UNIT 6
CONTROL STRUCTURE

LH – 6HRS

PRESENTED BY:

ER. SHARAT MAHARJAN

C PROGRAMMING

CONTENTS (LH – 6HRS)

6.1 Introduction,

6.2 Type of Control Structure (Branching: if, if else, if elseif and switch case, Looping: while, do while and for and Jumping: goto, break and continue)

6.3 Nested Control Structure

6.1 Introduction

- In general, the program statements are executed in same order in which they appear in the source program but if we want to alter the flow of normal execution of a program, control statements are used.
- Thus, programming construct that contains control statements to control the flow of execution in a program is called **control structure**.
- Control structures enable us to specify the flow of program control.
- They make it possible to make decisions, to perform tasks repeatedly or to jump from one section of code to another.

6.2 Type of Control Structure

Control statements can be categorized into three sub categories:

1. **Decision Making Statements**

- if statement
- if…else statement
- if…else if statement
- nested if…else statement
- switch statement

2. **Iteration (Looping) Statements**

- for loop
- while loop
- do… while loop

3. **Jumping Statements**

- return
- break
- continue
- goto

1. Decision Making Statements

- The decision making statements test a condition and allow executing some program statements on the basis of result of the test condition (either true or false).
- In decision making statements, if condition is true step/set of steps are executed otherwise another step/set of steps are get executed.

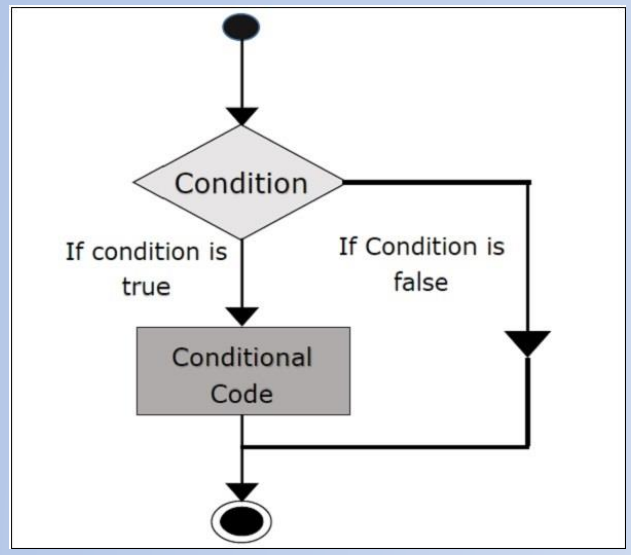
a. if Statement

- The if statement is a two-way decision statement and is used together with an expression, i.e. test condition.
- The if statement evaluates the test expression first and then, if the value of the expression is true, it executes the statement(s) within its block.
- Otherwise, it skips the statements within the block and continues from the first statement outside the if block.

- The general syntax is:

```
if(test_expression)
{
statement-block;
}
statement-x;
```

- The flowchart is:

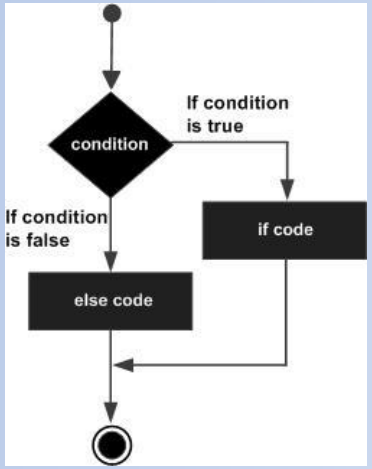


b. if else Statement

- The if…else statement is an extension of the simple if statement. It is used when there are two possible actions- one when a condition is true, and the other when the condition is false. If expression evaluates to true, true block statements is executed. If expression evaluates to false, false block statement is executed.
- The general form is:

```
if(test_expression)
{
true-block statement(s);
}
else
{
false-block statement(s);
}
```

- The flowchart of the if-else statement is as follow:

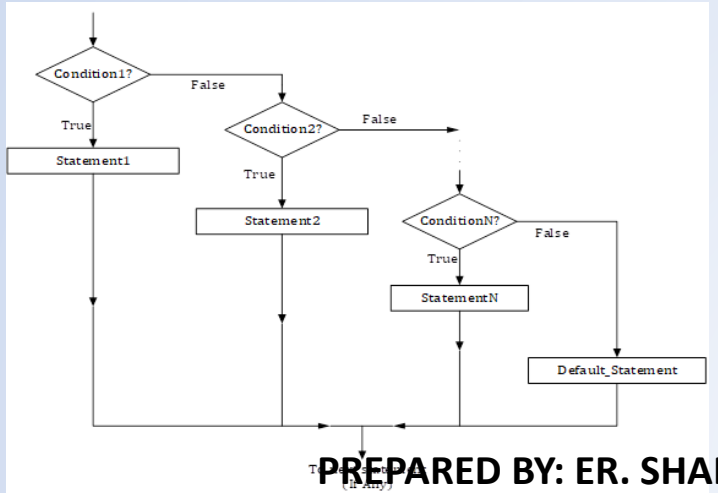


c. if -else-if Statement

- This is a type of nesting in which there is an if…else statement in every part except the last else part. This type of nesting is frequently used in programs and is also known as else if ladder.
- The general syntax for the if-else if ladder is:

```
if (condition1)
stat-A;
else if(condition2)
stat-B;
else if(condition3)
stat-C;
…
else
next statement;
```

- The conditions are evaluated from the top downward. As soon as true condition is found, the statement associated with it is executed and the rest of the ladder is bypassed. If none of the conditions are true, the final else is executed. If the final else is not present, no actions take place if all the other conditions are false.
- The following flow chart shows the logic of if-else if ladder:

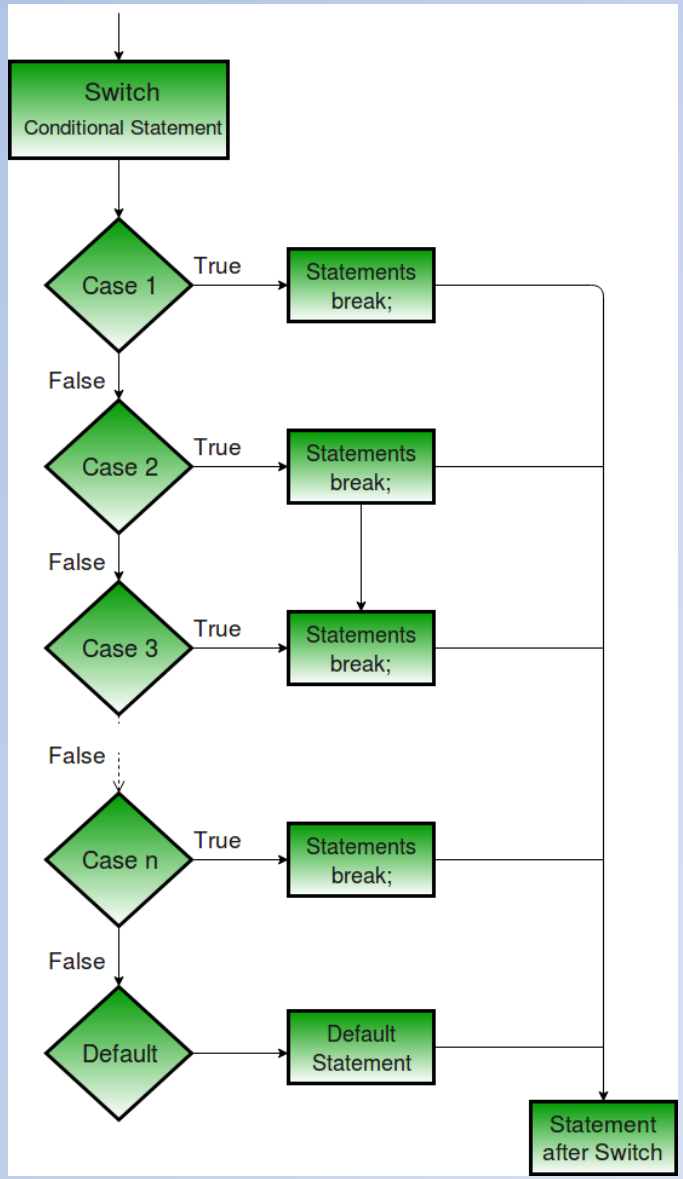


d. switch Statement

- The switch is a multiple branch selection statement that successively tests the value of an expression against a list of integer or character constant. When a match is found, the statements associated with that constant are executed. Useful when there are a number of else alternatives and one of them is to be selected on the basis of some criteria. The constants in switch statement may be either char or int type only.
- The general syntax for the switch statement is as follows:

```
switch(expression)
{
        case constant1:
        statement1(s)
        break;
        case constant2:
        statement2(s)
        break;
        …
        case constantN:
        statement(s)
        break;
        default:
        default_statement(s)
}
```

- The flowchart of switch statement is as follows:



In switch statement the expression must evaluate to an integer type

- The value of expression is tested against the constants present in the case labels.
- When a match is found, the statement sequence, if present, associated with that case is executed until the break statement or the end of the switch statement is reached.
- The statement sequence following default label is executed if no matches are found.
- The default label is optional, and if it is not present, no action takes place if all matches fail.

2. Iteration (Looping/Repetitive) Statements

- Iteration is the process of executing a group of statements more than one time as long as some condition remains true.
- A loop may be defined as a block of statements which are repeatedly executed for a certain number of times or until a particular condition is satisfied.
- There are three types of loop statements in C:

a. The **for** loop

b. The **while** loop

c. The **do while** loop

a. The for loop

- The for loop is useful to execute a statement for a number of times.
- The for statement is most often used in situations where the programmer knows in advance how many times a particular set of statement are to be repeated.
- Thus, this loop is also known as a definite loop or a counted loop.
- The general syntax of the for loop is as follows:

```
for([initialization]; [condition]; [increment/decrement])
{
[statement body;
}
```

✓**initialization:** this is usually an assignment to set a loop counter variable.

✓**test-condition:** determines when loop will terminate.

✓**increment:** defines how the loop control variable will change each time the loop is executed. This can be decrement or empty.

✓**statement body:** can be a single statement, no statement or a block of statements.

- The control flow using the for loop can be expressed with the following flowchart:

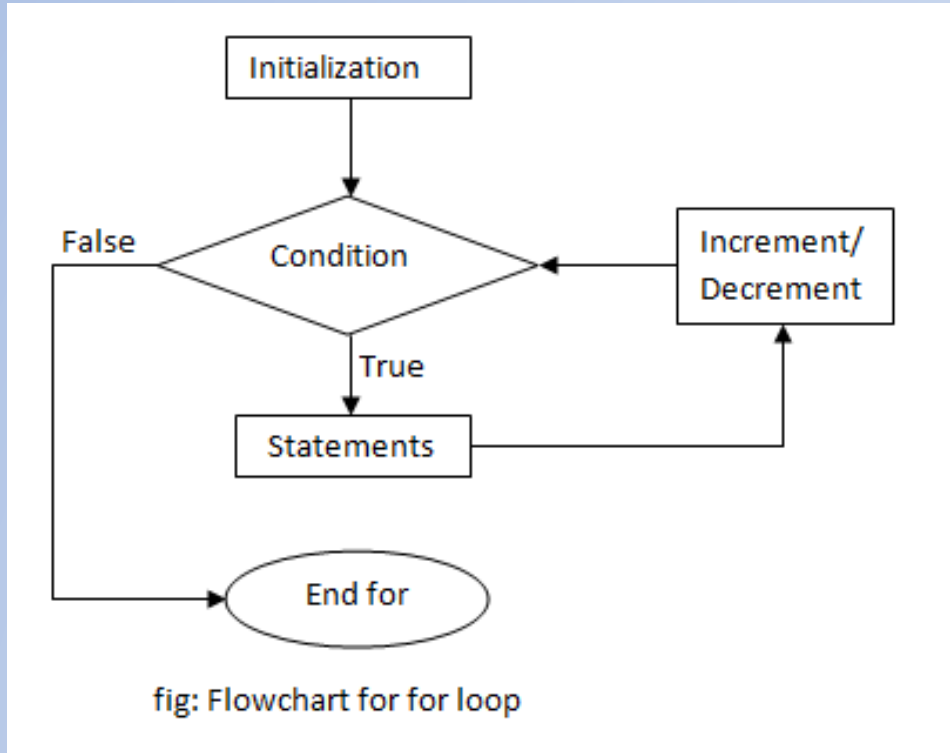


fig: Flowchart for for loop

b. The while loop

- The while statement is typically used in situations where it is not known in advance how many iterations are required.
- A while loop is the most basic type of loop. It will run as long as the condition is non-zero (true).
- The general syntax and flowchart of while statement can be expressed as follows:

```
while(condition)
{
statement body;
}
```

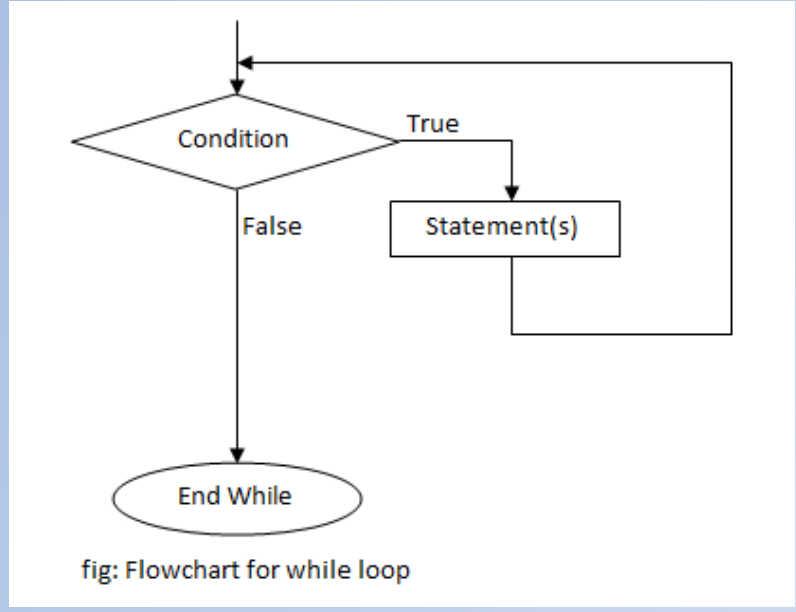


fig: Flowchart for while loop

- First the condition is evaluated, if it is true (non-zero) then the statements in the body of loop are executed.
- After the execution, again the condition is checked and if it is found to be true then again the statements in the body of loop are executed.
- This means that these statements are executed continuously till the condition is true and when it becomes false (zero), the loop terminates and the control comes out of the loop.
- Each execution of the loop body is known as iteration.

c. The do-while loop

- do-while loop executes a body first without checking any condition and then checks a test condition to determine whether the body of loop is to be executed for next time or not.
- The terminating condition in for loop and while loop is always tested before the body of the loop is executed -- so of course the body of the loop may not be executed at all.
- In the do-while the statement body is always executed at least once as the condition is tested at the end of the body of the loop.

The syntax and flowchart of the do-while loop can be expressed as follows:

```
do{
statement body;
}while(condition);
```

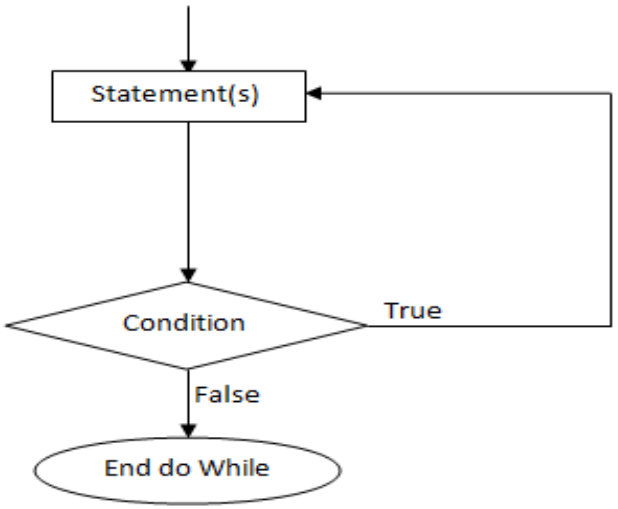


fig: Flowchart for do-while loop

- Here firstly the segments inside the loop body are executed and then the condition is evaluated.
- If the condition is true, then again the loop body is executed and this process continues until the condition becomes false.
- Unlike while loop, here a semicolon is placed after the condition.
- In a 'while' loop, first the condition is evaluated and then the statements are executed whereas in do while loop, first the statements are executed and then the condition is evaluated.
- So, if the condition is false the while loop will not execute at all, whereas the do while loop will always execute at least once.

3. Jumping Statements

- The C language has four statements such as **break**, **continue**, **return** and **goto** used to perform an unconditional branch are called jump statements.
- Of these, we may use **return** and **goto** anywhere in program.
- We can use the **break** and **continue** statements in conjunction with any of the loop statements.
- The C statements which unconditionally branch (jump) are as follows:

a. The break statement

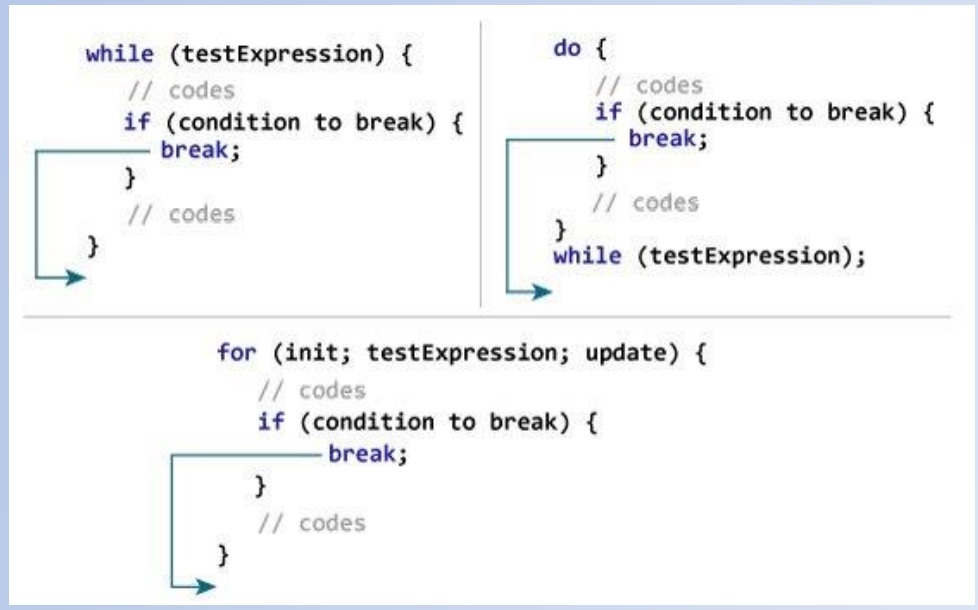
b. The continue statement

c. The return statement

d. The goto statement

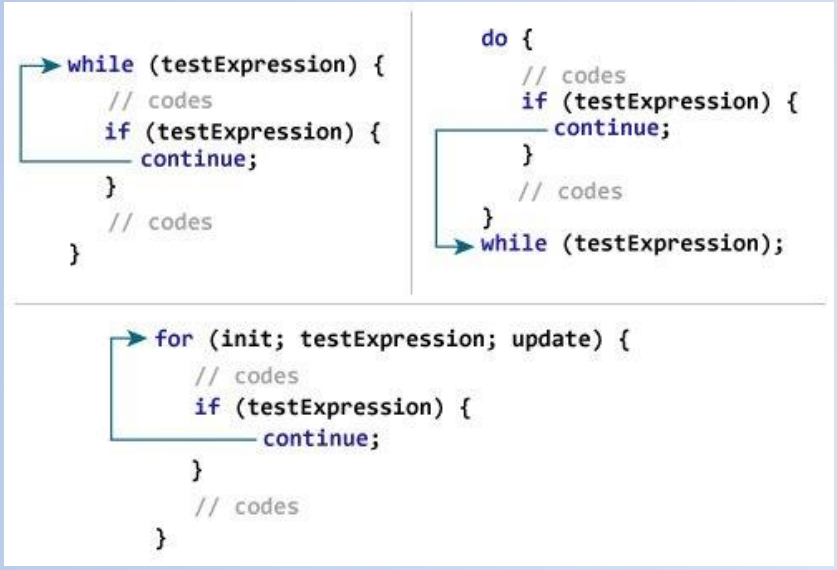
a. The break statement

- It terminates the execution of the loop and the control is transferred to the statement immediately following the loop.
- When a break statement is encountered inside a while, for, do/while or switch statement and the statement is immediately terminated and execution resumes at the next statement following the statement.
- The break statement has two uses. We can use it to terminate a case in the switch statement as well as we can also use it to force immediate termination of a loop, without going through loop termination condition.
- The following code segment shows the uses of the break statement within the loops and its effect.



b. The continue statement

- The continue statement terminates the current iteration of a while, for or do/while statement and resumes execution back at the beginning of the loop body with the next iteration.
- The continue statement is used to bypass the remainder of the current pass through a loop.
- The loop doesn't terminate when a continue statement is encountered. Instead the remaining loop statements are skipped and the computation proceeds directly to the next pass through the loop.
- The following skeleton of the code clearly shows the uses of the continue statement with loops.



c. The return statement

- It is used to return from a function. It is categorized as a jump statement because it causes execution to return back to the point at which the call to the function was made.
- A return may or may not have a value associated with it. If return has a value associated with it, that value becomes the return value of the function.
- The general form of the return statement is

  return expression;

- The expression is present only if the function is declared as returning a value. In this case, the value of expression will become the return value of the function. We can use as many return statements as we like within a function. However, the function will stop executing as soon as it encounters the first return.

```
int findMax(int a, int b){
        if(a>b)
                return a;
                return b;
}
```

d. The goto statement

- It is used to alter the normal sequence of program execution by unconditionally transferring control to some other part of the program.
- The goto statement transfers the control to the labeled statement somewhere in the current function.
- The general syntax of goto statement:

```
goto label;
------
------
label:
statement;
------
-----
```

- Here, label is any valid C identifier and it is followed by a colon. Whenever, the statement goto label is encountered, the control is transferred to the statement that is immediately after the label.
- Generally, the use of goto statement is avoided as it makes program illegible and unreliable.

6.3 Nested Control Structure

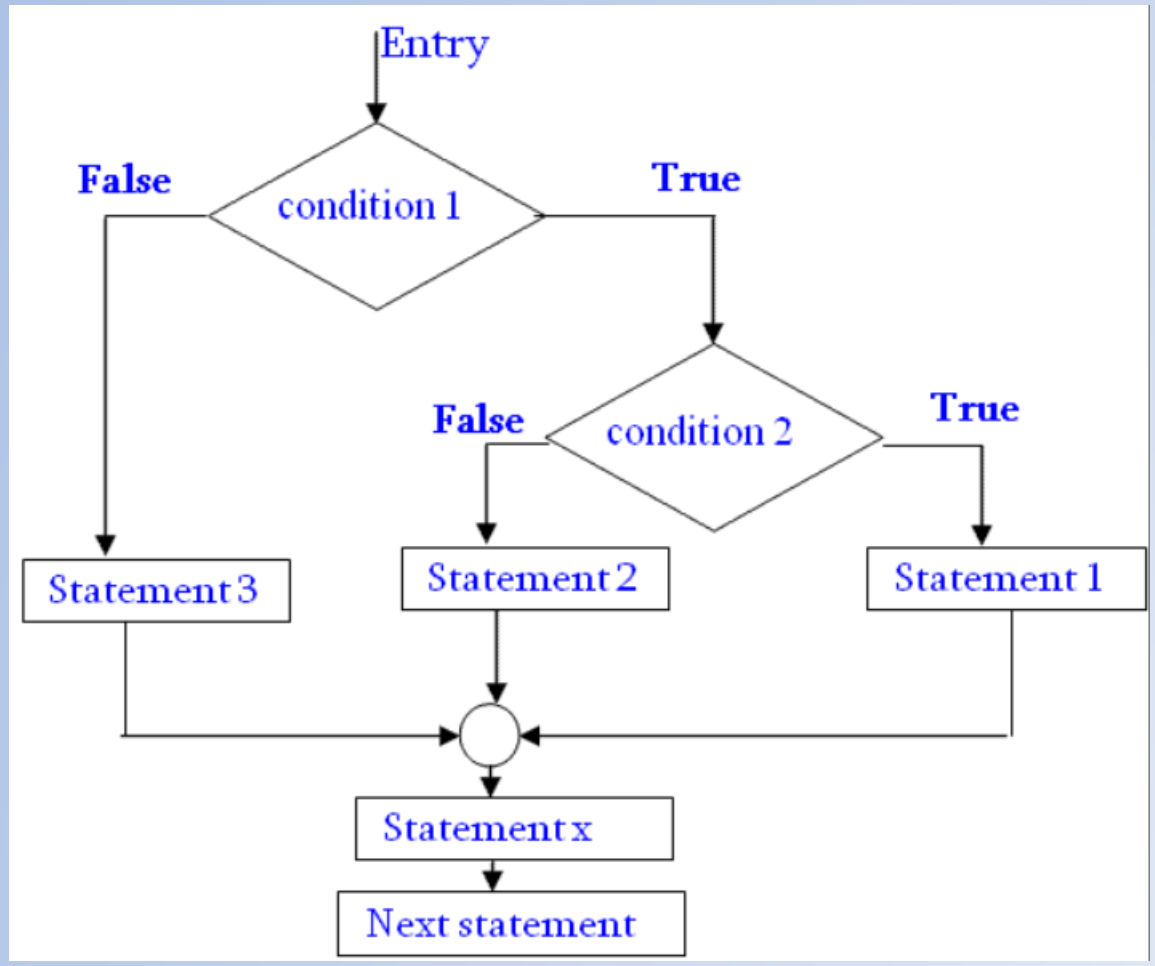
a. Nested if statement

- When series of decisions are involved, we may have to use more than one if else statement in nested.
- An if else statement contains another if else statement is called nested if else statement.
- The general format is:

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

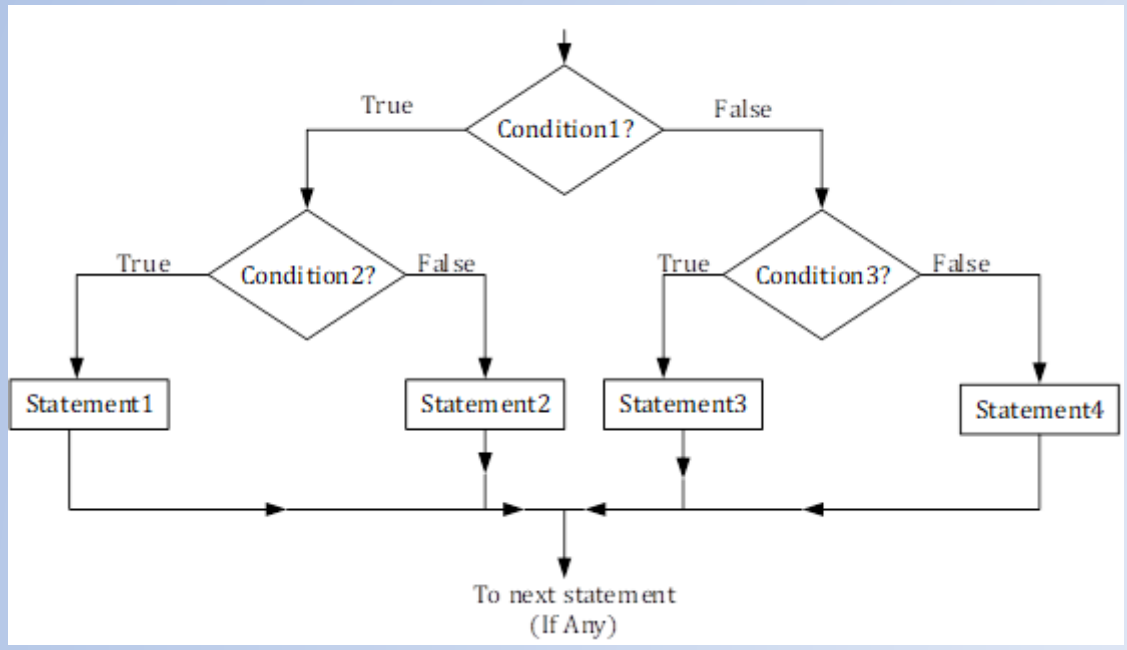
- In above syntax, if the condition 1 is false, the statement 3 will be executed; otherwise it continues to perform the second test. If the condition 2 is true, the statement 1 is executed; otherwise statement 2 will be executed and then the control is transferred to the next statement of the program if any.

- The flowchart of above syntax is as follows:



b. Nested if…else Statement

- Similar to nested if statement, if…else statements shall also be written inside the body of another if-else body called nested if-else statements.
- The if…else statement shall be declared within either anyone of if or else body statement or both.



c. Nesting of Loops

- When a loop is written inside the body of another loop, then it is known as nesting of loops.
- Any type of loop can be nested inside any other type of loop.
- For example, a for loop may be nested inside another for loop or inside a while or do/while loop.
- Similarly, while and do/while loops can be nested.

THANK YOU FOR YOUR ATTENTION