**UNIT 2**

Decision making structures require that the programmer specifies one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following are the decision making statements

* **simple – if statement**
* **if – else statement**
* **nested – if else statement**
* **else – if ladder**
* **switch statement**

## Simple – if statement

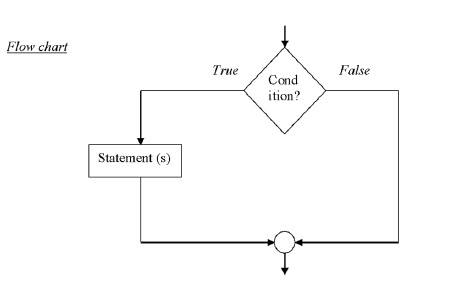
The ‘if’ keyword is used to execute a set of statements when the logical condition is true.

### Syntax

if (condition){

   Statement (s)

}



**Example**

**The following example checks whether a number is greater than 50.**

#include<stdio.h>

main(){

   int a;

   printf(“enter any number:  
”);

   scanf(“%d”,&a);

   if(a>50)

      printf(“%d is greater than 50”, a);

}

## Output

1) enter any number: 60

60 is greater than 50 .

2) enter any number 20

no output

## The if else statement

The if –else statement takes either True or False condition.

### Syntax

if (condition){

   True block statement(s)

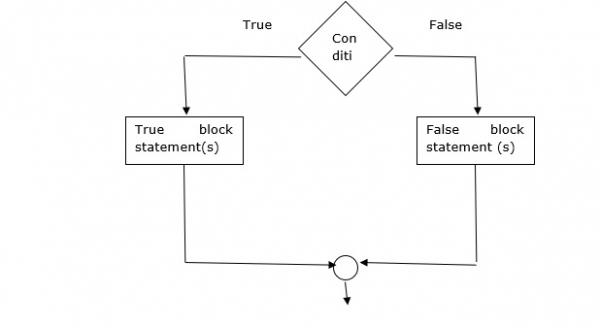
}

else{

   False block statement(s)

}

### Flowchart



## Example

Following is the program to check for even or odd number −

#include<stdio.h>

main(){

   int n;

   printf(“enter any number:  
”);

   scanf(“%d”,&n);

   if(n%2==0)

      printf(“%d is even number”, n);

   else

      printf(“%d is odd number”, n);

}

## Output

1) enter any number: 10

10 is even number

## Nested if - else statement

Here ‘if’ is placed inside another if (or) else −

### Syntax

if (condition1){

   if (condition2)

      stmt1;

   else

      stmt2;

   }

   else{

      if (condition3)

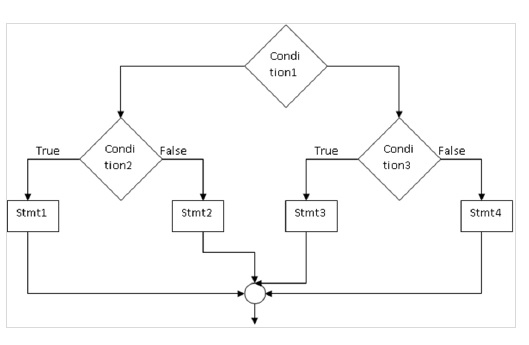
         stmt3;

      else

         stmt4;

   }

### Flowchart



## Example

Following example is to print the largest of 3 numbers from the given numbers −

#include<stdio.h>

main(){

   inta,b,c;

   printf(“enter 3 numbers”);

   scanf(“%d%d%d”,&a,&b,&c);

   if(a>b){

      if(a>c)

         printf(“%d is largest”, a);

      else

         printf(“%d is largest”, c);

   }else{

      if(b>c)

         printf(“%d is largest”, b);

      else

         printf(“%d is largest”, c);

   }

}

## Output

enter 3 numbers = 10 20 30

30 is largest

## Else – if ladder

It is a multi-way decision condition.

### Syntax

if (condition1)

   stmt1;

else if (condition2)

   stmt2;

   - - - - -

   - - - - -

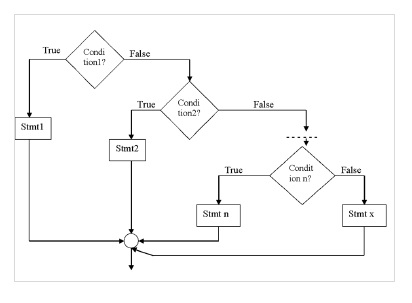
else if (condition n)

   stmt n;

else

   stmt x;

### Flowchart



## Example

Following example finds roots of quadratic equation −

#include<math.h>

main(){

   inta,b,c,d;

   float r1, r2

   printf("enter the values a b c");

   scanf(“%d%d%d”,&a,&b,&c);

   d= b\*b –4\*a\*c ;

   if(d>0){

      r1 =(-b+sqrt(d))/(2\*a);

      r2 =(-b-sqrt(d))/(2\*a);

      printf(“root1 ,root2 =%f%f”, r1, r2);

   }

   elseif(d==0){

      r1 =-b /(2\*a);

      r2 =-b/(2\*a);

   printf(“root1, root2 =%f%f”, r1, r2);

   }

   else

      printf("roots are imaginary”);

}

## Output

1) enter the values of a b c : 1 4 3

Root 1 = -1

Root 2 = -3

## Switch statement

It is helpful in selecting one among multiple decisions.

### Syntax

switch (expression){

   case value1 : stmt1;

      break;

   case value2 : stmt2;

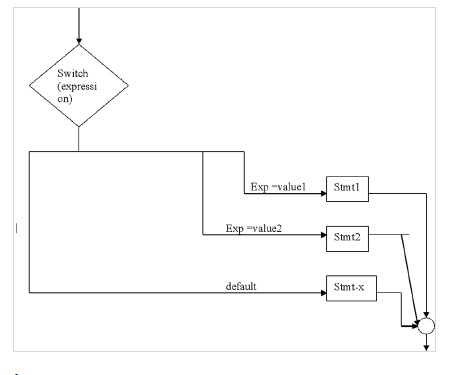
      break;

   - - - - - -

   default :stmt – x;

}

### Syntax



## Example

#include<stdio.h>

main(){

   int n;

   printf(“enter a number”);

   scanf(“%d”,&n);

   switch(n){

      case0:printf(“zero”)

         break;

      case1:printf(‘one”);

         break;

      default:printf(‘wrong choice”);

   }

}

## Output

enter a number

1

One

A **while** loop in C programming repeatedly executes a target statement as long as a given condition is true.

## Syntax

The syntax of a **while** loop in C programming language is −

while(condition) {

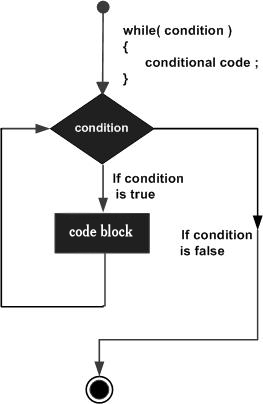
statement(s);

}

Here, **statement(s)** may be a single statement or a block of statements. The **condition** may be any expression, and true is any nonzero value. The loop iterates while the condition is true.

When the condition becomes false, the program control passes to the line immediately following the loop.

## Flow Diagram



Here, the key point to note is that a while loop might not execute at all. When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

## Example

#include<stdio.h>

int main (){

/\* local variable definition \*/

int a =10;

/\* while loop execution \*/

while( a <20){

printf("value of a: %d\n", a);

a++;

}

return0;

}

When the above code is compiled and executed, it produces the following result −

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

value of a: 16

value of a: 17

value of a: 18

value of a: 19

Syntax

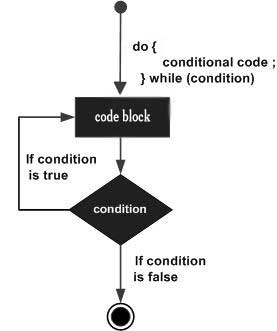
The syntax of a **do...while** loop in C programming language is −

do {

statement(s);

} while( condition );

Flow Diagram



## Example

[Live Demo](http://tpcg.io/pVAh6H)

#include <stdio.h>

int main () {

/\* local variable definition \*/

int a = 10;

/\* do loop execution \*/

do {

printf("value of a: %d\n", a);

a = a + 1;

}while( a < 20 );

return 0;

}

**Output**

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

value of a: 16

value of a: 17

value of a: 18

value of a: 19

**The For Statement**

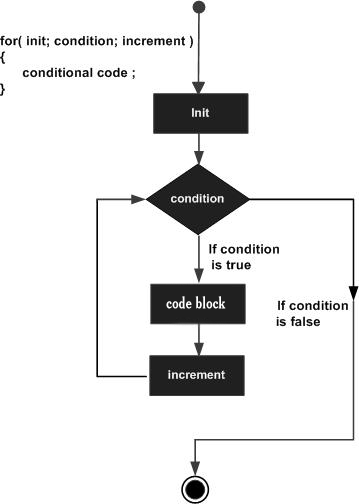
The syntax of a **for** loop in C programming language is −

for ( init; condition; increment ) {

statement(s);

}

## Flow Diagram



## Example

#include <stdio.h>

int main () {

int a;

/\* for loop execution \*/

for( a = 10; a < 20; a = a + 1 ){

printf("value of a: %d\n", a);

}

return 0;

}

**Output**

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

value of a: 16

value of a: 17

value of a: 18

value of a: 19

**Jumps in Loops**

## Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

C supports the following control statements.

|  |  |
| --- | --- |
| **Sr.No.** | **Control Statement & Description** |
| 1 | [break statement](https://www.tutorialspoint.com/cprogramming/c_break_statement.htm)  Terminates the **loop** or **switch** statement and transfers execution to the statement immediately following the loop or switch. |
| 2 | [continue statement](https://www.tutorialspoint.com/cprogramming/c_continue_statement.htm)  Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating. |
| 3 | [goto statement](https://www.tutorialspoint.com/cprogramming/c_goto_statement.htm)  Transfers control to the labeled statement. |

## The Infinite Loop

A loop becomes an infinite loop if a condition never becomes false. The **for** loop is traditionally used for this purpose. Since none of the three expressions that form the 'for' loop are required, you can make an endless loop by leaving the conditional expression empty.

#include <stdio.h>

int main () {

for( ; ; ) {

printf("This loop will run forever.\n");

}

return 0;

}

When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but C programmers more commonly use the for(;;) construct to signify an infinite loop.