**UNIT-2**

**Conditional Statements**

C if else Statement

The if-else statement in C is used to perform the operations based on some specific condition. The operations specified in if block are executed if and only if the given condition is true.

There are the following variants of if statement in C language.

* If statement
* If-else statement
* If else-if ladder
* Nested if

## 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. The syntax of the if statement is given below.

1. **if**(expression){
2. //code to be executed
3. }

**Flowchart of if statement in C**



Let's see a simple example of C language if statement.

#include<stdio.h>

**int** main(){

**int** number=0;

printf("Enter a number:");

scanf("%d",&number);

**if**(number%2==0){

printf("%d is even number",number);

}

**return** 0;

}

**Output**

Enter a number:4

4 is even number

enter a number:5

### Program to find the largest number of the three.

#include <stdio.h>

**int** 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");

    }

}

**Output**

Enter three numbers?

12 23 34

34 is largest

## 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. The syntax of the if-else statement is given below.

1. **if**(expression){
2. //code to be executed if condition is true
3. }**else**{
4. //code to be executed if condition is false
5. }

**Flowchart of the if-else statement in C**



Let's see the simple example to check whether a number is even or odd using if-else statement in C language.

#include<stdio.h>

**int** 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);

}

**return** 0;

}

**Output**

enter a number:4

4 is even number

enter a number:5

5 is odd number

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

#include <stdio.h>

**int** 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");

    }

}

**Output**

Enter your age?18

You are eligible to vote...

Enter your age?13

Sorry ... you can't vote

## f 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.

**if**(condition1){

}**else** **if**(condition2){

}

**else** **if**(condition3){

}

...

**else**{

}

**Flowchart of else-if ladder statement in C**



The example of an if-else-if statement in C language is given below.

#include<stdio.h>

**int** 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");

}

**return** 0;

}

**Output**

**enter a number:4**

**number is not equal to 10, 50 or 100**

**enter a number:50**

**number is equal to 50**

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

**#include <stdio.h>**

**int 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 ...");**

**}**

**}**

**Output**

**Enter your marks?10**

**Sorry you are fail ...**

**Enter your marks?40**

**You scored grade C ...**

**Enter your marks?90**

**Congrats ! you scored grade A ...**

# C 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.

The syntax of switch statement in [c language](https://www.javatpoint.com/c-programming-language-tutorial) is given below:

**switch**(expression){

**case** value1:

**break**;

**case** value2:

**break**;

......

**default**:

 code to be executed **if** all cases are not matched;

}

### Rules for switch statement in C language

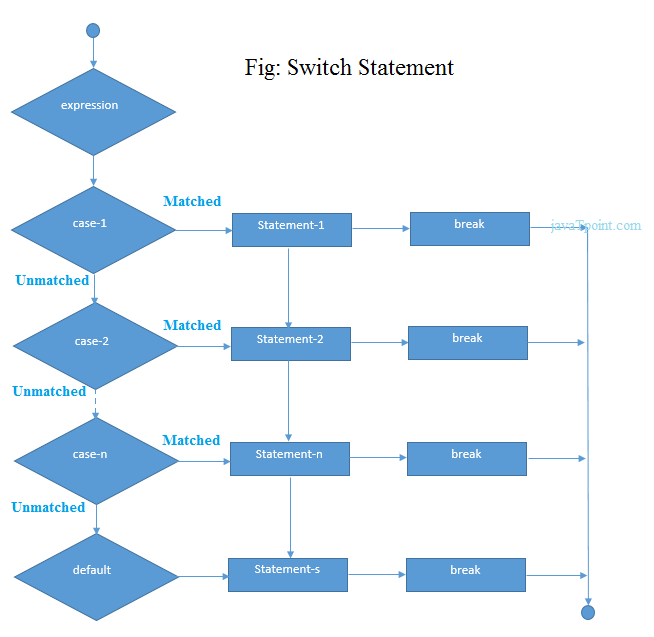
1) The *switch expression* must be of an integer or character type.

2) The *case value* must be an integer or character constant.Backward Skip 10sPlay VideoForw

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.

#### Flowchart of switch statement in C



#include<stdio.h>

**int** main(){

**int** number=0;

printf("enter a number:");

scanf("%d",&number);

**switch**(number){

**case** 10:

printf("number is equals to 10");

**break**;

**case** 50:

printf("number is equal to 50");

**break**;

**case** 100:

printf("number is equal to 100");

**break**;

**default**:

printf("number is not equal to 10, 50 or 100");

}

**return** 0;

}

**Output**

enter a number:4

number is not equal to 10, 50 or 100

enter a number:50

number is equal to 50

# 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](https://www.javatpoint.com/c-programming-language-tutorial) that is given below:

1. do while
2. while
3. for

# do while loop in C

The do while loop is a post tested loop. Using the do-while loop, we can repeat the execution of several parts of the statements. The do-while loop is mainly used in the case where we need to execute the loop at least once. The do-while loop is mostly used in menu-driven programs where the termination condition depends upon the end user.

#### do while loop syntax

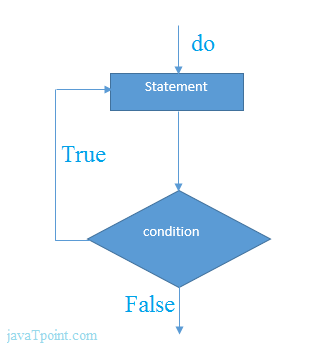
The syntax of the C language do-while loop is given below:

**do**{

//code to be executed

}**while**(condition);

#### Flowchart of do while loop

;

#### do while example

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

#include<stdio.h>

**int** main(){

**int** i=1;

**do**{

printf("%d \n",i);

i++;

}**while**(i<=10);

**return** 0;

}

#### Output

1

2

3

4

5

6

7

8

9

10

# while loop in C

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.

#### Syntax of while loop in C language

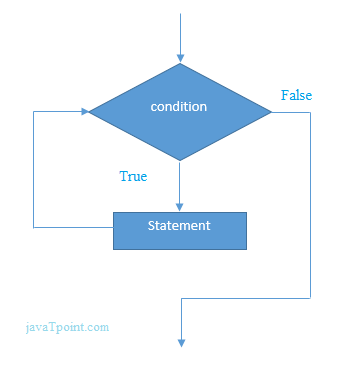
The syntax of while loop in c language is given below:

**while**(condition){

//code to be executed

}

#### Flowchart of while loop in C



## Example of the while loop in C language

Let's see the simple program of while loop that prints table of 1.

#include<stdio.h>

**int** main(){

**int** i=1;

**while**(i<=10){

printf("%d \n",i);

i++;

}

**return** 0;

}

#### Output

1

2

3

4

5

6

7

8

9

10

## Program to print table for the given number using while loop in C

#include<stdio.h>

**int** main(){

**int** i=1,number=0,b=9;

printf("Enter a number: ");

scanf("%d",&number);

**while**(i<=10){

printf("%d \n",(number\*i));

i++;

}

**return** 0;

}

#### Output

Enter a number: 50

50

100

150

200

250

300

350

400

450

500

## Properties of while loop

* A conditional expression is used to check the condition. The statements defined inside the while loop will repeatedly execute until the given condition fails.
* The condition will be true if it returns 0. The condition will be false if it returns any non-zero number.
* In while loop, the condition expression is compulsory.
* Running a while loop without a body is possible.
* We can have more than one conditional expression in while loop.
* If the loop body contains only one statement, then the braces are optional.

# for loop in C

The **for loop in C language** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like the array and linked list.

### Syntax of for loop in C

The syntax of for loop in c language is given below:

1. **for**(Expression 1; Expression 2; Expression 3){
2. //code to be executed
3. }

### Flowchart of for loop in C



### C for loop Examples

Let's see the simple program of for loop that prints table of 1.

#include<stdio.h>

**int** main(){

**int** i=0;

**for**(i=1;i<=10;i++){

printf("%d \n",i);

}

**return** 0;

}

**Output**

1

2

3

4

5

6

7

8

9

10