i;j++)
```

```
        {
printf("* ");
        }
printf("\n");
  }
  for (i =3; i>=1; i--)
  {
       for(j=1;j<=i;j++)
        {
printf("* ");
        }
printf("\n");
  }
  return 0;
}
```

3.1.3Jumping Statements

In c, there are control statements that do not need any condition to control the program execution flow. These control statements are called as unconditional control statements or jumping statements.

Cprogramminglanguageprovidesthefollowing jump statements.



The above three statements do not need any condition to control the program execution flow.

break statement

- When the break statement present inside a loop is executed, it terminates the loop and the program control automatically passes to the first statement after the loop.
- When the break statement presents inside a nested loop, it only terminates the execution of the nearest enclosing loop. The execution resumes with the statement present next to the terminated loop.
- There isnoconstraintaboutthe numberofbreak statements that can be present inside a loop.

```
while (testExpression) {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
```

```
do {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
```

continue statement

- A continue statement terminates the current iteration of the loop.
- When continue is encountered inside any loop, control automatically passes to the beginning of the loop.
- When continue statement present inside a nested loop, it only terminates the current iteration of the nearest enclosing loop.
- There is no constraint about the number of continue statements that can be present inside a loop.

Note: break and continue is usually associated with an if.

```
while (testExpression) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

```
do {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
} while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

goto statement

- goto is used to transfer the program control to a predefined label. label is a name given to the instruction or line in the program.
- However, using goto is avoided these days since it makes the program lessreadable and complicated. It also disturbs the normal flow of execution.
- Using goto statement we can jump from top to bottom or bottom to top.

Syntax of goto statement:

label:

//code

goto label;

Difference between break and continue

break	Continue
break is used to exit a loop entirely.	continue is used to skip the rest of the loop body for the current iteration and proceed to the next iteration.
The break statement can be used in switch statement. It can also be used inside the while loop, do-while loop, and for loop.	continue can still use it in for loops, do-while loops, and while loops. This means that continue can only happen in a loop, **not in a switch.**
Control is transfer outside the loop.	Control remains in the same loop.
Keyword 'break' is used in break statement.	Keyword 'continue' is used in continue statement.
Syntax: It can be written as: break;	**Syntax:** It can be written as: continue;
//program on break #include<stdio.h> void main() { int i; for (i = 0; i< 10; i++) { if (i == 5) break; // exit loop when i equals 5 printf("%d ", i); } } Output: 0 1 2 3 4	**//program on continue** #include<stdio.h> void main() { int i; for (i = 0; i<10; i++) { if (i == 5) continue; // skip iteration when i equals 2 printf("%d ", i); } }

	Output: 0 1 2 3 4 6 7 8 9

2017-18 (RCS-101)

1. Why we use do-while loop in c? Also tell any properties which you know? 2

2. Write a C program to add first seven terms of the following series using for loop. 1/1! +2/2! +3/3! +-------- 7

3. Write a program to check the number is palindrome of not. The program should accept any arbitrary number typed by user. 7

4. Write a program to print check a number is prime number or not. 7

2017-18 (RCS-201)

1. Write a program to find the Armstrong number from 1 to 100. 7

2. Write a program to generate a following numbers structure: 7

   12345

   1234

   123

   12

3. Write down the output of the following. 2

```
main()
{
int i=1;
for( ; ; )
{
printf("%d",i);
if(i= =7)
break;
}
}
```

2018-19(KCS-101)

1. Write the syntax format for while, do while and for loops. 3

2. How to use break statement in C? Explain it with some sort of code. 2

2018-19(KCS-201)

1. Write a program in C to print following pattern with appropriatecomments:

   10

   9 8

   7 6 5

   4 3 2 1 10

2. What is the use of break statement in loops? Write a program in C using

   while loop to elaborate the use of break statement. 10

3. Differentiate while and do while loop. 2

2019-20(KCS-101)

1. Take the three digit number from the user then write a program to check entered number is palindrome or not. 10
2. Write a program in C to generate the Fibonacci series upto the last Fibonacci number less than 100. Also finds the sum of all Fibonacci numbers and total count of all Fibonacci numbers. 10
3. Write a program in C to print the following pattern: 10

   2 3 4 5 6 7

   3 4 5 6 7

   4 5 6 7

   5 6 7

   6 7

   7

4. Differentiate while and do while loop. 2

2020-21(KCS-101T)

1. Write a program to find the entered number is palindrome or not. 10
2. Write a program in C to print the following pattern: 10

   ****

   ***

**

*

3. Differentiate while and do while loop. 2

2021-22(KCS-101T)

1. Differentiate between type conversion and type casting. Write a program to input a floating-point number and find leftmost digit of integral part of a number. 10

2. Show the usage of break statement. 2

2021-22(KCS-201T)

1. Write a program to print the pattern 10

1

12

123

1234

123

12

1

2. Write a Program for pattern 10

*****

****

***

**

*

**

***

****

*****

2022-23(BCS-101)

1. Write a program to print the pattern 7

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

2. Find the output of the following code: 2

```
#include<stdio.h>
main()
{
int i=1;
for(;;)
{ printf("%d",i);
if(i=5)
break;
}
}
```

2022-23(BCS-201)

1 Write a program in C to reverse a given number N having any number of digits. 7

2 Differentiate between while and do-while loop. Write a program in C to print the following pattern: 7

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

2023-24(BCS-101)

1 Write a program to print the pattern 7

*

* *

* * *

* * * *

* * *

* *

*

2 Write a program to check whether the entered number is prime or not. 7

3 Find the output of the code 2

```
#include<stdio.h>
main()
 {
 int a,b;
for(a=6,b=4;a<=24;a=a+6)
{
  if(a%b==0)
        break;
 }
printf("%d",a);
}
```

4. Write a Program for pattern **10**

*****

 ****

  ***

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

Program:

```
#include <stdio.h>
int main() {
    int i, j;
    int n = 5; // Number of rows for the top part of the pattern
    // Printing the top part of the pattern
    for (i = n; i > 0; i--) {
        for (j = 0; j < n - i; j++) {
            printf(" ");
        }
        for (j = 0; j < i; j++) {
            printf("*");
        }
        printf("\n");
    }
    // Printing the bottom part of the pattern
    for (i = 2; i <= n; i++) {
        for (j = 0; j < n - i; j++) {
            printf(" ");
        }
        for (j = 0; j < i; j++) {
            printf("*");
        }
```

```
        printf("\n");
    }
    return 0;
}
```