CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

cspit
Chandubhai S Patel Institute of Technology

CE143: COMPUTER CONCEPTS & PROGRAMMING

UNIT-6
Looping

N. A. Shaikh
nishatshaikh.it@charusat.ac.in

Topics to be covered

- **Need of looping**
- **entry-controlled loop(pre-test)**
  - **while**
  - **for**
- **exit-controlled loop(post-test)**
  - **do...while**
- **difference between Counter and Sentinel controlled loops**
- **Nesting of looping statements**
- **use of break & continue**
- **use of if...else in loop**
- **infinite loop.**

- In general, statements are executed **sequentially**.
- There are circumstances where you want to do the **same thing many times.**
- **Loops** in programming come into use when we need to repeatedly execute a block of statements.
- **For example:** Suppose we want to print “Hello World” 10 times.

Iterative Method

```c
#include <stdio.h>
void main()
{
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
    printf( "Hello World\n");
}
```

Using goto

```c
#include <stdio.h>
void main()
{
    int count=1;
    start:
        if(count<=10)
        {
            printf( "Hello World\n");
            count++;
            goto start;
        }
}
```

Using loop

In Loop, the statement needs to be written only once and the loop will be executed 10 times

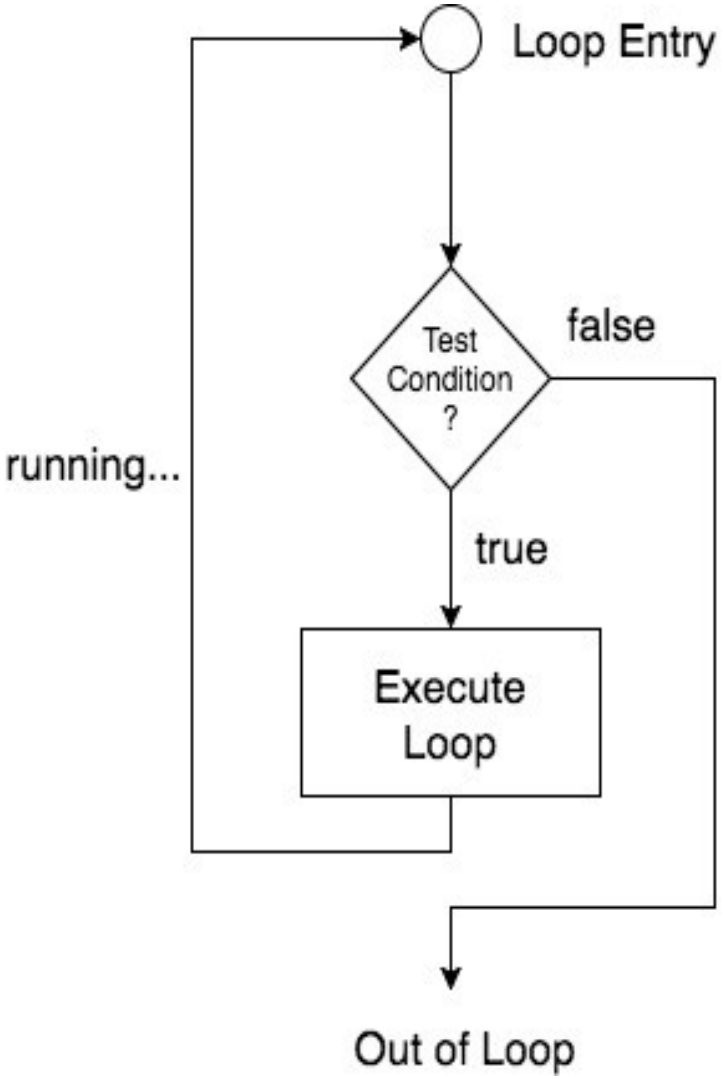
```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

- On such occasions where the exact number of repetitions are known , there are more convenient methods**(looping**)

- It enable us to develop concise programs containing repetitive processes **without the use of goto statements**.

In computer programming, a **loop is a sequence of instructions that is repeated until a certain condition is reached.**

Looping statements are also called as **iterative statements**



Loops in C language are mainly divided into two categories:

1. **Entry Controlled Loop(pre-test loop)**
2. **Exit Controlled Loop(post-test loop)**

Entry Controlled Loop	Exit Controlled Loop
Also known as **pre-test loop**	Also known as **post-test loop**
Test **condition is checked first**, and then loop body will be executed.	Loop **body will be executed first**, and then condition is checked.
If Test condition is false, loop body will not be executed.	If Test condition is false, loop body will be executed once.
Examples : **for loop and while loop**	Example : **do while loop**
Use : when checking of test condition is mandatory before executing loop body.	**Use** : when checking of test condition is mandatory after executing the loop body.

Types of Loop

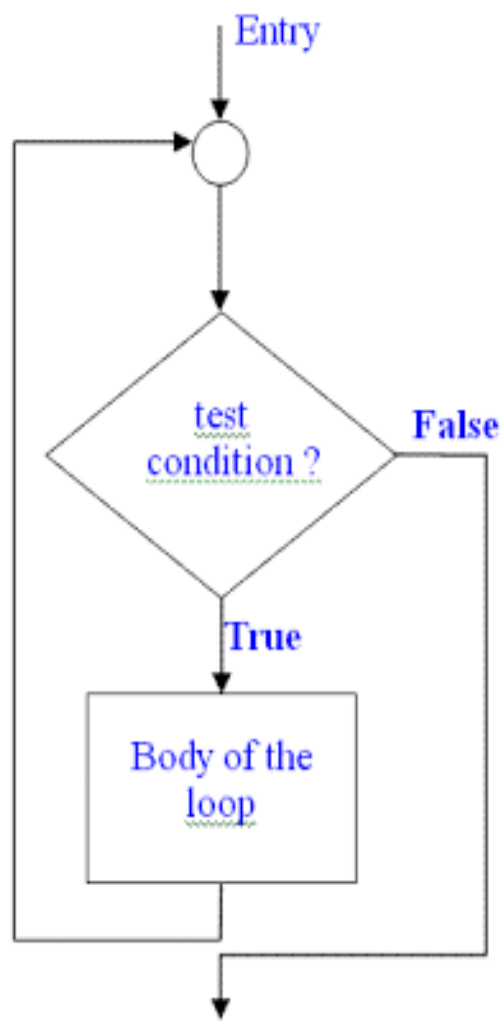


Figure : Entry controlled loop

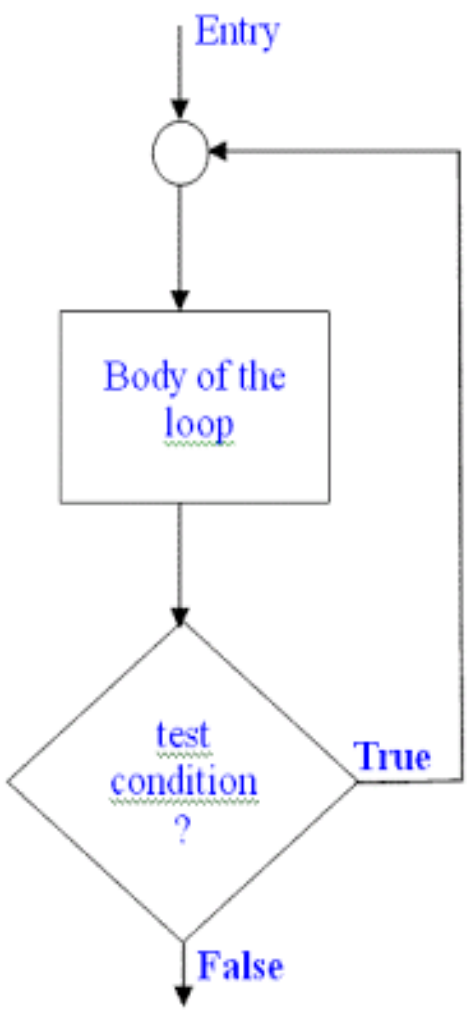
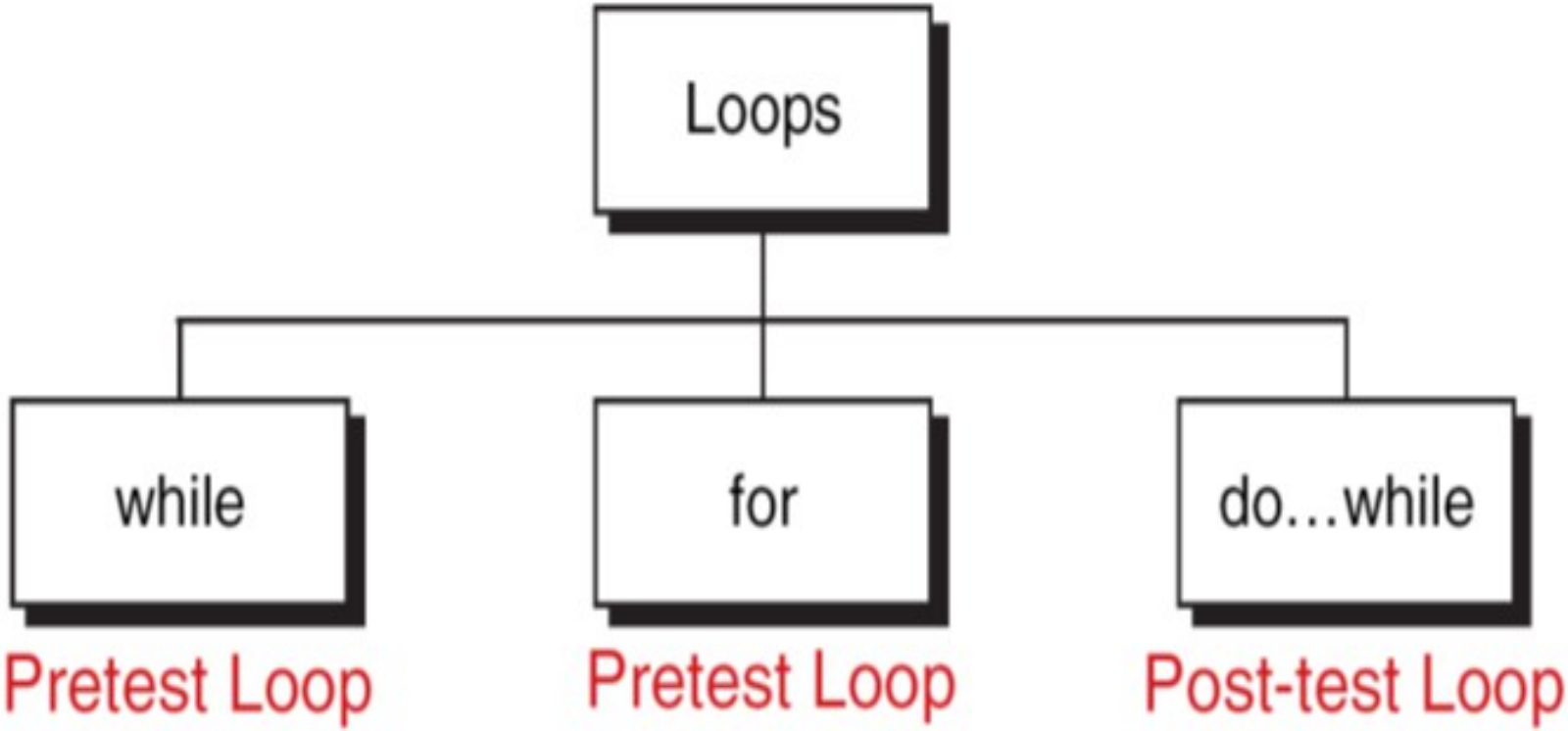


Figure : Exit controlled loop

Types of Loop



A looping process would include the following steps:

1. Setting and **initialization** of a condition variable.
2. **Execution** of the statements in the loop.
3. **Test** for a specified value of the condition variable for execution of the loop
4. Incrementing or **updating** the condition variable

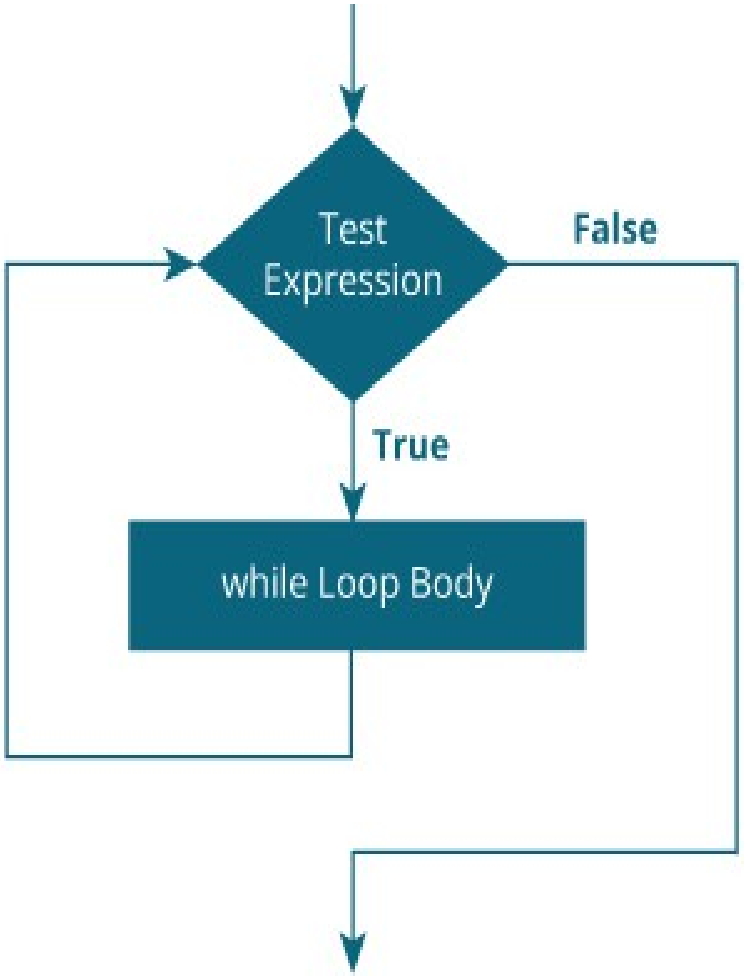
- **Simplest** looping structure
- **Entry-controlled loop statement**
- Used in situations where we **do not know the exact number of iterations** of loop beforehand.
- The loop execution is terminated on the basis of test condition.

Syntax:

```
initialization_expression;

while(test_expression)
    {
        // Body of the loop
        // Statements we want to execute
        update_expression;
    }
```

Flowchart:



How while loop works?

- 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).

```c
// C program to illustrate while loop
#include <stdio.h>
void main()
{
    // initialization expression
    int i = 1;

    // test expression
    while (i <= 10)
    {
        printf( "Hello World\n");

        // update expression
        i++;
    }
}
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

```c
//print series of numbers from 1 to 10
#include<stdio.h>
int main()
{
	int num=1;
	while(num<=10)
	{
		printf("%d\n",num);
		num++;
	}
return 0;
}
```

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

```c
//print series of numbers from 10 to 1
#include<stdio.h>
int main()
{
	int num=10;
	while(num>=1)
	{
		printf("%d\n",num);
		num--;
	}
return 0;
}
```

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

```c
//print the multiplication table of the number entered from the keyboard
#include<stdio.h>
void main()
{
    int i=1,number;
    printf("Enter a number: ");
    scanf("%d",&number);

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

```
Enter a number: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

```c
//even odd
#include<stdio.h>
void main()
{
    int a=1;
    while(a<=10)
    {
        if(a%2==0)
            printf("%d is even\n",a);
        else
            printf("%d is odd\n",a);
        a++;
    }
}
```

```
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
```

```c
//Sum of Natural Numbers
#include <stdio.h>
int main()
{
    int n, i, sum = 0;
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    i = 1;

    while (i <= n)
    {
        sum += i;
        ++i;
    }

    printf("Sum = %d", sum);
    return 0;
}
```

```
ter a positive integer: 4
m = 10
```

Two numbers are entered through the keyboard. Write a program to find the value of one number raised to the power of another. (Use While loop)

```
#include<stdio.h>
void main()
{
    int a, b, c;

    printf("Enter the value of a: ");
    scanf("%d", &a);
    printf("Enter the value of b: ");
    scanf("%d", &b);
    //using <math.h>--->c = pow(a,b);
    c = 1;

    while(b!=0)
    {
        c = c * a;
        b = b - 1;
    }
    printf("a raise to b is:%d",c);
}
```

```
Enter the value of a: 2
Enter the value of b: 3
a raise to b is:8
```

- Just like **relational operators (<, >, >=, <=, ! =, ==)**, we can also use **logical operators** in while loop.

The following scenarios are valid :

```
while(num1<=10 && num2<=10)

while(num1<=10||num2<=10)

while(num1!=num2 &&num1 <=num2)

while(num1!=10 ||num2>=num1)
```

```c
#include <stdio.h>
int main()
{
    int i=1, j=1;

    while (i <= 4 || j <= 3)
    {
     printf("%d %d\n",i, j);
     i++;
     j++;
    }
return 0;
}
```

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

Example 1

```c
#include<stdio.h>
void main ()
{
    int j = 1;
    while(j+=2,j<=10)
    {
        printf("%d ",j);
    }
}
```

```
3 5 7 9
```

Example 2

```c
#include<stdio.h>
void main ()
{
    int x = 10, y = 2;
    while(x+y-1)
    {
        printf("%d %d",x--,y--);
    }
}
```

Example 3

```
1  #include<stdio.h>
2  void main ()
3  {
4      while()
5      {
6          printf("Hello World");
7      }
8  }
```

```
4    error: expected expression before ')' token
```

1. **WAP to print all even numbers between 1 to 100**
2. **WAP to print sum of even numbers between the range entered by user**
3. **WAP to enter a number through keyboard and find the sum of its digits. (Input:234; output:9)**
4. **WAP to enter few numbers and count the positive and negative numbers together with their sums. When 0 is entered program should be terminated.**
5. **WAP to print the entered number in the reversed order.(Input:5428; Output:8245)**
6. **WAP to calculate factorial of a given number.**

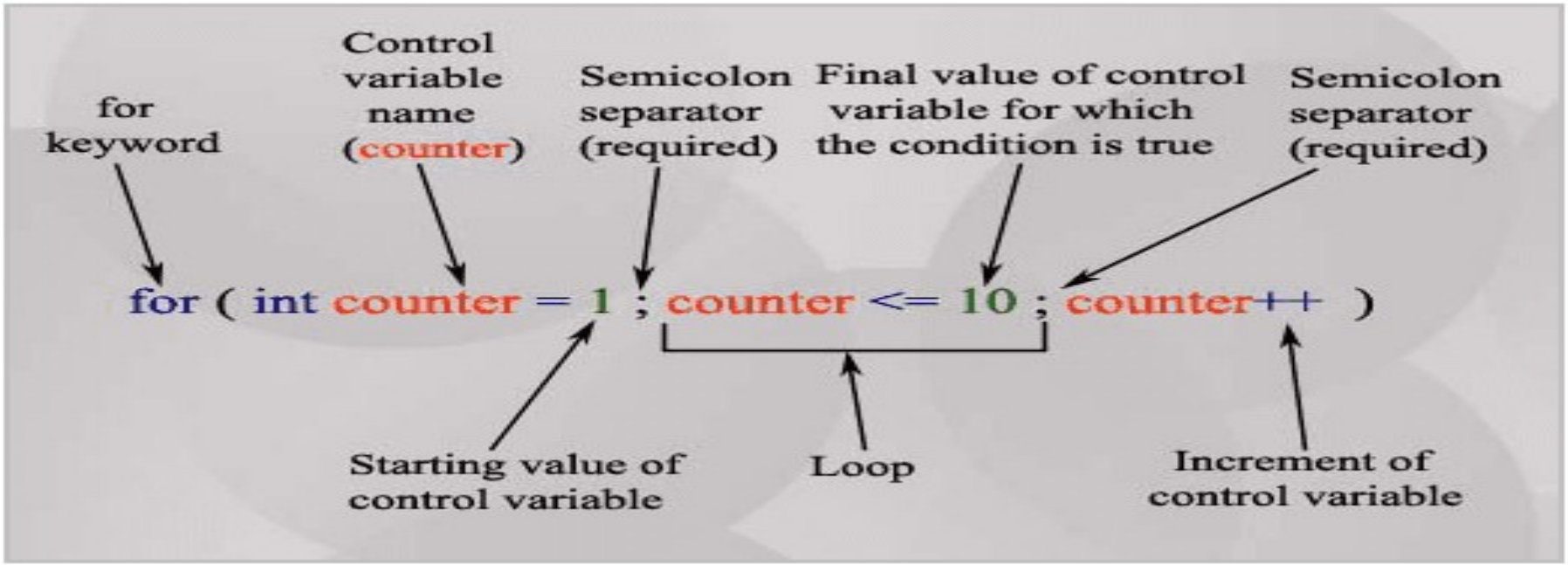
- **Entry-controlled loop statement**

- Used in situations where we know the exact number of iterations of loop beforehand

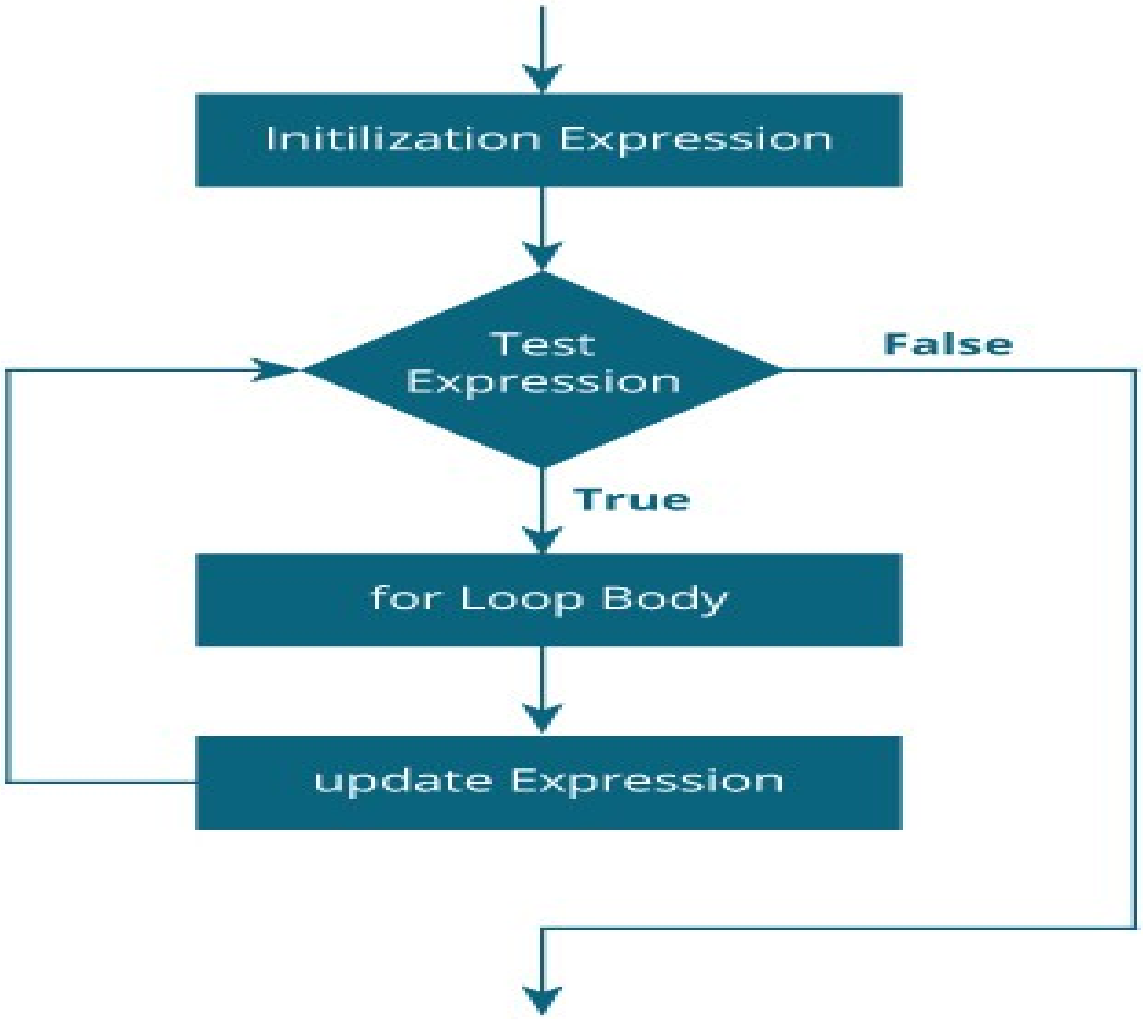
  - **i.e. the number of times the loop body is needed to be executed is known to us.**

Syntax:

```
for (initialization_expr; test_expr; update_expr)
    {
        // body of the loop
        // statements we want to execute
    }
```



Flowchart:


Initilization Expression
Test Expression
False
True
for Loop Body
update Expression

How for loop works?

- 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**.

```c
// C program to illustrate for loop
#include <stdio.h>

int main()
{
    int i;

    for (i = 1; i <= 10; i++)
    {
        printf( "Hello World\n");
    }

    return 0;
}
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

For Loop

```c
// Print numbers from 1 to 10
#include <stdio.h>

int main()
{
   int i;

   for (i = 1; i < 11; ++i)
   {
      printf("%d\n", i);
   }
   return 0;
}
```

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

```c
// Print numbers from 10 to 1
#include <stdio.h>

int main()
{
   int i;

   for (i = 10; i > 0; --i)
   {
      printf("%d\n", i);
   }
   return 0;
}
```

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

Write a program to print the multiplication table of the number entered from the keyboard Using for loop

```c
#include <stdio.h>
int main()
{
    int n, i;
    printf("Enter an integer: ");
    scanf("%d", &n);

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

```
Enter an integer: 12
12 * 1 = 12
12 * 2 = 24
12 * 3 = 36
12 * 4 = 48
12 * 5 = 60
12 * 6 = 72
12 * 7 = 84
12 * 8 = 96
12 * 9 = 108
12 * 10 = 120
```

```c
// Program to calculate the sum of first n natural numbers
// Positive integers 1,2,3...n are known as natural numbers
#include <stdio.h>
void main()
{
    int num, i, sum = 0;
    printf("Enter a positive integer: ");
    scanf("%d", &num);

    for(i = 1; i <= num; ++i)
    {
        sum += i;
    }
    printf("Sum = %d", sum);
}
```

```
Enter a positive integer: 4
Sum = 10
```

```c
//WAP to print Fibonacci Series up to n terms
#include<stdio.h>
void main()
{
    int n1=0,n2=1,n3,i,n;

    printf("Enter the number of elements:");
    scanf("%d",&n);

    printf("\n%d %d",n1,n2);//printing 0 and 1

    for(i=2;i<n;++i)//loop starts from 2 because 0 and 1 are already printed
        {
            n3=n1+n2;
            printf(" %d",n3);
            n1=n2;
            n2=n3;
        }
}
```

```
Enter the number of elements:10

0 1 1 2 3 5 8 13 21 34
```

```c
//WAP to print nth Fibonacci number
#include<stdio.h>
void main()
{
    int n1=0,n2=1,n,i,fib;

    printf("\n\nEnter the value of n:");
    scanf("%d",&n);

    for(i=1;i<=n-2;i++)
    {
        fib=n1+n2;
        n1=n2;
        n2=fib;
    }
    printf("\nnth fibonacci number=%d ",fib);
}
```

```
Enter the value of n:5

nth fibonacci number=3
```

Example (Initialization)

```
//initialize more than one variable in Expression 1
#include <stdio.h>
int main()
{
    int a,b,c;
    for(a=0,b=12,c=23;a<2;a++)
    {
        printf("%d ",a+b+c);
    }
}
```

```
35 36
```

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

```
1 2 3 4
```

Example (conditional expression)

```
//Expression-2 can have more than one condition
//loop will iterate until the last condition becomes false
//Other conditions will be treated as statements.
#include <stdio.h>
int main()
{
    int i,j,k;
    for(i=0,j=0,k=0;i<4,k<8,j<10;i++)
    {
        printf("%d %d %d\n",i,j,k);
        j+=2;
        k+=3;
    }
}
```

```
0 0 0
1 2 3
2 4 6
3 6 9
4 8 12
```

Example (conditional expression)

```
//infinite loop
#include <stdio.h>
int main()
{
    int i;
    for(i=0;;i++)
    {
        printf("%d",i);
    }
}
```

[redacted]

Example (Update expression)

```
//We can update more than one variable at the same time.
#include<stdio.h>
void main ()
{
    int i,j=2;
    for(i = 0;i<5;i++,j=j+2)
    {
        printf("%d %d\n",i,j);
    }
}
```

```
0 2
1 4
2 6
3 8
4 10
```

Various forms of for loop

```
for (num=10; num<20; )
    {
        //Statements
        num++;
    }
```

```
int num=10;
for (;num<20;)
    {
        //Statements
        num++;
    }
```

```
for( x=(m+n)/2;x>0; x=x/2 )
```

```
sum=0;
for(i=1; i<20 && sum<100; i++)
{
    //statements
}
```

```
for ( int x = 1; x <= 100 ; x++ )
    {
        printf("%d\n",x);
    }
```

Example

```
//variable declared inside for loop is valid only for that block and not outside
#include<stdio.h>
void main ()
{
    int i;
    for(i=0;i<10;i++)
    {
        int i = 20;
        printf("%d ",i);
    }
}
```

```
20 20 20 20 20 20 20 20 20 20
```

Example

```
//infinite for loop.
#include<stdio.h>
void main ()
{
    for(;;)
    {
        printf("welcome to CHARUSAT");
    }
}
```

1. **WAP to check whether entered number is prime or not**
2. **WAP to print all the prime numbers between 1 and n.**
3. **WAP to print even numbers between range entered by user**
4. **WAP to find sum of all even numbers between range entered by user.**

- The **while loop** test the condition before the loop is executed. Therefore, body of loop may not be executed at all if the condition is not satisfied at the very first attempt.
- On some occasions it might be necessary to execute the body of the loop before the test is performed.
- Such situations can be handled with the help of the **do while** statement.
- It is **exit controlled** loop(check the condition at the bottom of the loop) and therefore the body of the loop is always **executed at least once**

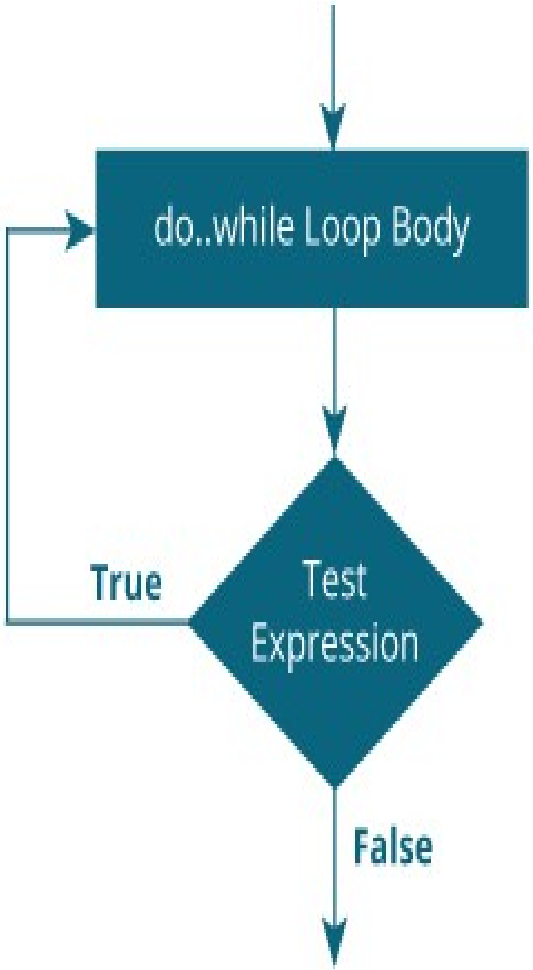
- The do-while loop is mainly used in the case where we need to execute the loop at least once
- Mostly used in **menu-driven programs** where the termination condition depends upon the end user.

Syntax:

```
initialization_expression;

do
{
    //Body of the loop
    //statement we want to execute
    update_expression;
}while(test_expression);
```

Flowchart:



How do…while loop works?

- The body of do...while loop is **executed once**. Only then, 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.

NOTE: The do-while loop will run **infinite times** if we pass any non-zero value as sion.

```
do
{
    //statement
}while(1);
```

do while Loop

```c
//C Program to illustrate do-while loop
#include<stdio.h>
void main()
{
    //Initialization_expression
    int i=1;

    do
    {
        //loop body
        printf("Hello World\n");
        i++; //update_expression
    }while(i<=10); //test_expression
}
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

do while Loop

```c
//Print series of numbers from 1 to 10
#include <stdio.h>
int main ()
{
    int num = 1;
    do
    {
        printf("%d\n", num);
        num++;
    }while( num <= 10 );
return 0;
}
```

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

```c
//Print series of numbers from 10 to 1
#include <stdio.h>
int main ()
{
    int num = 10;
    do
    {
        printf("%d\n", num);
        num--;
    }while( num >= 1 );
return 0;
}
```

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

do while Loop

```c
//Print the multiplication table of the number entered from the keyboard
#include<stdio.h>
int main()
{
    int i=1,number=0;
    printf("Enter a number: ");
    scanf("%d",&number);

    do
    {
        printf("%d * %d = %d\n",number,i,number*i);
        i++;
    }while(i<=10);
return 0;
}
```

```
Enter a number: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

do while Loop

```c
// Program to add numbers until the user enters zero
#include <stdio.h>
int main()
{
    double number, sum = 0;

    // the body of the loop is executed at least once
    do
    {
        printf("Enter a number: ");
        scanf("%lf", &number);
        sum += number;
    }
    while(number != 0.0);

    printf("Sum = %.2lf",sum);
return 0;
}
```

```
Enter a number: 1.5
Enter a number: 2.4
Enter a number: -3.4
Enter a number: 4.2
Enter a number: 0
Sum = 4.70
```

do while Loop

```c
//Menu-Driven Program using do while
#include<stdio.h>
#include<stdlib.h>
void main ()
{
    int choice;

    do
    {
    printf("\n1. Print Hello\n2. Print CHARUSAT\n3. Exit\n");
    scanf("%d",&choice);

       switch(choice)
       {
           case 1 : printf("Hello");
                    break;
           case 2:  printf("CHARUSAT");
                    break;
           case 3:  exit(0);
                    break;
           default: printf("please enter valid choice");
       }

    }while(choice!=3);
}
```

```
1. Print Hello
2. Print CHARUSAT
3. Exit
1
Hello
1. Print Hello
2. Print CHARUSAT
3. Exit
2
CHARUSAT
1. Print Hello
2. Print CHARUSAT
3. Exit
4
please enter valid choice
1. Print Hello
2. Print CHARUSAT
3. Exit
3
```

Difference between While and do while

While

```c
int i = 0;
while(i > 0)
{
    printf("%d", i);
    i--;
}
```

Output: No Output

do-While

```c
int i = 0;
do
{
    printf("%d", i);
    i--;
} while(i > 0);
```

Output: 0

Difference between While and do while

while	do while
Condition is checked first then statement(s) is executed.	Statement(s) is executed at least once, thereafter **condition** is checked.
It might occur statement(s) is executed **zero** times, If condition is false.	At least **once** the statement(s) is executed.
No **semicolon** at the end of while. while(condition)	**Semicolon** at the end of while. while(condition);
If there is a single statement, **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.
while loop is **entry** controlled loop.	do-while loop is **exit** controlled loop.
while(condition) { statement(s); }	do { statement(s); } while(condition);

When should I prefer do while over while?

WAP which allows user to enter an integer until he/she enters a value zero

Using while

```c
#include<stdio.h>
void main()
{
    int n;
    printf("Enter an integer\n");
    scanf("%d",&n);

    while(n!=0)
    {
        printf("Enter an integer\n");
        scanf("%d",&n);
    }

    printf("You are out of the loop");
}
```

Using do while

```c
#include<stdio.h>
void main()
{
    int n;

    do
    {
        printf("Enter an integer\n");
        scanf("%d",&n);
    }while(n!=0);

    printf("You are out of the loop");
}
```

Comparison of the Three loops

for

```
for(n=1;n<=10;n++)
{
    ________________
    ________________
}
```

while

```
n=1;
while(n<=10)
{
    ________________
    ________________

    n++;
}
```

do while

```
n=1;
do
{
    ________________
    ________________

    n++;
}
while(n<=10);
```

- Use **for** loop when **number of iterations is known** beforehand. i.e. the number of times the loop body is needed to be executed is known.

- Use **while** loops where exact **number of iterations is not known** but the **loop termination condition is known.**

- Use **do while** loop if the code **needs to be executed at least once** like in **Menu driven programs**

Example: Prime Number

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

    for (i = 2; i <= n / 2; ++i)
    {
        // condition for non-prime
        if (n % i == 0)
        {
            flag = 1;
            break;
        }
    }

    if (n == 1)
    {
        printf("1 is neither prime nor composite.");
    }
    else
    {
        if (flag == 0)
            printf("%d is a prime number.", n);
        else
            printf("%d is not a prime number.", n);
    }
}
```

```
Enter a positive integer: 17
17 is a prime number.
```

Example: Perfect Number

```c
#include<stdio.h>
void main()
{
    int n,sum = 0, i;
    printf("Enter a number\n");
    scanf("%d", &n);

    // find all divisors and add them
    for(i = 1; i < n; i++)
        {
            if (n % i == 0)
            {
                sum = sum + i;
            }
        }

        if (sum == n)
            printf(" %d is a Perfect Number",n);
        else
            printf("\n %d is not a Perfect Number",n);
}
```

```
Enter a number
6
 6 is a Perfect Number
```

Example: Factorial Number

```c
#include<stdio.h>
int main()
{
 int i,fact=1,n;
 printf("Enter a number: ");
 scanf("%d",&n);

    for(i=1;i<=n;i++)
        {
            fact=fact*i;
        }
  printf("Factorial of %d is: %d",n,fact);
return 0;
}
```

```
Enter a number: 5
Factorial of 5 is: 120
```

Write a menu driven program which has following options:

1. Prime or not
2. Perfect number or not
3. Factorial of a number
4. Exit

Use do...while statement so that the menu is displayed at least once. Also use Switch statement.

```c
#include <stdio.h>
void main()
{
int choice,n,i,flag=0,sum=0,fact=1;

  do
  {
    printf("\n\n********** Main Menu **********\n");
    printf("\n(1).Prime or not");
    printf("\n(2).Perfect number or not");
    printf("\n(3).Factorial of a number");
    printf("\n(4).Exit\n");
    scanf("%d",&choice);
```

```
  switch (choice)
  {
  case 1: printf("********** Prime number or composite number **********\n");
          //logic of Prime
          break;
  case 2:printf("********** Perfect number finder **********\n");
         //logic of Perfect number
         break;
  case 3:printf("********** Factorial Calculator **********\n");
         //logic of Factorial
         break;
  case 4:printf("Program terminated.\nPress any key to exit");
         break;
  default: printf("wrong input");
  }

 }while(choice != 4);
  printf("bye");
}
```

OUTPUT

```
********** Main Menu **********

(1).Prime or not
(2).Perfect number or not
(3).Factorial of a number
(4).Exit
1
********** Prime number or composite number **********
Enter a positive integer: 17
17 is a prime number.

********** Main Menu **********

(1).Prime or not
(2).Perfect number or not
(3).Factorial of a number
(4).Exit
2
********** Perfect number finder **********
Enter a number
6
 6 is a Perfect Number

********** Main Menu **********

(1).Prime or not
(2).Perfect number or not
(3).Factorial of a number
(4).Exit
3
********** Factorial Calculator **********
Enter a number: 5
Factorial of 5 is: 120

********** Main Menu **********

(1).Prime or not
(2).Perfect number or not
(3).Factorial of a number
(4).Exit
5
```

```
********** Main Menu **********

(1).Prime or not
(2).Perfect number or not
(3).Factorial of a number
(4).Exit
4
Program terminated.
```

Example: Armstrong Number

```c
#include <stdio.h>
#include <math.h>
void main()
{
  int Number, Temp, Reminder, Times =0, Sum = 0;
  printf("\nPlease Enter number to Check for Armstrong \n");
  scanf("%d", &Number);

  //Helps to prevent altering the original value
  Temp = Number;
  while (Temp != 0)
   {
      Times = Times + 1;
      Temp = Temp / 10;
   }
```

Example: Armstrong Number

```
Temp = Number;
while( Temp > 0)
 {
   Reminder = Temp %10;
   Sum = Sum + pow(Reminder, Times);
   Temp = Temp /10;
 }

printf("\n Sum of entered number is = %d\n", Sum);

if ( Number == Sum )
    printf("\n %d is Armstrong Number.\n", Number);
else
    printf("\n %d is not a Armstrong Number.\n", Number);
}
```

```
Please Enter number to Check for Armstrong
153

 Sum of entered number is = 153

 153 is Armstrong Number.
```

Example: Reverse Number

```c
#include<stdio.h>
int main()
{
    int n, reverse=0, rem;
    printf("Enter a number: ");
    scanf("%d", &n);

    while(n!=0)
        {
            rem=n%10;
            reverse=reverse*10+rem;
            n/=10;
        }
  printf("Reversed Number: %d",reverse);
return 0;
}
```

```
Enter a number: 123
Reversed Number: 321
```

Example: Palindrome Number

```c
#include <stdio.h>
void main()
{
    int n, reverse = 0, rem, original;
    printf("Enter an integer: ");
    scanf("%d", &n);
    original = n;

    // reversed integer is stored in reverse
    while (n != 0)
        {
            rem = n % 10;
            reverse = reverse * 10 + rem;
            n /= 10;
        }

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

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

1. **WAP to find sum of first and last digit of any number.**
2. **WAP to swap first and last digit of a number**
3. **WAP to print number in words**
4. **WAP to find all factors of a number**
5. **WAP to find prime factors of a number**
6. **WAP to check whether a number is Strong number or not**

- Based on the nature of the **control variables** and the kind of value assigned to, the loops may be classified into two general categories:

1. **Counter Controlled Loop**
2. **Sentinel Controlled Loop**

- Also known as **definite repetition loop**.
- **Number of iterations is known** before the loop begins to execute.

It has the following components:

1. A **control variable**.
2. The increment (or decrement)value
3. The loop terminating condition

Example: printing the first 10 natural numbers.

```
int count;
for( count=1; count<=100; count++)
    printf("%d",count);
```

- Also known as **indefinite repetition loop**
- **Number of iterations is not known** before the loop starts executing.
- A special value called **sentinel value** is used to change the loop control expression from true to false

- **Example:** Reverse a given number

```
int reverseNumber=0;
int number=12345;

while(number>0)
{
    reverseNumber= (reverseNumber*10)+ number%10;
    number/=10;
}
printf("Reverse Number is: %d\n", reverseNumber);
```

Counter Vs Sentinel Controlled Loop

BASIS OF COMPARISON	COUNTER CONTROLLED LOOP	SENTINEL CONTROLLED LOOP
Definition	A counter controlled loop is the definite repetition loop as the number of repetitions is known before the loop begins executing	A sentinel controlled loop is the indefinite repetition loop as the number of repetitions is not known before the loop begins executing
Controlling variable	Controlled variable used is know as counter.	Controlled variable variable used is know as sentinel variable.
Number of iteration	Known number of iteration.	Unknown number of iteration
Value of variable	The value of the variable is strict.	The value of the variable is not strict and it varies.
Limitation of variable	The Limitation of the variable is strict also.	The Limitation of the variable is strict.

Example	`int sum = 0;`
	`int n = 1;`
	`while (n <= 10)`
	`{`
	`sum = sum + n*n;`
	`n = n+ 1;`
	`}`
		`do`
		`{`
		`printf("Input a number.n");`
		`scanf("%d", &num);`
		`}while(num>0);`

- **Nested loop** means a **loop statement inside another loop statement.**
- nested loops are also called as "**loop inside loop**"

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

Outer_loop and **Inner_loop** are the valid loops that can be a **'for' loop**, **'while' loop** or **'do-while' loop**.

Note: There is no rule that a loop must be nested inside its own type. In fact, there can be any type of loop nested inside any type and to any level.

Nested for loop

```
for (initialization; condition; update)
{
    for(initialization; condition; update)
    {
        // inner loop statements.
    }
    // outer loop statements.
}
```

Nested while loop

```
while(condition)
{
    while(condition)
    {
        // inner loop statements.
    }
// outer loop statements.
}
```

Nested do…while loop

```
do
{
    do
    {
        // inner loop statements.
    }while(condition);
// outer loop statements.
}while(condition);
```

```
Enter the number of rows : 5
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

```c
#include<stdio.h>
void main()
{
    int i,j,n;
    printf("\n Enter the number of rows : ");
    scanf("%d",&n);

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

```
Enter the number of rows : 5
*
* *
* * *
* * * *
* * * * *
```

```
Enter the number of rows5
 * * * * *
 * * * *
 * * *
 * *
 *
```

```c
#include<stdio.h>
void main()
{
    int i,j,n;
    printf("\n Enter the number of rows : ");
    scanf("%d",&n);

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

```c
#include <stdio.h>
void main()
{
    int n,i,j;
    printf("\n Enter the number of rows");
    scanf("%d",&n);

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

```
Enter the number of rows5
1
10
101
1010
10101
```

```c
void main()
{
    int i,j,n,r;
    printf("Enter the number of rows");
    scanf("%d",&n);

    for(i=1;i<=n;i++)
    {
        for(j=1;j<=i;j++)
        {
          r=j%2;
          printf("%d", r);
        }
        printf("\n");
    }
}
```

```
Enter the no of lines
5

12345
2345
345
45
5
```

```c
void main()
{
    int i,j,n;

    printf("Enter the no of lines\n");
    scanf("%d",&n);
    printf("\n");

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

```
Enter the number of rows5
*****
 ****
  ***
   **
    *
```

```
void main()
{
    int n,m,j;
    printf("Enter the number of rows");
    scanf("%d",&n);
    m=n;
   for(int i=1;i<=n;i++)
   {
       for(j=1;j<i;j++)
       {
           printf(" ");
       }
       for(j=1;j<=m;j++)
       {
           printf("*");
       }
       m--;
       printf("\n");
    }
}
```

```
Enter the number of rows5
    *
   **
  ***
 ****
*****
```

```
void main()
{
    int n,m=1,j;
    printf("Enter the number of rows");
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        for(j=1;j<=n-i;j++)
        {
          printf(" ");
        }
        for(j=1;j<=m;j++)
        {
            printf("*");
        }
        m++;
        printf("\n");
    }
}
```

```
Enter the number of rows5
ABCDE
 1234
  ABC
   12
    A
```

```
Enter the number of rows5
    1
   12
  123
 1234
12345
```

```c
main()
{
    int i,j,n;
    printf("Enter the number of rows");
    scanf("%d",&n);

    for(i=n;i>=1;i--)
    {
        for(j=1;j<=n-i;j++)
        {
            printf(" ");
        }
        for(j=1;j<=i;j++)
        {
            if(i%2==0)
            {
                printf("%d",j);
            }
            else
            {
                printf("%c",j+64);
            }
        }printf("\n");
    }
}
```

```c
void main()
{
    int n,i,j;
    printf("Enter the number of rows");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n-i;j++)
        {
          printf(" ");
        }
        for(j=1;j<=i;j++)
        {
            printf("%d",j);
        }
        printf("\n");
    }

}
```

```
Enter the number of rows5
    *
   ***
  *****
 *******
*********
```

```
void main()
{
    int n,j;
    printf("Enter the number of rows");
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        for(j=1;j<=n-i;j++)
        {
          printf(" ");
        }
        for(j=1;j<=(2*i-1);j++)
        {
            printf("*");
        }
        printf("\n");
    }
}
```

```
Enter the number of rows5
*********
 *******
  *****
   ***
    *
```

```
void main()
{
    int n,j;
    printf("Enter the number of rows");
    scanf("%d",&n);
    for(int i=n;i>=1;i--)
    {
        for(j=1;j<=n-i;j++)
        {
          printf(" ");
        }
        for(j=1;j<=(2*i-1);j++)
        {
            printf("*");
        }
        printf("\n");
    }
}
```

```
Enter the no of lines:4
   1
  121
 12321
1234321
```

```c
void main()
{
        int i,n,j;
        printf("Enter the no of lines:");
        scanf("%d",&n);

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

Multiplication Table Using nested for loop

```c
#include <stdio.h>
int main()
{
    int num,i;

    for(num=1;num<=4;num++)
        {
            printf("Multiplication Table of %d\n",num);

            for (i = 1; i <= 10; i++)
            {
                printf("%d*%d=%d\n", num, i, num * i);
            }
        }
return 0;
}
```

OUTPUT

```
Multiplication Table of 1
1*1=1
1*2=2
1*3=3
1*4=4
1*5=5
1*6=6
1*7=7
1*8=8
1*9=9
1*10=10
Multiplication Table of 2
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
2*10=20
Multiplication Table of 3
3*1=3
3*2=6
3*3=9
3*4=12
3*5=15
3*6=18
3*7=21
3*8=24
3*9=27
3*10=30
Multiplication Table of 4
4*1=4
4*2=8
4*3=12
4*4=16
4*5=20
4*6=24
4*7=28
4*8=32
4*9=36
4*10=40
```

1. **A class of n students take an annual examination in m subjects. WAP program to read the marks obtained by each student in various subjects and to compute and print the total marks obtained by each of them.**

2.

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

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

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

```
A A A A
1 1 1
B B
2
```

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

- Loops perform a set of repetitive task until text expression becomes false.
- But it is sometimes desirable to **skip some statement/s** inside **loop(continue**) or **terminate the loop(break)** immediately.
- Jump statements are used to **interrupt the normal flow of program.**

Jump Statements in c:

- **break**
- **continue**
- **goto**

- Used to terminate case in the **switch** statement.
- Also used in **terminating the loop**(for, while, do while) immediately after it is encountered.
- In C programming, break statement is used with **conditional if statement**.

Syntax:

break;

NOTE: If you are using nested loops, the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

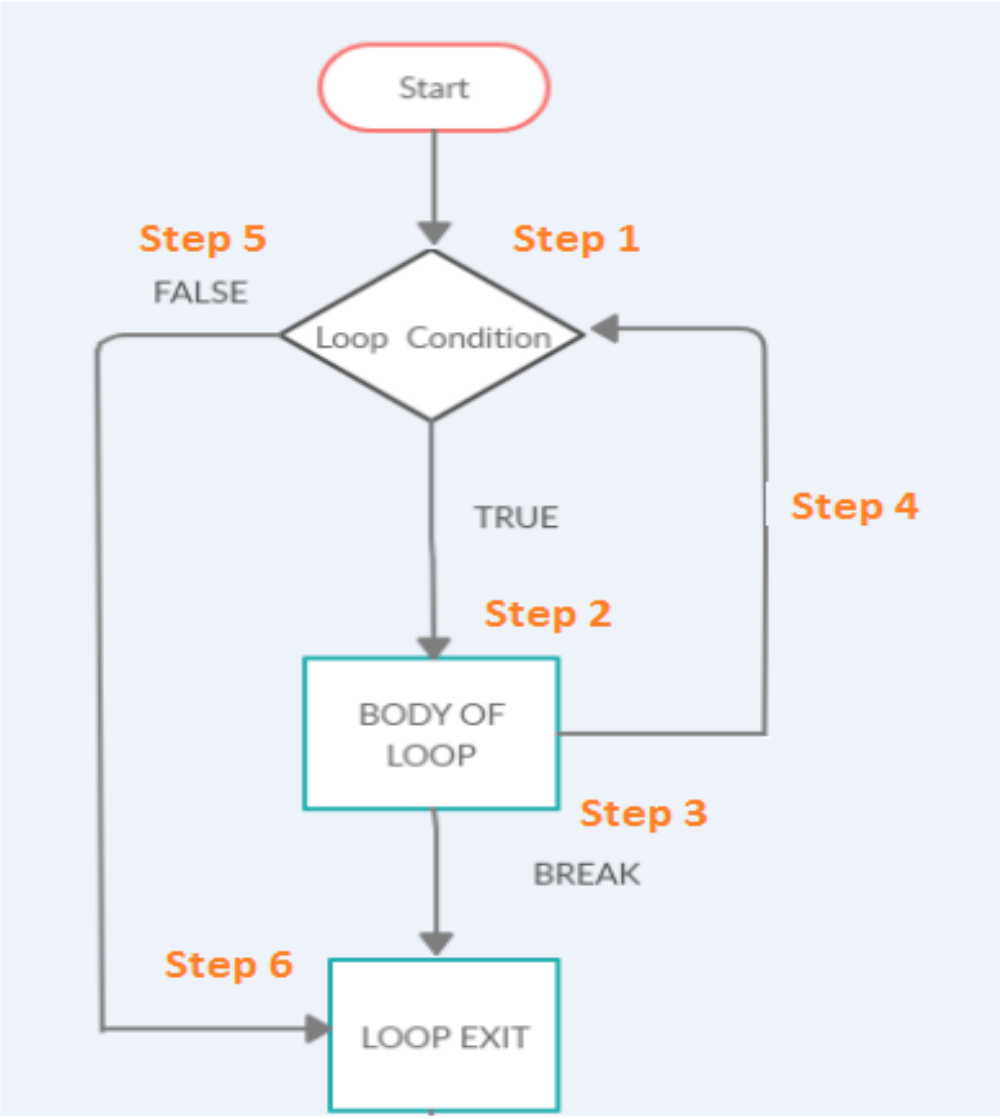
break Statements

```
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
}
```

break Statements



break Statements

```c
int main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        if(i==5)
        {
            printf("\nComing out of loop when i=5\n");
            break;
        }
        printf("\n%d",i);
    }
return 0;
}
```

```
1
2
3
4
Coming out of loop when i=5
```

break Statements

```c
int main ()
{
int a = 10;

while( a < 20 )
    {
        printf("value of a: %d\n", a);
        a++;

        if( a > 15)
            {
                /* terminate the loop using break statement */
                break;
            }
}
return 0;
}
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

break Statements

```c
// Program to calculate the sum of numbers (10 numbers max)
// If the user enters a negative number, the loop terminates
#include <stdio.h>
void main()
{
    int i;
    double number, sum = 0.0;

    for (i = 1; i <= 10; ++i)
        {
            printf("Enter a n%d: ", i);
            scanf("%lf", &number);

            // if the user enters a negative number, break the loop
            if (number < 0.0)
                {
                    break;
                }
            sum += number; // sum = sum + number;
        }
        printf("Sum = %.2lf", sum);
}
```

```
Enter a n1: 2.4
Enter a n2: 4.5
Enter a n3: 3.4
Enter a n4: -3
Sum = 10.30
```

break Statements

```c
//break statement with the nested loop
#include<stdio.h>
int main()
{
    int i=1,j=1;
    for(i=1;i<=3;i++)
        {
            for(j=1;j<=3;j++)
            {
                printf("%d %d\n",i,j);
                if(i==2 && j==2)
                    {
                        break;//will break loop of j only
                    }
            }//end of for loop
        }
return 0;
}
```

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

- It is sometimes desirable to **skip some statements** inside the loop. In such cases, continue statement is used.
- When a continue statement is encountered inside a loop, **control jumps to the beginning of the loop for next iteration**, skipping the execution of statements inside the body of loop for the current iteration.
- The continue statement in C programming works somewhat like the break statement. Instead of forcing termination, it **forces the next iteration** of the loop to take place, skipping any code in between.
- Just like break, continue is also used with **conditional if statement.**

Syntax:

continue;

- For the **for loop**, continue statement causes the conditional test and increment portions of the loop to execute.
- For the **while** and **do...while loops**, continue statement causes the program control to pass to the conditional tests.

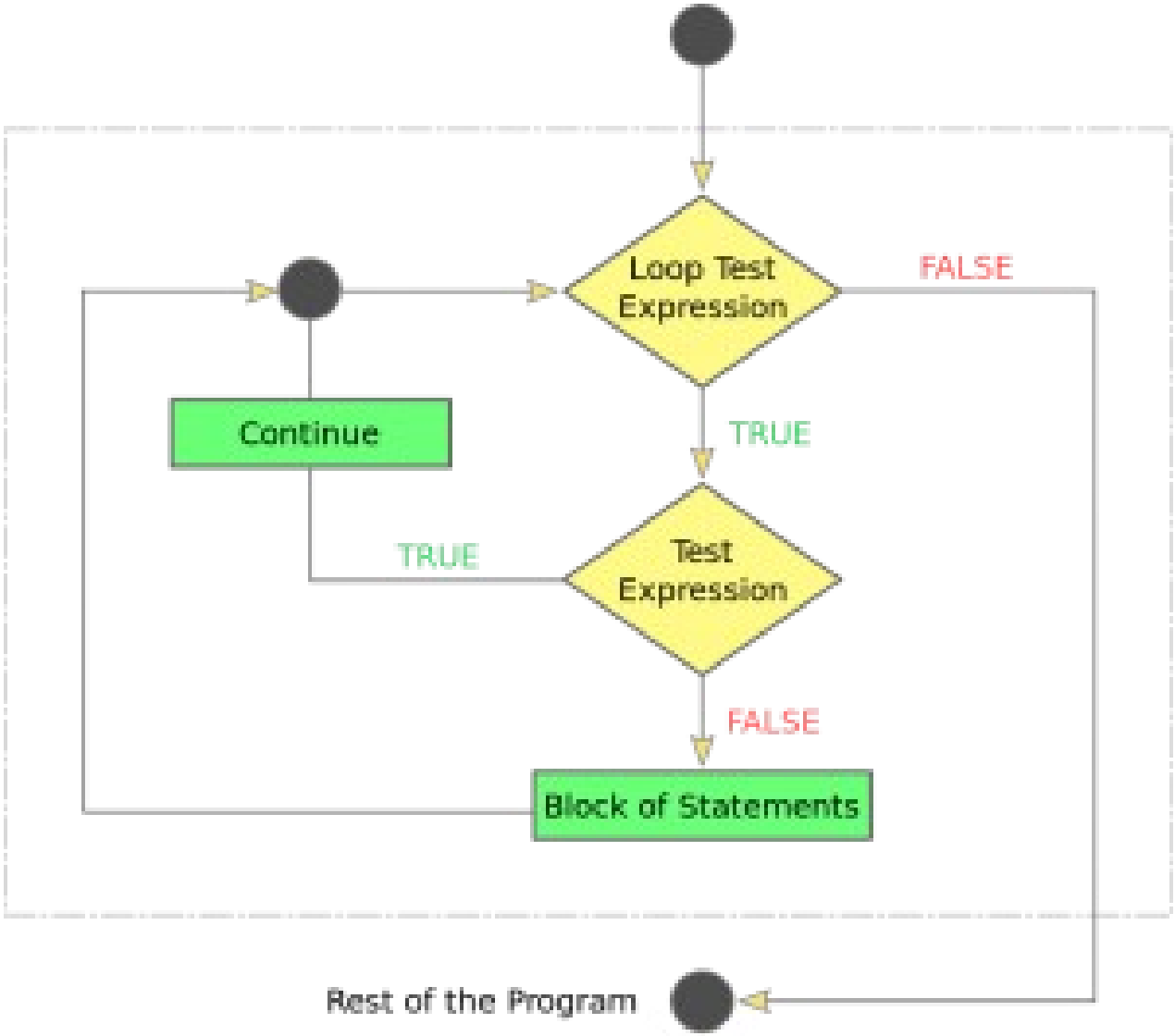
continue Statements

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

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

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

continue Statements



continue Statements

```c
int main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        if(i==5)
        {
            printf("\nSkipping %d from display using continue",i);
            continue;
        }
        printf("\n%d",i);
    }
return 0;
}
```

```
1
2
3
4
Skipping 5 from display using continue
6
7
8
9
10
```

continue Statements

```c
int main ()
{
int a = 10;
do
    {
        if( a == 15)
        {
            /* skip the iteration */
            a = a + 1;
            continue;
        }
        printf("value of a: %d\n", a);
        a++;
    } while( a < 20 );

return 0;
}
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

continue Statements

```c
// Program to calculate the sum of numbers (10 numbers max)
// If the user enters a negative number, it's not added to the result
#include <stdio.h>
void main()
{
    int i;
    double number, sum = 0.0;

    for (i = 1; i <= 10; ++i)
     {
         printf("Enter a n%d: ", i);
         scanf("%lf", &number);

         if (number < 0.0)
             {
                 continue;
             }
             sum += number; // sum = sum + number;
    }
    printf("Sum = %.2lf", sum);
}
```

```
Enter a n1: 1.1
Enter a n2: 2.2
Enter a n3: 5.5
Enter a n4: 4.4
Enter a n5: -3.4
Enter a n6: -45.5
Enter a n7: 34.5
Enter a n8: -4.2
Enter a n9: -1000
Enter a n10: 12
Sum = 59.70
```

continue Statements

```c
#include<stdio.h>
int main()
{
int i=1,j=1;//initializing a local variable
for(i=1;i<=3;i++)
    {
        for(j=1;j<=3;j++)
        {
            if(i==2 && j==2)
            {
                continue;//will continue loop of j only
            }
            printf("%d %d\n",i,j);
        }
    }//end of for loop
return 0;
}
```

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

continue Statements

```c
#include<stdio.h>
int main()
{
int i=1,j=1;//initializing a local variable
for(i=1;i<=3;i++)
    {
        for(j=1;j<=3;j++)
        {
            if(i==2 && j==2)
            {
                continue;//will continue loop of j only
            }
            printf("%d %d\n",i,j);
        }
    }//end of for loop
return 0;
}
```

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

```c
//infinite loop
#include<stdio.h>
void main ()
{
    int i = 0;
    while(i!=10)
    {
        printf("%d", i);
        continue;
        i++;
    }
}
```

```c
/* Even Odd Using Continue */
#include <stdio.h>
int main()
{
 int i, number;
 printf("\n Please Enter any integer\n");
 scanf("%d", &number);

 for(i=1;i<= number; i++)
  {
   if(i%2 != 0)
   {
     continue;
   }
   printf("\n Even numbers = %d\n",i);
 }
}
```

```
 Please Enter any integer
10

 Even numbers = 2

 Even numbers = 4

 Even numbers = 6

 Even numbers = 8

 Even numbers = 10
```

1. **Write a program to read age of 100 persons and count the number if persons in the age group of 60 to 70. Use for and continue statements.**

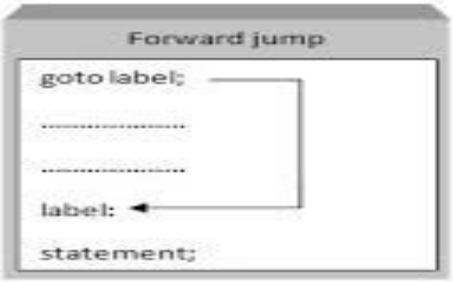
```c
int main()
{
    int i;
    for(i=1;i<=10;i++)
    {
        if(i==3)
        {
            continue;
        }
        if(i==5)
        {
            break;
        }
        printf("\n%d",i);
    }
return 0;
}
```

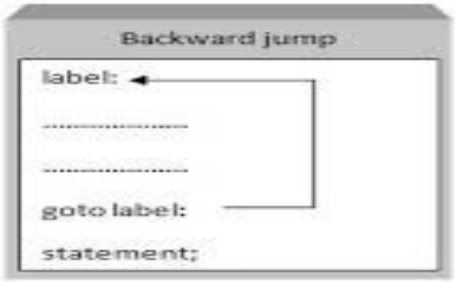
```
1
2
4
```

Difference between break & continue

break	continue
It terminates the execution of **remaining iteration** of the loop.	It terminates only the **current iteration** of the loop.
'break' resumes the control of the program to the end of loop enclosing that 'break'.	'continue' resumes the control of the program to the next iteration of that loop enclosing 'continue
It causes **early termination of loop**.	It causes **early execution of the next iteration.**
'break' **stops the continuation of loop.**	'continue' do not stops the continuation of loop, it only **stops the current iteration.**
A break can appear in both **switch** and **loop (for, while, do)** statements.	A continue can appear only in **loop (for, while, do)** statements.

- **Transfers control to labelled** statement.
- A goto statement in C programming provides an **unconditional jump** from the 'goto' to a labeled statement in the same function.





- **NOTE:** Use of goto statement is highly discouraged in any programming language because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. Any program that uses a goto can be rewritten to avoid them.

goto Statements

```c
#include <stdio.h>
int main()
{
    printf("hello\n");
    goto l1;
    printf("How are you\n");
    l1:
        printf("Hi\n");

    return 0;
}
```

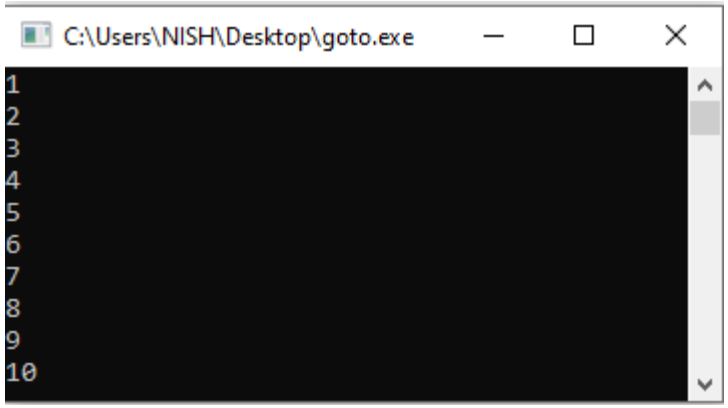
goto Statements

```c
#include <stdio.h>
int main()
{
    int number=1;

repeat:
    printf("%d\n",number);
    number++;

    if(number<=10)
        goto repeat;

    return 0;
}
```



In C, ”exit()” **terminates the calling process** (in our case, the C program) without executing the rest code which is after the exit() function.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    printf("START");
    exit(0);
    // The program is terminated here
    // This line is not printed
    printf("End of program");
}
```

```
START
```

Here, 0 is the exit status of the program which can be

exit()

```c
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int exit_status=10;
    printf("START");
    exit(exit_status);
    // The program is terminated here
    // This line is not printed
    printf("End of program");
}
```

```
START
```

We can take int variable instead of the exit staus.

It is also possible that we could use if...else statement in loop as and when required.

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

```
2
4
6
8
10
12
14
16
18
```

- An infinite loop is a looping construct that **does not terminate** the loop and **executes the loop forever**.
- It is also called an **indefinite** loop or an **endless** loop.

For loop

```
for(; ;)
{
    // body of the for loop.
}
```

while loop

```
while(1)
{
    // body of the loop..
}
```

do..while loop

```
do
{
    // body of the loop..
}while(1);
```

goto statement

```
infinite_loop;
// body statements.
goto infinite_loop;
```

End of Unit-06