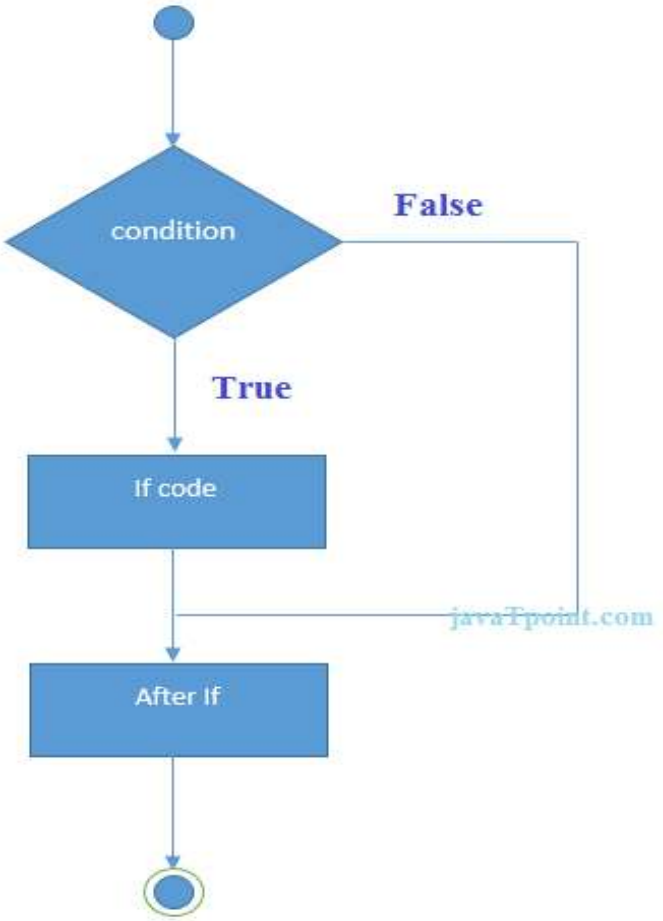
CONDITION STATEMENTS:

- **If Statement**
- The if statement is used to check some given condition and perform some operations depending upon the correctness of that condition. It is mostly used in the scenario where we need to perform the different operations for the different conditions.
  - **Syntax :**

```
if(expression)
{
//code to be executed
}
```

Flowchart of if statement in C:



Simple example of even number:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int number=0;
        printf("Enter a number:");
        scanf("%d",&number);
        if(number%2==0)
```

```
        {
        printf("%d is even number",number);
        }
        getch();
}
```

Program to find the largest number of the three.

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int a, b, c;
    printf("Enter three numbers?");
   scanf("%d %d %d",&a,&b,&c);
   if(a>b && a>c)
   {
     printf("%d is largest",a);
   }
   if(b>a  && b > c)
   {
     printf("%d is largest",b);
```

```
 }
    if(c>a && c>b)
    {
     printf("%d is largest",c);
    }
   if(a == b && a == c)
   {
      printf("All are equal");
    }
}
```

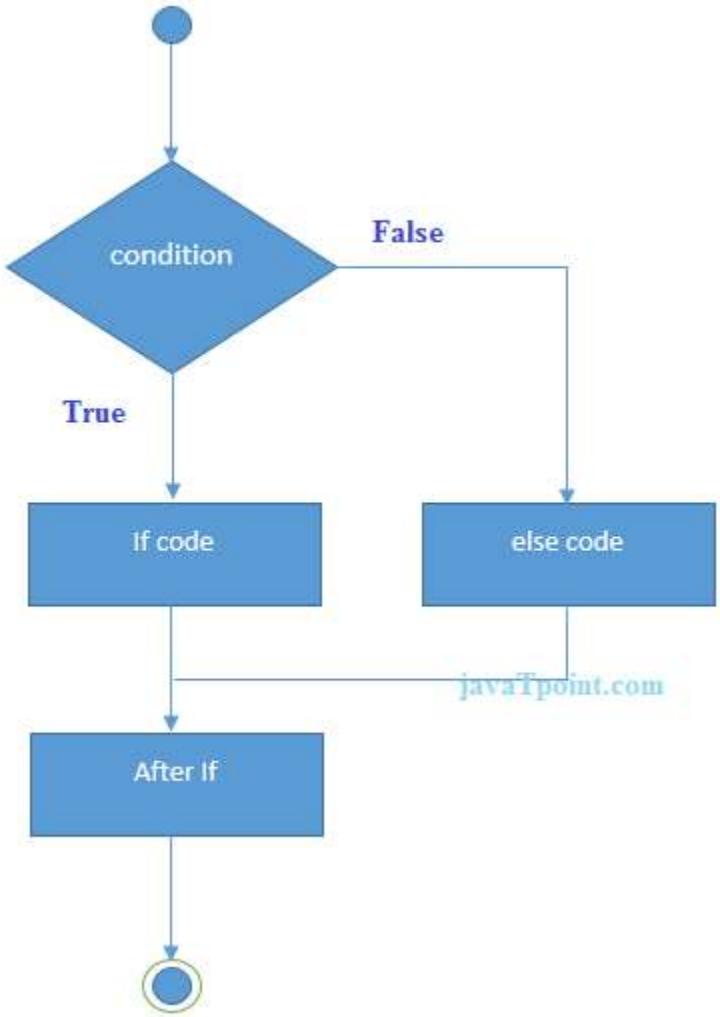
If-else Statement:

The if-else statement is used to perform two operations for a single condition. The if-else statement is an extension to the if statement using which, we can perform two different operations, i.e., one is for the correctness of that condition, and the other is for the incorrectness of the condition. Here, we must notice that if and else block cannot be executed simiulteneously. Using if-else statement is always preferable since it always invokes an otherwise case with every if condition.

Syntax:

```
if(expression)
{
//code to be executed if condition is true
}
else
{
//code to be executed if condition is false
}
```

Flowchart of the if-else statement in C:



Simple example to check whether a number is even or odd using if-else statement in C language.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int number=0;
        printf("enter a number:");
        scanf("%d",&number);
        if(number%2==0)
        {
        printf("%d is even number",number);
        }
        else
        {
        printf("%d is odd number",number);
        }
        getch();
}
```

Program to check whether a person is eligible to vote or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int age;
    printf("Enter your age?");
    scanf("%d",&age);
    if(age>=18)
    {
        printf("You are eligible to vote...");
    }
```

```
        else
    {
        printf("Sorry ... you can't vote");
    }
getch();
}
```

Task:

1. **Write a program for find out whether given no is equal or not.**
2. **Write a program for find out whether given no is divisiable by 5.**
3. **Write a program for find out minimum number from two different values.**

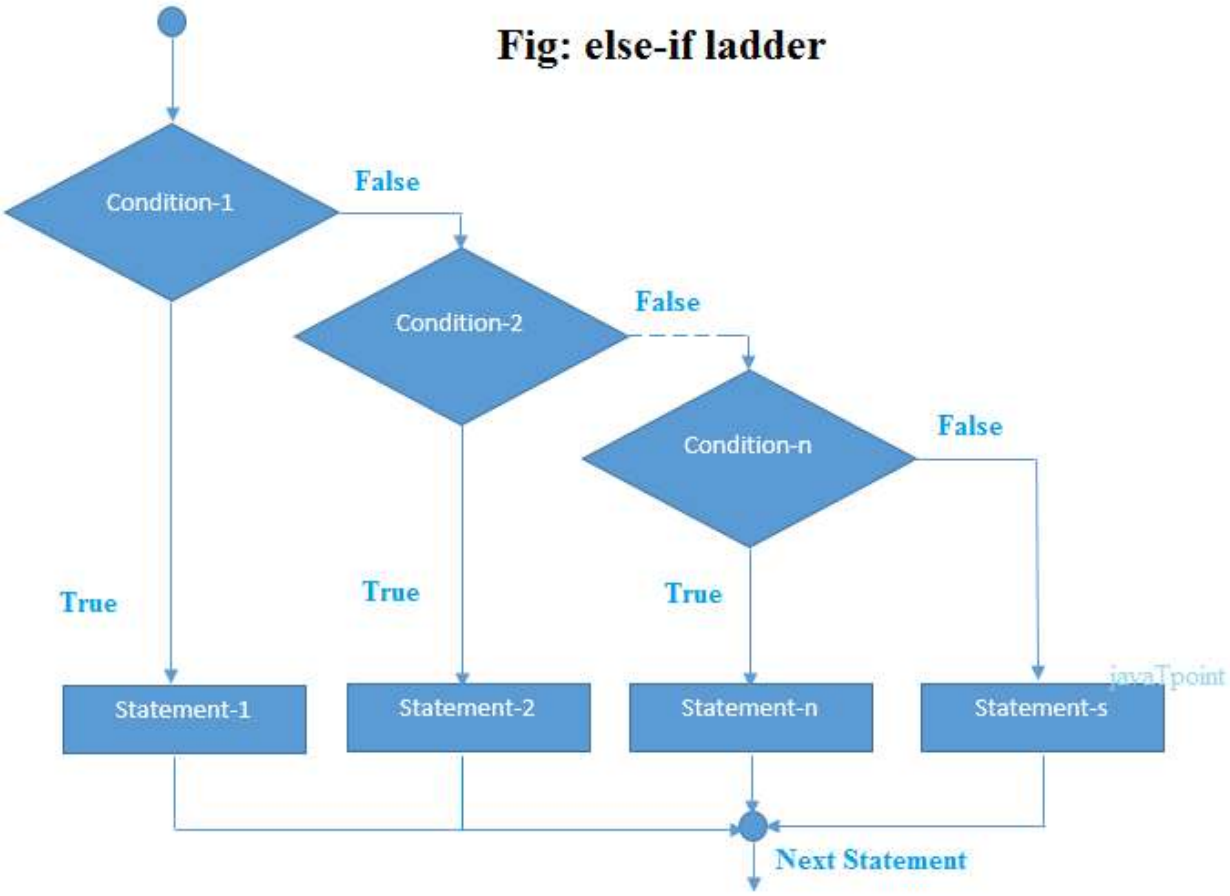
If else-if ladder Statement:

The if-else-if ladder statement is an extension to the if-else statement. It is used in the scenario where there are multiple cases to be performed for different conditions. In if-else-if ladder statement, if a condition is true then the statements defined in the if block will be executed, otherwise if some other condition is true then the statements defined in the else-if block will be executed, at the last if none of the condition is true then the statements defined in the else block will be executed. There are multiple else-if blocks possible. It is similar to the switch case statement where the default is executed instead of else block if none of the cases is matched.

Syntax:

```
if(condition1)
{
//code to be executed if condition1 is true
}
else if(condition2)
{
//code to be executed if condition2 is true
}
else if(condition3)
{
//code to be executed if condition3 is true
}
...
else
{
//code to be executed if all the conditions are false
}
```

Flowchart of else-if ladder statement in C



The example of an if-else-if statement:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int number=0;
printf("enter a number:");
scanf("%d",&number);
if(number==10)
{
printf("number is equals to 10");
}
else if(number==50)
```

```
{
printf("number is equal to 50");
}
else if(number==100)
{
printf("number is equal to 100");
}
else
{
printf("number is not equal to 10, 50 or 100");
}
getch();
}
```

Program to calculate the grade of the student according to the specified marks.

```
#include <stdio.h>
#include<conio.h>
void main()
{
   int marks;
   printf("Enter your marks?");
   scanf("%d",&marks);
   if(marks > 85 && marks <= 100)
   {
      printf("Congrats ! you scored grade A ...");
   }
   else if (marks > 60 && marks <= 85)
   {
      printf("You scored grade B + ...");
   }
   else if (marks > 40 && marks <= 60)
```

```
    {
        printf("You scored grade B ...");
    }
    else if (marks > 30 && marks <= 40)
    {
        printf("You scored grade C ...");
    }
    else
    {
        printf("Sorry you are fail ...");
    }
}
```

Task :C program to print weekday based on given number.

Task :C program to evaluate as per given criteria.

1. if a>b than d=a+b-c
2. if a>c than d=a+c-b
3. if b>c than d=b+c-a.

nested if statements

which means you can use one if or else if statement inside another if or else if statement(s).

Syntax:

```
if(expression 1)
{
  if(expression 2)
```

```
   {
     Executes code
   }
  else
  {
    Executes code
  }
else
{
    Executes code
}

}
```

Example :

```
int n1 = 10, n2 = 22, n3 = 25
if (n1 >= n2 )
{
    if (n1 >= n3)
    {
        print("The largest number is ", n1)
    }
    else
    {
        print("The largest number is ", n3)
    }
}
else
 {
    if (n2 >= n3)
    {
        print("The largest number is ",  n2)
    }
    else
```

```
    {
        print("The largest number is ", n3)
     }
}
```

Program for analysis of people of certain age groups who are eligible for getting a suitable job if their condition and norms get satisfied using nested if statement.

```
#include <stdio.h>

#include<conio.h>

void  main()

{

        int age;

        printf("Please Enter Your Age Here:\n");

        scanf("%d",&age);

        if ( age < 18 )

        {

                printf("You are Minor.\n");

                printf("Not Eligible to Work");

        }

        else
```

```
        {
        if (age >= 18 && age <= 60 )
        {
                printf("You are Eligible to Work \n");
                 printf("Please fill in your details and apply\n");
        }
        else
        {
        printf("You are too old to work as per the Government rules\n");
        printf("Please Collect your pension! \n");
        }
 }
getch();
}
```

Task :Program to take certain numbers as input from the user and then calculating from those numbers the largest and then giving the result whether or not it is greater or equal after manipulation with nested if statement.

Switch Statement:

The switch statement in C is an alternate to if-else-if ladder statement which allows us to execute multiple operations for the different possibles values of a single variable called switch variable. Here, We can define various statements in the multiple cases for the different values of a single variable.

syntax of switch statement :

```
switch(expression){
case value1:
 //code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
.....

default:
 code to be executed if all cases are not matched;
}
```

Rules for switch statement in C language

1) The *switch expression* must be of an integer or character type.

2) The *case value* must be an integer or character constant.

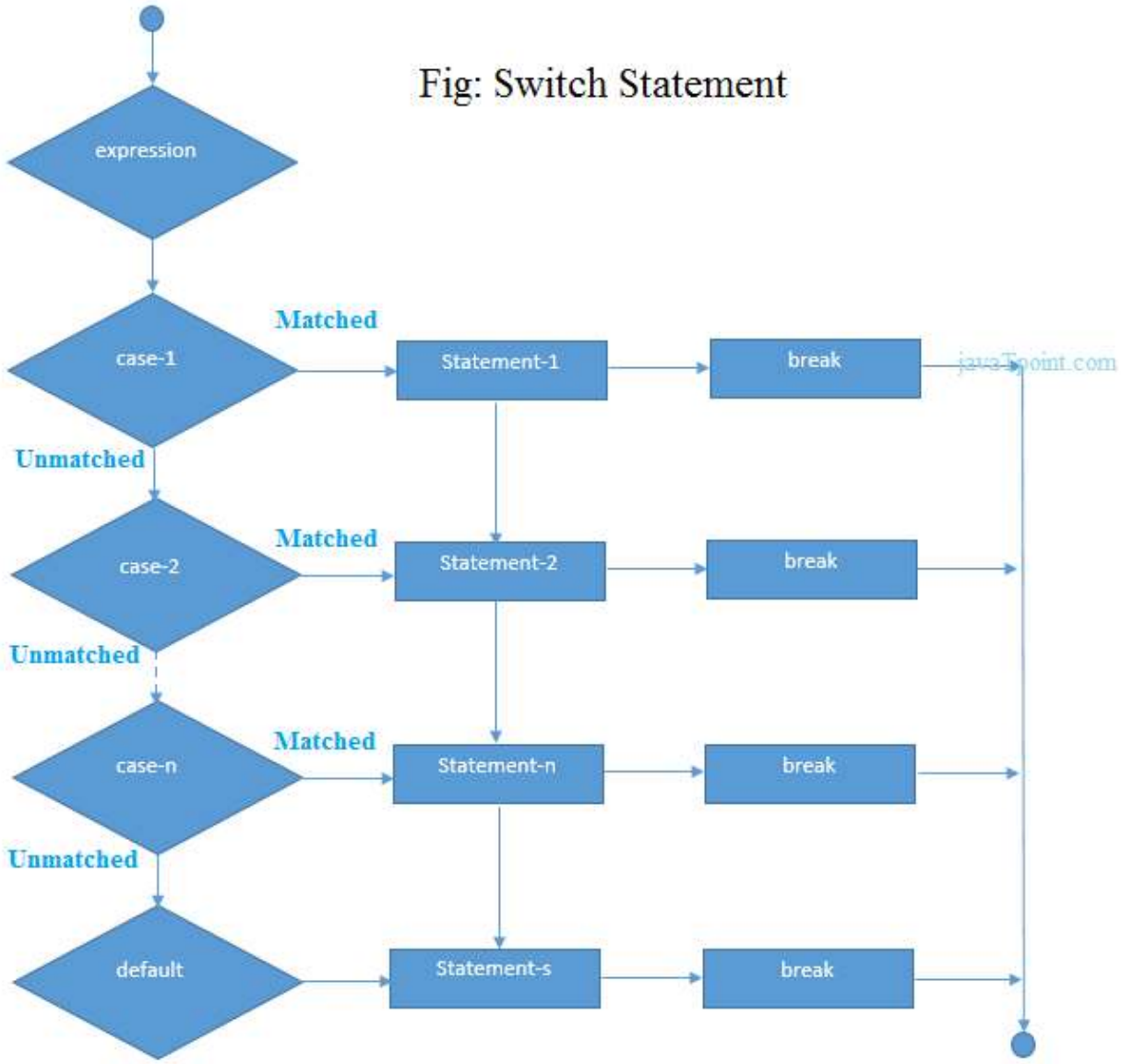
3) The *case value* can be used only inside the switch statement.

4) The *break statement* in switch case is not must. It is optional. If there is no break statement found in the case, all the cases will be executed present after the matched case. It is known as *fall through* the state of C switch statement.

```
int x,y,z;
char a,b;
float f;
```

Valid Switch	Invalid Switch	Valid Case	Invalid Case
switch(x)	switch(f)	case 3;	case 2.5;
switch(x>y)	switch(x+2.5)	case 'a';	case x;
switch(a+b-2)		case 1+2;	case x+2;
switch(func(x,y))		case 'x'>'y';	case 1,2,3;

Flowchart of switch statement in C



switch statement by the example given below.

```
#include<stdio.h>
#include<conio.h>
void main()
```

```
{
int number=0;

printf("enter a number:");
scanf("%d",&number);

switch(number)
{
        case 10:
        printf("number is equal to 10\n");
         break;
        case 50:
        printf("number is equal to 50\n");
        break;
        case 100:
        printf("number is equal to 100\n");
         break;
        default:
        printf("number is not equal to 10, 50 or 100");
}
getch();
}
```

Example -2

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int x = 10, y = 5;
   switch(x>y && x+y>0)
   {
      case 1:
      printf("hi");
      break;
      case 0:
```

```
            printf("bye");
            break;
            default:
            printf(" Hello bye ");
        }
            getch ();
    }
```

Examples:

1. Write a program for display day name according to day number.
2. Write a program for find out whether given character is vowel or not using switch case.
3. Write a program for find out whether given character is vowel or not using if else statement.
4. Write a program for find out whether given year is leap year or not using if-else statement.

Conditional ternary Operator

The conditional operator is also known as a **ternary operator**. The conditional statements are the decision-making statements which depends upon the output of the expression. It is represented by two symbols, i.e., '?' and ':'.

As conditional operator works on three operands, so it is also known as the ternary operator.

The behavior of the conditional operator is similar to the 'if-else' statement as 'if-else' statement is also a decision-making statement.

We use the ternary operator in C to run one code when the condition is true and another code when the condition is false.

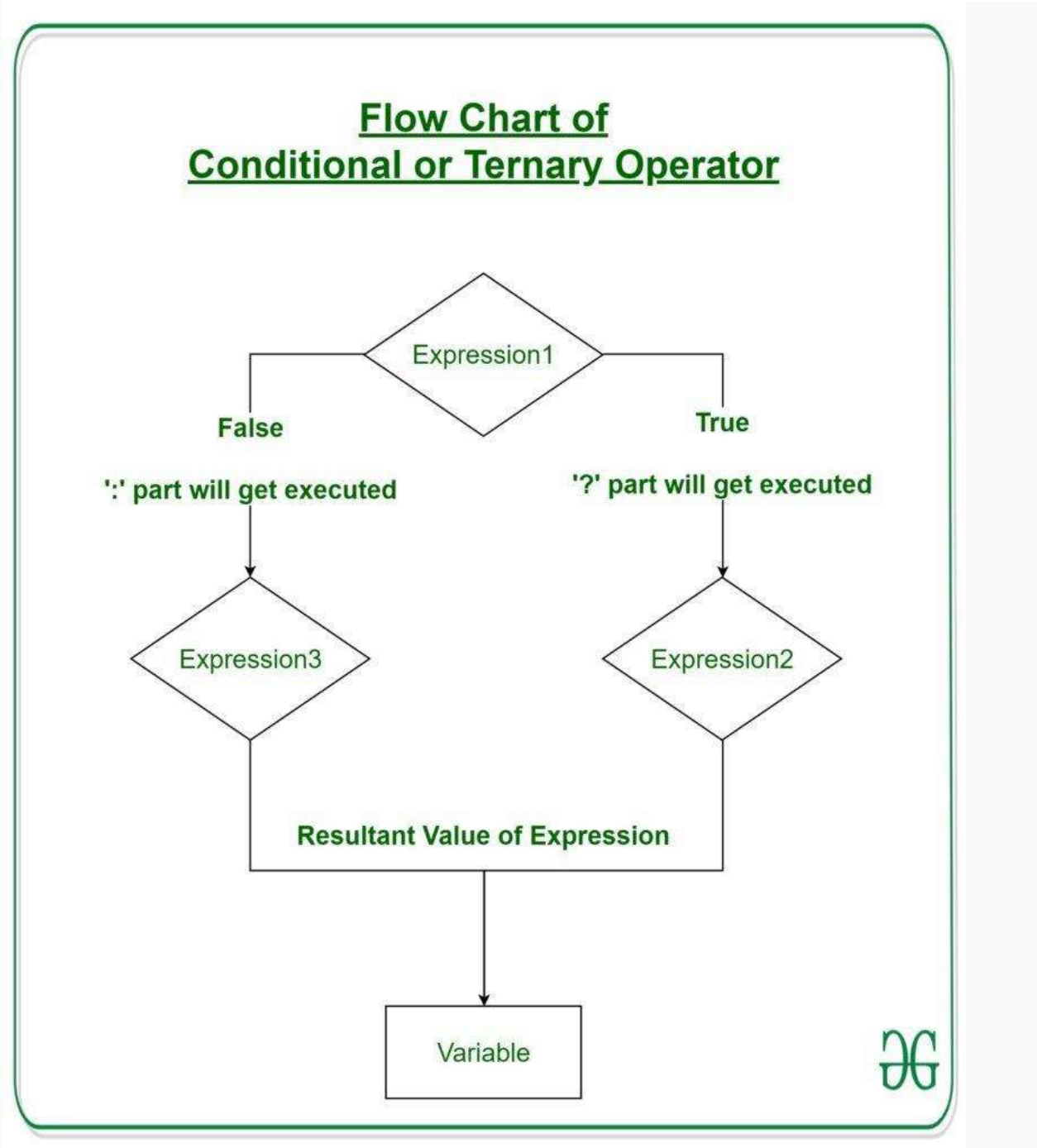
Syntax of Ternary Operator:

```
testCondition ? expression1 : expression 2;
```

The `testCondition` is a boolean expression that results in either **true** or **false**. If the condition is

- `true` - **expression1** (before the colon) is executed
- `false` - **expression2** (after the colon) is executed

The ternary operator takes 3 operands `(condition, expression1 and expression2)`. Hence, the name **ternary operator**.



Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
  int age;
  printf("Enter your age: ");
  scanf("%d", &age);

  // ternary operator to find if a person can vote or not
  (age >= 18) ? printf("You can vote") : printf("You cannot
vote");
  getch();
}
```

In the above example, we have used a ternary operator that checks whether a user can vote or not based on the input value. Here,

- `age >= 18` - test condition that checks if input value is greater or equal to **18**
- `printf("You can vote")` - **expression1** that is executed if condition is **true**
- `printf("You cannot vote")` - **expression2** that is executed if condition is **false**

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a=5,b;  // variable declaration
  b=((a==5)?(3):(2)); // conditional operator
  printf("The value of 'b' variable is : %d",b);
   getch();
}
```

In the above code, we have declared two variables, i.e., 'a' and 'b', and assign 5 value to the 'a' variable. After the declaration, we are assigning value to the 'b' variable by using the conditional operator. If the value of 'a' is equal to 5 then 'b' is assigned with a 3 value otherwise 2.

Example:
1.Write a program for find out whether given number is even or not using ternary operator.
2. write a program for find out whether given number is smallest or not using ternary operator.

C Loops

The looping can be defined as repeating the same process multiple times until a specific condition satisfies. There are three types of loops used in the C language. In this part of the tutorial, we are going to learn all the aspects of C loops.

Why use loops in C language?

The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times. For example, if we need to print the first 10 natural numbers then, instead of using the printf statement 10 times, we can print inside a loop which runs up to 10 iterations.

Advantage of loops in C

1) It provides code reusability.

2) Using loops, we do not need to write the same code again and again.

3) Using loops, we can traverse over the elements of data structures (array or linked lists).

Types of C Loops

There are three types of loops in C language

that is given below:

1. do while
2. while
3. for

do-while loop in C

The do-while loop continues until a given condition satisfies. It is also called post tested loop. It is used when it is necessary to execute the loop at least once (mostly menu driven programs).it is an exit-controlld loop.(An exit controlled loop is **that category of loops in which the test condition is checked after the execution of the body of the loop).**

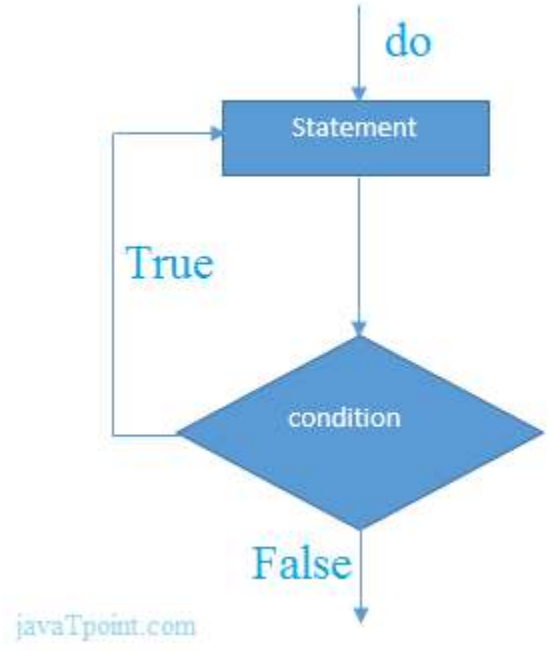
Syntax:

```
do{
//code to be executed
}while(condition);
```

How do...while loop works?

- The body of `do...while` loop is executed once. Only then, the `testExpression` is evaluated.
- If `testExpression` is **true**, the body of the loop is executed again and `testExpression` is evaluated once more.
- This process goes on until `testExpression` becomes **false**.
- If `testExpression` is **false**, the loop ends

Flowchart of do...while Loop



There is given the simple program of c language do while loop where we are printing the table of 1.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i=1;
        do{
        printf("%d \n",i);
        i++;
        }while(i<=10);
        getch();
}
```

Program to print table for the given number using do while loop

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int i=1,number=0;
        printf("Enter a number: ");
        scanf("%d",&number);
        do{
        printf("%d \n",(number*i));
        i++;
        }while(i<=10);
        return 0;
}
```

Example:

1. Write a program to calculate the sum of first 10 numbers using do-while loop.

while loop

While loop is also known as a pre-tested loop. In general, a 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.

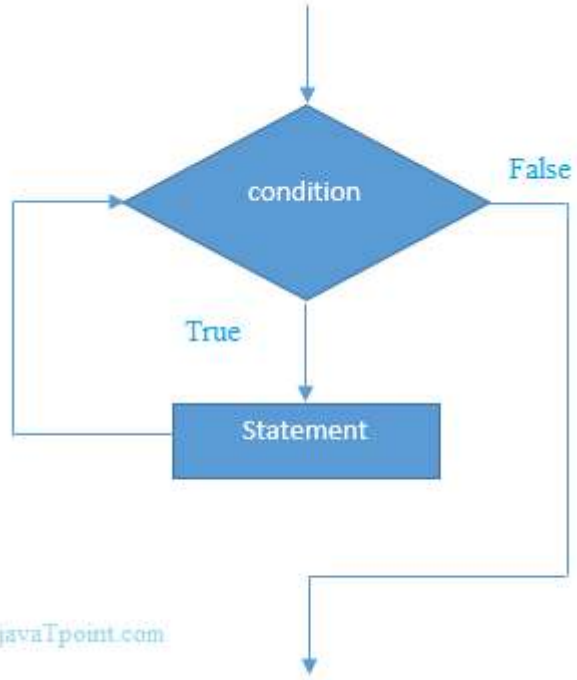
It is an entry controlled loop (Entry controlled loop is **a loop in which the test condition is checked first, and then the loop body will be executed**.)

While loop and for loop are entry controlled loop in c.

Syntax of while loop

```
while(condition){
//code to be executed
}
```

Flowchart of while loop in C



How while loop works?

- The `while` loop evaluates the `testExpression` inside the parentheses `()`.
- If `testExpression` is **true**, statements inside the body of `while` loop are executed. Then, `testExpression` is evaluated again.
- The process goes on until `testExpression` is evaluated to **false**.
- If `testExpression` is **false**, the loop terminates (ends).

Example:1

```
#include <stdio.h>
#include<conio.h>
void main ()
{
  int a = 10;
  while( a < 20 )
{
     printf("value of a: %d\n", a);
     a++;
  }
 getch();
}
```

Example:2

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int i=1,j=1;
   while (i <= 4 || j <= 3)
   {
 printf("%d %d\n",i, j);
 i++;
 j++;
   }
   getch();
}
```

Example:

1. Program to calculate factorial of a given number use while loop.

```
# include<stdio.h>
 # include<conio.h>
  void main ( )
```

```
{
long int n, fact =1;
clrscr( ) ;
printf( " Enter the Number:");
scanf("%ld", &n);
 while(n>=1)
 {
        fact = fact*n;
        n--;
 }
printf(" or factorial of given number is %d", fact);
 getch( );
}
```

Output:

Enter the Number : 5

Factorial of given number is 120

Differences between entry controlled and exit controlled loop

Entry Controlled Loop	Exit Controlled 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.
for loop and while loop are the examples of Entry Controlled Loop.	do while loop is the example of Exit controlled loop.

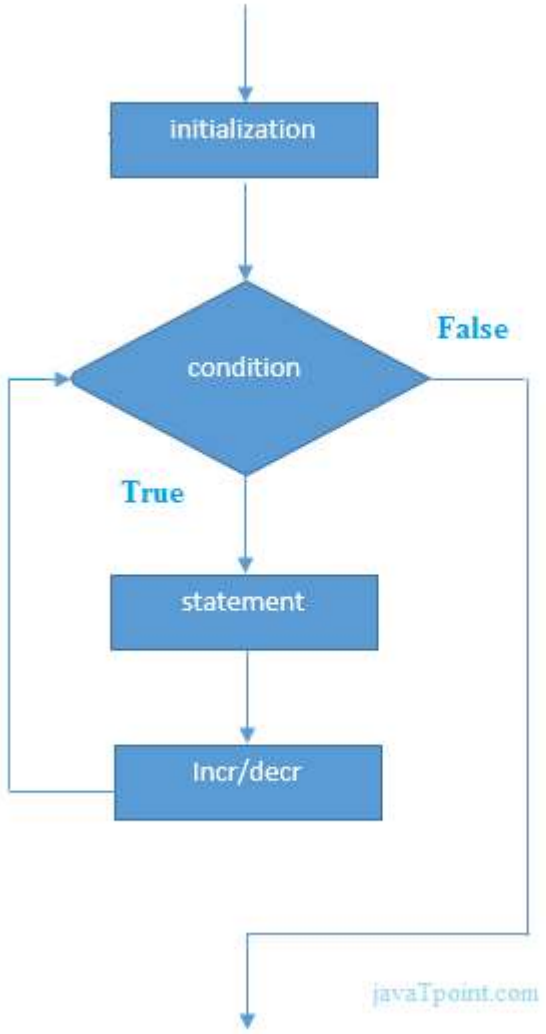
for loop in C:

The initialization statement describes the starting point of the loop, where the loop variable is initialized with a starting value. A loop variable or counter is simply a variable that controls the flow of the loop. The test expression is the condition until when the loop is repeated. It is entry control loop.

Syntax of for loop

```
for(initialization; test condition; increment/decrement){
//code to be executed
}
```

Flowchart of for loop in C



the simple program of for loop that prints table of 1.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=0;
for(i=1;i<=10;i++)
{
printf("%d \n",i);
}
```

```
getch();
}
```

Example:

Print the first five numbers starting from one together with their squares.

```
#include<stdio.h>
#include<conio.h>
void   main( )
{
        int i;
        clrscr( ) ;
        for(i = 1; i<=5; i++)
        printf("in Number: %d its Square: %d", i, i*i);
        getch( );
}
```

Output:

Number: 1 its Square: 1
Number: 2 its Square: 4
Number: 3 its Square: 9
Number: 4 its Square: 16
Number: 5 its Square: 25

Positive integers 1,2,3...n are known as natural numbers

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num, count, sum = 0;

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

```
    for(count = 1; count <= num; ++count)
    {
        sum += count;
    }

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

    getch();
}
```

The value entered by the user is stored in the variable `num`. Suppose, the user entered 10.

The `count` is initialized to 1 and the test expression is evaluated. Since the test expression `count<=num` (1 less than or equal to 10) is true, the body of `for` loop is executed and the value of `sum` will equal to 1.

Then, the update statement `++count` is executed and `count` will equal to 2.

Again, the test expression is evaluated. Since 2 is also less than 10, the test expression is evaluated to true and the body of the `for` loop is executed.

Now, `sum` will equal 3.

This process goes on and the sum is calculated until the `count` reaches 11.

When the `count` is 11, the test expression is evaluated to 0 (false), and the loop terminates.

Then, the value of `sum` is printed on the screen.

Write a C program to calculate the prime no of a given number:

```
1. #include <stdio.h>
2. int main()
3. {
4.
5.    int n, i, count = 0;
6.
7.     printf("Enter number to check prime number or not");
8.     scanf("%d",&n);
```

```
9.
10.        for(i=1; i<=n; i++)
11.       {
12.                if(n%i==0)
13.           {
14.              count++;
15.            }
16.        }
17.
18.        if (count==2)
19.           printf("%d is a prime number.",n);
20.        else
21.           printf("%d is not a prime number.",n);
22.
23.        getch();
24.     }
```

Write a C program to calculate the factorial of a given number:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,f=1,num;

  printf("Input the number : ");
  scanf("%d",&num);

  for(i=1;i<=num;i++)
      f=f*i;

  printf("The Factorial of %d is: %d\n",f);

}
```

Write a C program to check whether a given number is an armstrong number or not.

```
#include<stdio.h>

#include<conio.h>

     void main()

{

          int n,amr,r,c;

          printf("Input  a number: \n");

          scanf("%d",&n);

           c=n;

          while(n>0)

     {

                r=n%10;

                arm=(r*r*r)+arm;

                 n=n/10;

  }

if(c==arm)

                printf("Armstrong number”);

          else

                printf("Not Armstrong number”);

     }
```

Nested"for"loop:

• We can also use loop within loops. • i.e. one for statement within another/or statement is allowed in C.

Syntax:

```
for( initialize , test condition , updation) /* outer loop */
{
    for(initialize ; test condition ; updation) /* inner loop */
    {
    Body of loop;
     }
}
```

(1) **Write a program to display the stars as shown below**

```
*
**
***
****
******
#include<stdio.h>
#include<conio.h>
void  main ( )
{
        int x, i, j ;
        printf("How many lines stars (*) should be print..?");
```

```
        scanf("%d", &x);
        for(i=1;i<=x;i++)
        {
                for (j=1;j<=i;j++)
                 {
                        printf( "*");

                }

                printf( " \n");

        }

        getch( );

}
```

2) write a program print number pattern display below.

```
1
12
123
1234
12345
```

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

int main()
{
   int i, j;
   for(i=1;i<=5;i++)
   {
      for(j=1;j<=i;j++)
      {
         printf("%d",j);
      }
```

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

      return 0;
  }
```

write a program print number pattern display below.

```
1

23

456

78910

1112131415

#include <stdio.h>

int main()

{

   int i,j,k=1;

   for(i=1;i<=5;i++)

   {

      for(j=1;j<=i;j++)

      {

         printf("%d",k);

         k=k+1;

      }
```

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

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

```
#include <stdio.h>
#include<conio.h>
void main()
{
        Int i,j,k;
        for(i=1; i<=5; i++)
        {
                for(j=i;j<5;j++)
                { printf(" ");
                }
                for(k=1;k<=i;k++)
                {
```

```
                    printf(“*”);
                }
        }
        getch();
}
```

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

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

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

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

Jumping Statements:

(1) **The"break"Statement**:

• A break statement terminates the execution of the loop and the control is transferred to the statement immediately following the loop.

• i.e., the break statement is used to terminate loops or to exit from a switch.

• It can be used within a for, while, do-while, or switch statement.

• The break statement is written simply as break;

Example:

```
switch (choice)
{
    case "R":
    printf("Red");
    break;
    case "W":
    printf("White");
    break;
    case "B":
```

```
        printf("Blue");
        break;
        default:
        printf("An Error..");
    }
```

The last group does not require a break statement, since control will automatically be transferred out of the switch statement after the last group has been executed.

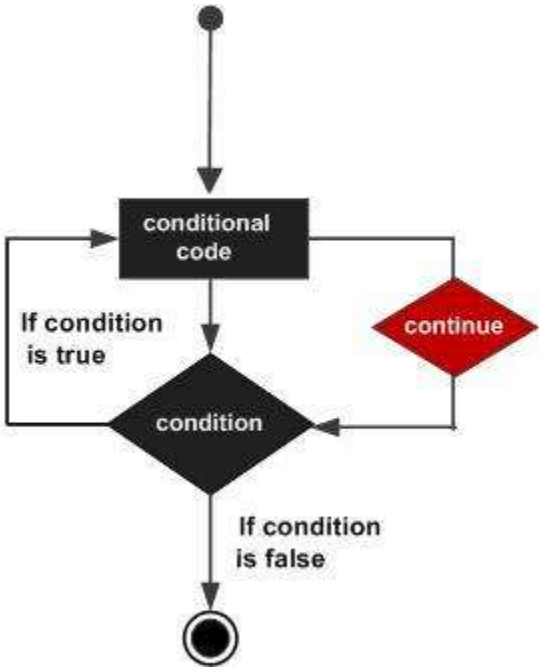
C continue statement

The **continue statement** in C language is used to bring the program control to the beginning of the loop. The continue statement skips some lines of code inside the loop and continues with the next iteration. It is mainly used for a condition so that we can skip some code for a particular condition.

Syntax:

```
//loop statements
continue;
//some lines of the code which is to be skipped
```

Flow chart:



Example:

```
#include<stdio.h>
int main(){
int i=1;//initializing a local variable
//starting a loop from 1 to 10
for(i=1;i<=10;i++){
if(i==5){//if value of i is equal to 5, it will continue the loop
continue;
}
printf("%d \n",i);
}//end of for loop
return 0;
}
```

C goto statement

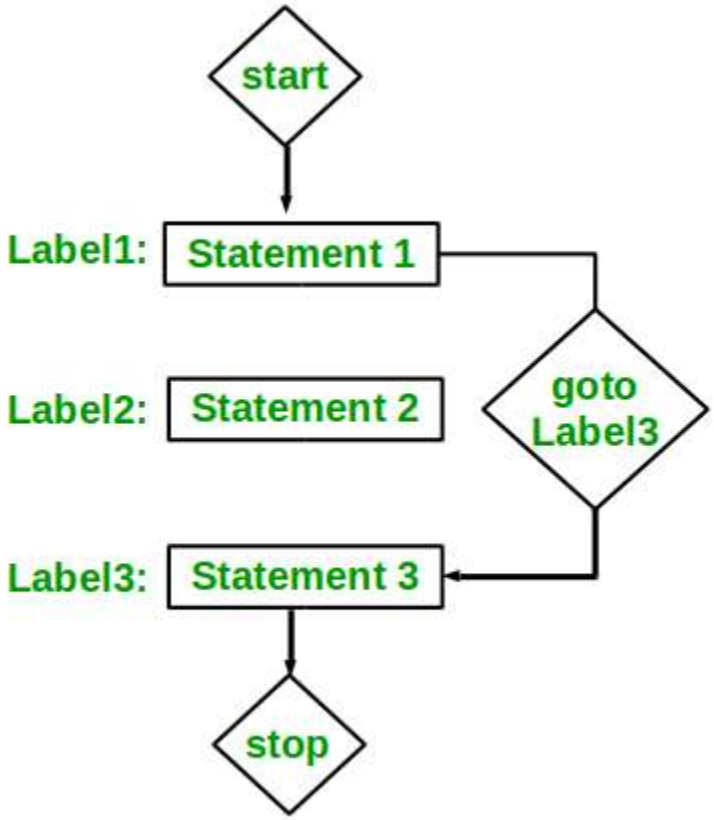
The goto statement is known as jump statement in C. As the name suggests, goto is used to transfer the program control to a predefined

label. The goto statment can be used to repeat some part of the code for a particular condition.

Syntax:

```
Syntax1       |   Syntax2
----------------------------
goto label;   |    label:
.             |    .
.             |    .
.             |    .
label:        |    goto label;
```

flow chart:



Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n=1;
        jump:
        printf("%d",n);
        n++;
```

```
        if(n<=10)
        goto jump;
        getch();
}
```