CONTROL STATEMENTS

In a C program the instructions are executed sequentially in the order in which they appear. This happens when no option or the repetition of some instructions are necessary. In such case each instruction is executed once and only once. In practice, we may have a number of situations where the order of execution of the statement is changed based on the certain conditions. This involves a kind of decision making to see whether a given condition is true or not. Control structures are used in C program to check the given condition and to execute certain statement accordingly.

C has three types of control statements:-

1. Decision making and branching statements
   a. if…….
   b. if…….else
   c. nested if……else
   d. else….if ladder
   e. conditional statement
   f. switch…..case

2. Decision making and looping statements
   a. while
   b. do…..while
   c. for

3. Control statements without decision making or jump statements
   a. break;
   b. continue;
   c. goto;
   d. return;

1. Decision making and branching statement:

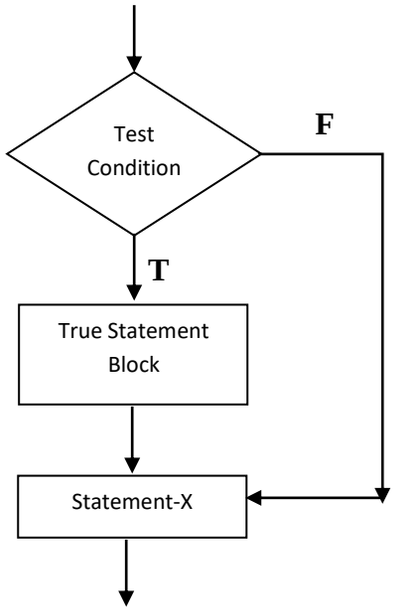
a). if….statement- The if… statement is a powerful decision making statement and is used to control the flow of execution of statements. The general form of a simple if statement is as follows:-

```
if(text_expression)
  {
   statement block;
   }
statement-X
```

In a general form of if statement, the statement block may be a single statement or a group of statements. If the test expression is found to be true the statement block is executed otherwise it is skipped.

The test expression of the if statement may be a relational or logical expression. The result of relational and logical expression is evaluated first and if statements take action according to the result.

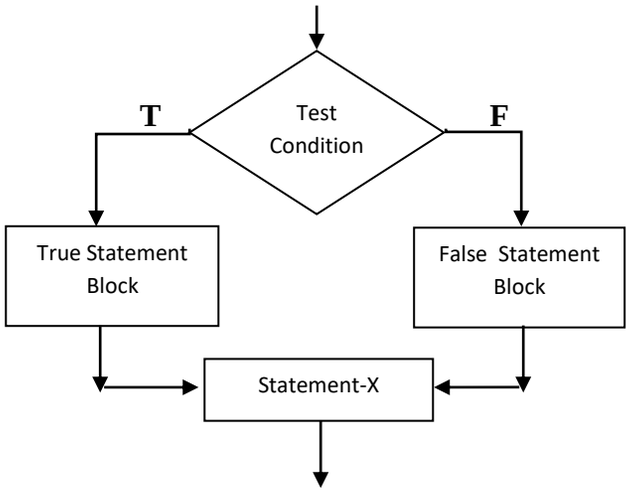
The flowchart of if statement is as follows:



b). if…..else statement:- The if….else statement is two way decision making statement. The general form of a simple if...else statement is as follows:-

```
if(test expression)
   {
    true statement block;
   }
else
   {
    false statement block;
   }
Statement-X;
```

The flowchart of if…..else statement is as follows:



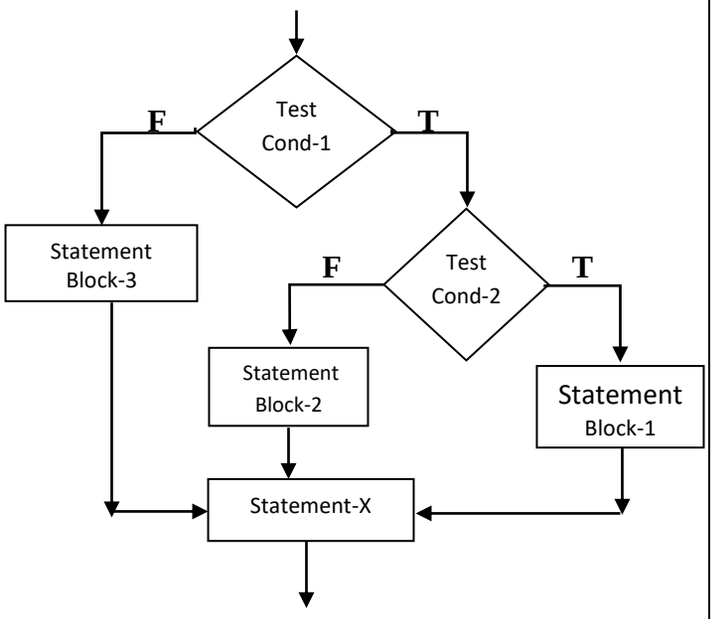
In if….else statement, the test expression is evaluated first if it is found to be true, the true statement block is executed and the false statement block is skipped. Otherwise, the false statement block is executed and true statement block is skipped. In no case both the blocks executed and in no case both the blocks are skipped. The statement block may have one or more than one executable statements. The test expression may be relational or logical expression.

In if…else, the statement-X is get executed only after the execution of one of the block specified in if…else statement.

c). nested if…..else statement:- When there are number of decisions involved in a series, we use more than one if…else statement in nested form as shown below:-

```
if(test expression-1)
        {
         if(test expression-2)
              {
               statement block-1;
               }
         else
               {
                statement block-2;
               }
         }
    else
         {
          statement block-3;
         }
```

The flowchart of nested if…..else statement is as follows:



In nesting of if…..else statement there are two or more than two if statements in series which are evaluated in chain and the particular statement block is executed according to the decisions.

Nested if..…else statement is a multi-way decision making and branching statement.

Disadvantage of nested if….else statement:-

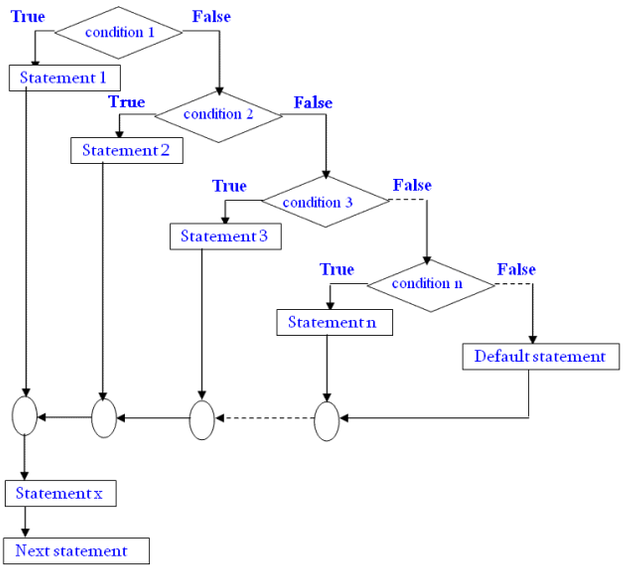
i. As the number of condition goes on increasing, the level of indentation also goes on increasing. As a result the whole program creeps to the right.
ii. Care needs to be taken to match the corresponding if's and else's
iii. Care needs to be taken to match the corresponding pair of braces.

d). else…if ladder:- Here is another way of putting multiple if's together when multipath decision is involved. A multipath decision is a chain of if in which the statement associated with each else is an if, such multipath decision is called else….if ladder.

The general form of else….if ladder is given below-

```
if(test condition-1)
        {
        statement block-1;
        }
else if(test condition-2)
        {
        statement block-2;
        }
        ----------------------------------
        ----------------------------------
 else if(test condition-n)
        {
        statement block-n;
        }
 else
       {
       default statement block;
        }
```

The flowchart of else…if ladder is as follows:



e). switch….case statement:- In nested if…else or else…if ladder the complexity and indentation of program increases dramatically when the number of alternatives increase. The program becomes difficult to read and follow. In such a situation the program becomes confusing for the programmer also. To avoid such situation, C provides us another multi-way decision making and branching statement known as switch……case statement.

The switch…….case statement tests the value of a given expression/variable against a list of case values and when a match is found, a block of statements associated with that case is executed.

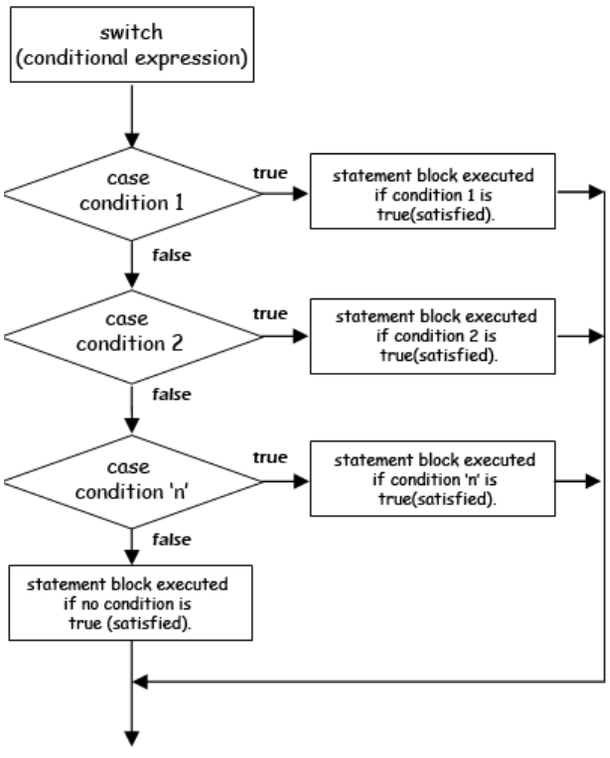
The general form switch…case statement is as follows:-

```
switch(expression)
        {
         case: casevalue-1: statement block-1;
```

```
        case: casevalue-2: statement block-2;
        case: casevalue-3: statement block-3;
        ---------------------------------
        ---------------------------------
        case: casevalue-n: statement block-n;
        default: default block;
        }
    statement-X;
```

When the switch is executed then the value of the expression is successfully compared against the casevalue-1, casevalue-2, ……., casevalue-n. If the match is found the block of statement following that case are executed. The break statement at the end of each block signals the end of the particular case and causes an exit from the switch statement transferring the control to the statement-X just following the switch case statement.

The flowchart of switch…..case statement is as follows:



Rules for switch statement:-

a) The switch expression must be of integral type.
b) Case labels must be a constant or a constant expression.
c) Case label must be unique.
d) The break statement transfers the control out of the switch statement.
e) The break is optional.
f) There can be 257 case labels in a switch case statement.
g) ANSI C allows 15 levels of nesting of switch case statement i.e. up to 15 level switch case may be a part of case block.
h) There can be at most one default label in a switch case statement.
i) The default may be placed anywhere in a switch case statement but placed at the end switch case to use properly.

f) Conditional Operator(?:):- The C language has a ternary operator useful for making two way decision like if…else. This operator is combination of question mark[?] and colon[:] and takes three operands. This operator is popularly known as conditional operator.

The general form of the operator is as follows:-

conditional_exp?exp-1:exp-2;

In conditional operator the conditional expression is evaluated first if the result is non-zero, exp-1 is evaluated and is returned as the value of the conditional expression otherwise exp-2 is evaluated and its value is returned.

```
z = x>y ? x-y : y-x;
x>y ? printf("x is greater"):printf("y is greater");
```

The conditional operator may be nested to evaluate more complex assignment decision. The use of conditional operator increases the conciseness of the program but decreases the readability. So it is recommended to use nested if… else statement when more than a signal nesting of conditional operator is required.

2. Decision making and looping statement:-

The versatility of the computer lies in its ability to execute instructions repeatedly. This involves execution of a part of a program either a specified number of the time or until a particular condition is being satisfied. The repetitive operation is done through a loop control structure.

A loop control structure tests certain conditions and then directs the repeated execution of the statements contained in the body of loop.

Types of loop control structure:-

Depending on the position of test condition in a control statement, control structures may be classified into two categories:-

1. **Entry controlled loop:** In the entry control loop the condition is tested before the start of the loop execution. If the condition is not satisfied(true), the body of loop will not be executed. The body of loop execute rapidly until the condition remains true. In C while and for loop are considered to be entry controlled loop.
2. **Exit controlled loop:** In an exit control loop test is performed at the end of the body of the loop and the loop is repeated until the condition at the exit remains true therefore in exit control loop the body of loop is executed unconditionally for the first time. The do…while() structure in C is an example of the exit control loop structure.

Depending on the number of repetition of a loop control structures are classified into two categories:-

1. **Counter controlled loop:-** In a counter controlled loop number of repetition of a loop is specified in the loop control structure and after specified repetitions the loop stops its execution.
2. **Sentinel controlled loop:-** In sentinel control loop the number of repetition depend upon the condition occurs during program execution. A sentinel control loop may execute one time, more than one time or the large number of times or not even a single time.

Loop Control Structures available in C:- There are three loop control structures available in C.

a). while( ):- The while statement is most simplest loop structure in C language. The general form of while **statement is as follows:-**

```
while(test condition)
        {
         body of loop;
  }
    statement-X;
```

The flowchart of while() statement is as follows:

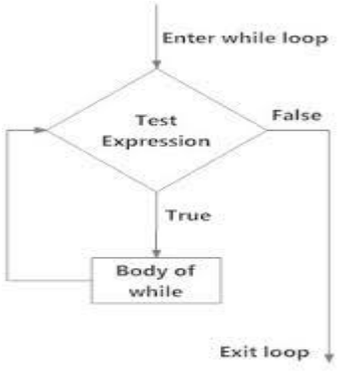
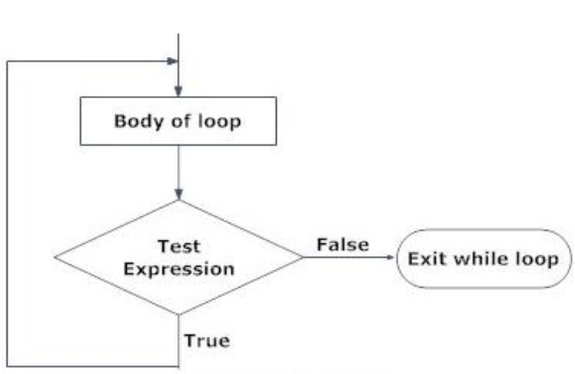


Fig: operation of while loop

The while is an entry controlled loop statement. The test condition is evaluated first and if the condition is found to be true then the body of loop is executed. After execution of the body the test condition is once again evaluated. This process is repeated until the test condition finally becomes false and the control is transferred to statement-X just out of the loop.

b). do....while( ):- The do….while statement is an exit control loop structure. The general form of do…while loop control structure is as follows:-

```
do
   {
    body of loop;
   }while(test condition);
statement-X;
```



In do…..while control proceeds to execute the body of loop while reaching on do statement. At the end of the loop, the test condition in while statement is evaluated. If the condition is true the program continues to execute the body of loop once again. This process continues as long as the condition is true. When the condition becomes false the loop is terminated and the control goes out of the loop structure to execute the following statement-X. In do…while body of loop gets executed first time unconditionally. Hence execution of the body of loop in do….while may be guaranteed for once.

c). The for( ) statement: The for loop is an entry controlled loop which has more concise loop control structure. The general form of for loop is as follows:-

```
for(initialization ; conditional check ; incr./decr,)
      {
       body of loop;
      }
statement-X;
```

There are three parts in for statement separated by two semi colons(;). First part is called initialization part where the loop control variables are initialized. The second part is test condition where the value of loop control variable is tested evaluating the test condition. The loop continues if the test condition is found to be true otherwise the loop is terminated and the control jumps out of the loop and executes the statement-X, the next statement following the loop.

When the body of loop is executed, the control is transferred back to the third part of for statement which is known as increment/decrement part. In this part the value of control variable(s) is/are incremented or decremented and the new value of the control variable is again tested to see whether it satisfies the loop condition. If the condition is satisfied, the body of loop is again executed. This process continues till the value of control variable fails to satisfy the test condition.

Additional features of for loop:- A for loop may have more than one control variable. The initialization part of for statement can initialize more than one control variable using comma operator. Similarly the increment or decrement part

may also contain more than one expressions separated by comma operator. The test condition part may contain more than one condition in the form of relational expression combined together using logical operator.

```
for(i=0,j=100;i<50||j>50;i++,j--)
```

In for statement all the three parts are not compulsory. First, third may remain absent but to make the ***for*** statement meaningful the initialization and increment are done by other means. In for statement condition part may also remain absent. If the test condition is not present in for structure, it is considered to be true. In such case, for statement is terminated using break statement anywhere in the body of loop.

3) Control statements without decisions making or jump statements:-

a. break: Break statement is used within loop control structure to break the repetition of the loop. It is also used at the end of each case block of switch case statement to transfer the control outside the switch case statement. The general format of break statement is as follows:-

break;

When loop control structures are nested the break statement stops the execution of nearest loop or the loop directly containing the break statement.

b. continue: C supports a statement called continue which causes the loop to be continued with the next iteration after skipping statements between continue statement and the end of the loop. The general format of continue statement is as follows:-

continue;

c. goto: C supports the goto statement to branch the control from one point to another unconditionally although it is not essential to use the goto statement in a highly structured language there may be some situation in C program when the use of goto might be desirable.

The general form of goto statement is:-

goto LABEL;

LABEL:- A LABEL is any valid variable name followed by a colon. The LABEL is placed immediately before the statement where the control is to be transferred. The LABEL can be placed after the goto statement or before the goto statement. When LABEL is placed after the goto statement the transfer of control is called forward jump and when the LABEL is placed before the goto statement the transfer of control is called backward jump.

```
void main()
    {
      -------------------------
      -------------------------
      INPUT:
      -------------------------
      -------------------------
      goto INPUT;
      -------------------------
    }
```

(BACKWARD JUMP)

```
void main()
    {
    --------------------
    ----------------------
    goto OUTPUT:
    --------------------
    --------------------
    OUTPUT:
    ----------------------
    ------------------
    }
```

(FORWARD JUMP)

The goto statement must be used within a decision making and branching statement otherwise in case of forward jump it will skip a part of program every time and in case of backward jump it will create infinite loop. The goto statement be use carefully in a program. We should try to avoid using goto as far as possible.

c. return:- The return statement is used to transfer the control from called function to the calling function. It can return a value or may be used to just transfer the control. It is also used to return a value from called function to calling function.

The general format of return is as follows:

return;

***re*turn value;**

return statement is normally used at the end of the function body. It can return only one value of the type of the function. If return statement is used anywhere else in a program it must be within a decision making and branching statement.

```
void main()
    {
    int sum(int, int);
    int x,,y,z;
    z=sum(x,y);
    ---------------;
    ---------------;
    }
int sum(int a, int b)
    {
    return a+b;
    }
```

Nesting of control statements:-

In C it is possible to use one control statement within another control statement. In other word nesting of control statement is possible. One can use decision making and branching statement within a looping statement or looping statement within a branching statement.