Control Flow Statements

Control flow or flow of control is the order in which instructions, statements and function calls being executed or evaluated when a program is running. The control flow statements are also called as Flow Control Statements.

In C, statements inside your code are generally executed sequentially from top to bottom, in the order that they appear. It is not always the case your program statements to be executed straightforward one after another sequentially, you may require to execute or skip certain set of instructions based on condition, jump to another statements, or execute a set of statements repeatedly.

In C, control flow statements are used to alter, redirect, or to control the flow of program execution based on the application logic.

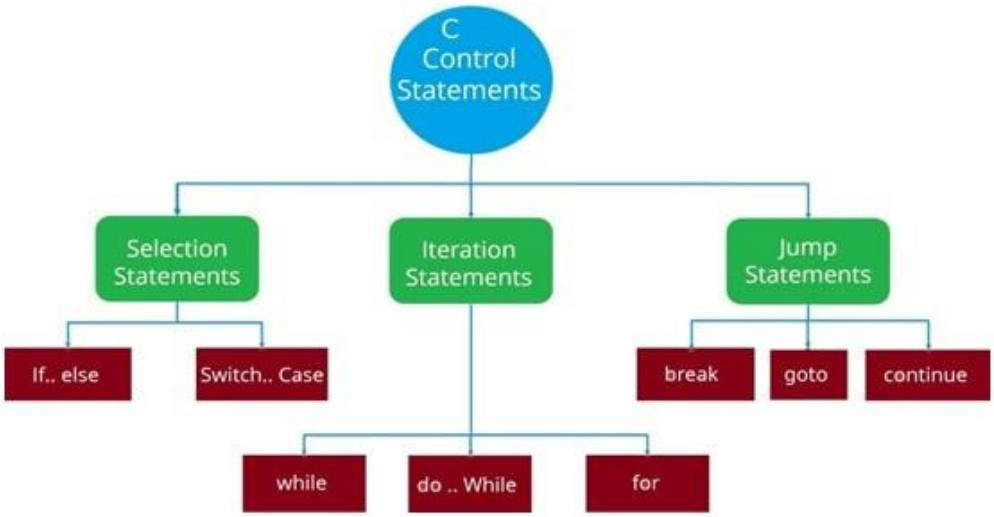
Control Flow Statement Types

In C, Control flow statements are mainly categorized in following types –

Selection statements

Iteration statements

Jump statements

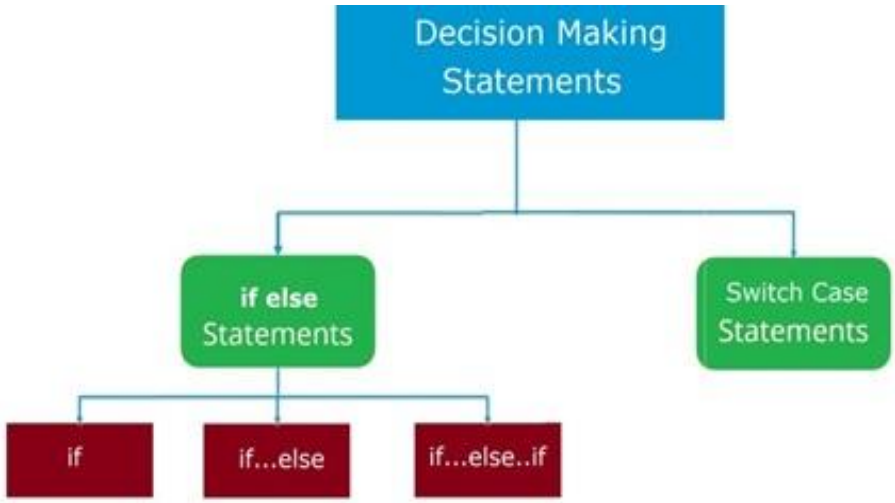


C Selection Statements

In C, Selection statements allow you to control the flow of the program during run time on the basis of the outcome of an expression or state of a variable. Selection statements are also referred to as Decision making statements. Selection statements evaluate single or multiple test expressions which results in “TRUE” or “FALSE”. The outcome of the test expression/condition helps to determine which block of statement(s) to execute if the condition is “TRUE” or “FALSE” otherwise.

In C, we have following selection statements –

- if Statements
- if else Statements
- if else if Statements
- Nested If Statements
- Switch Case Statement



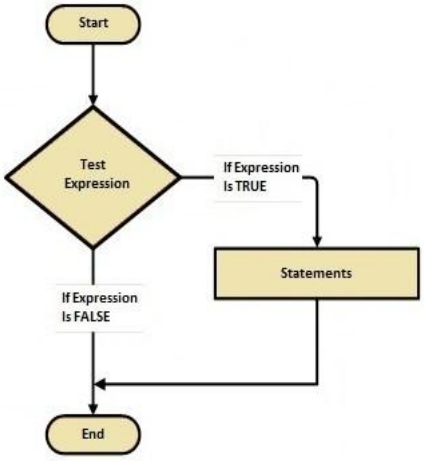
If Statement

If statement allows a block of code to be executed only when a specified condition is true. An if statement evaluates a boolean expression followed by one or more statements. The given boolean expression results in a boolean value that can only be either true or false.

Syntax:-

```
if(condition){
// statements
}
```

If Statement Flow Diagram



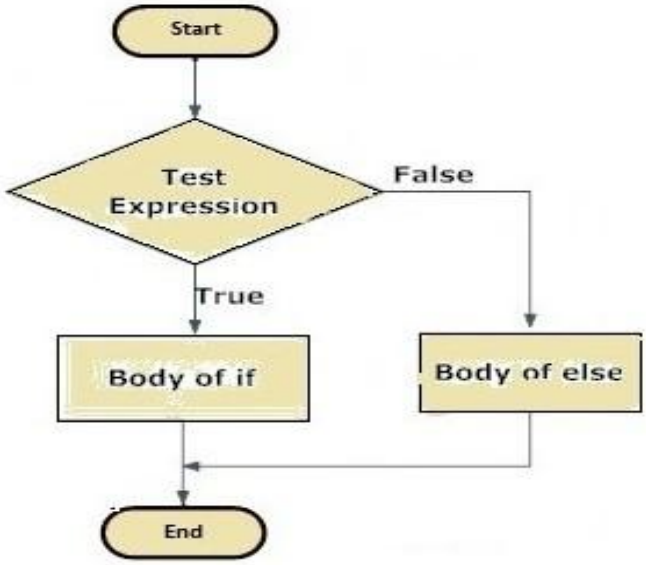
If Else Statement

In C, when we want to execute a block of code when if condition is true and another block of code when if condition is false, In such a case we use if…else statement.

Syntax:-

```
if(condition){
 // statements
} else {
 // statements
}
```

If…else Statement Flow Diagram



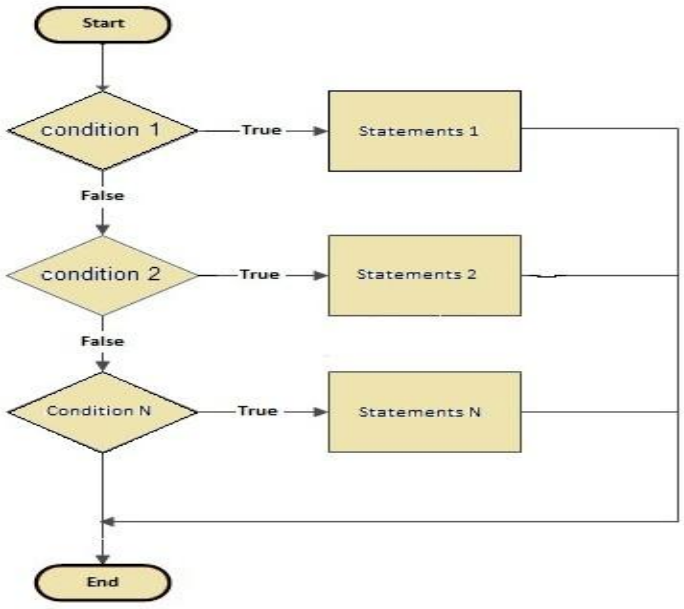
If Else If Statement

When we want to add multiple condition checks in single if else statement then by using if else-if else statement we can easily add multiple conditions. In C, if..else..if statement allows us add alternative set of test conditions in if..else statement using **else-if** and single **else** statements for **if** condition. In such way if..else..if statement is used to select one among several blocks of code to be executed.

Syntax:-

```
if(condition1) {
 // statement(s)
}
else if(condition2){
 // statement(s)
}
else if(conditionN){
 // statement(s)
}
else {
 // statement(s)
}
```

if..else..if Statement Flow Diagram



Nested If Statement

In C, when there is an if statement inside another if statement then it is known as nested if else. Nested if else can also be simplified using Switch Case Statement.

Syntax:-

```
if(condition1) {
 if(condition2){
 // statements
 } else {
 // statements
 }
}
else {
 if(condition3){
 // statements
 } else {
 // statements
 }
}
```

Switch Case Statement

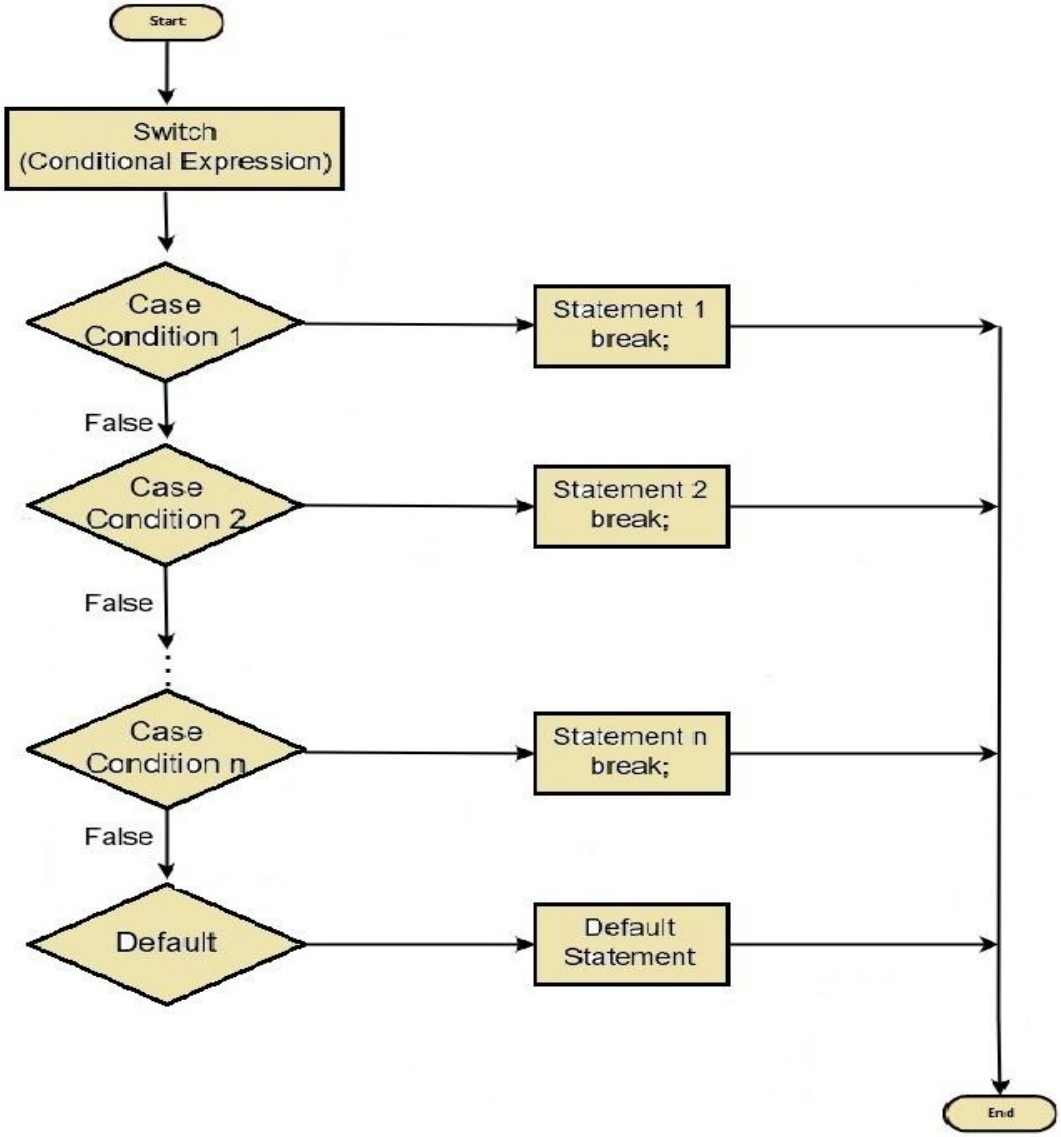
In C, switch case statement is simplified form of the Nested if else statement , it helps to avoid long chain of if..else if..else statements. A switch case statement evaluates an expression against multiple cases in order to identify the block of code to be executed.

Syntax:-

```
switch(expression){
case value1:
 // statements
      break;
case value2:
 // statements
      break;
```

```
default:
 // statements
       break;
}
```

Switch Case Flow Diagram



Iteration Statements (Loops)

In C, Iteration statements are used to execute the block of code repeatedly for a specified number of times or until it meets a specified condition. Iteration statements are commonly known as loops or looping statements.
In C, we have following iteration statements available-

for loop
while loop
do while loop

for loop

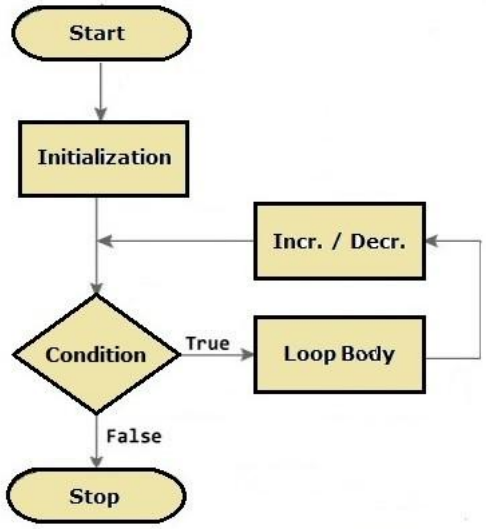
The for loop is used when we want to execute block of code known times. It takes a variable as iterator and assign it with an initial value, and iterate through the loop body as long as the test condition is true.

Syntax :-

```
for(Initialization; Condition; inc/dec)
{
 // loop body
}
```

initialization :- In this section is used to declare and initialize the loop iterator. This is the first statement to be executed and executed only once at the beginning.
condition :- Next, the **test condition** is evaluated before every iteration, and if it returns a true value then the loop body is executed for current iteration. If it returns a false value then loop is terminated and control jumps to the next statement just after the for loop.
incr/decr :- Once, the loop body is executed for current iteration then the **incr/decr** statement is executed to update the iterator value and if the test condition is evaluated again. If it returns a true value then the loop body is executed another time. If test condition returns a false value then loop is terminated and control jumps to the next statement just after the for loop.

For Loop Flow Diagram



Infinite for loop

When a loop executes repeatedly and never stops, it said to be an infinite for loop. In C, if we leave the “initialization”, “increment” and “condition” statement blank, then the for loop will becomes an infinite loop.

Syntax:-

```
// infinite for loop
for ( ; ; ) {
  // statement(s)
}
```

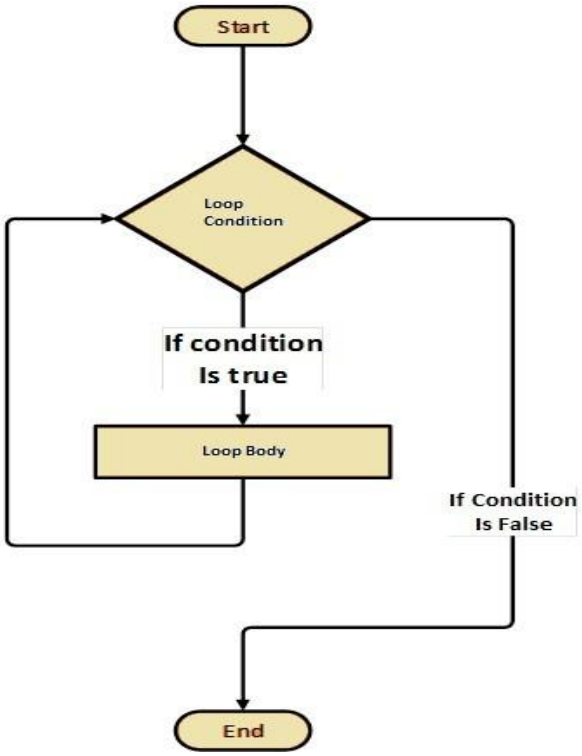
while loop

The while loop will execute a block of statement as long as a test expression is true .

Syntax:-

```
while(condition)
{
   // loop body
}
```

While Loop Flow Diagram



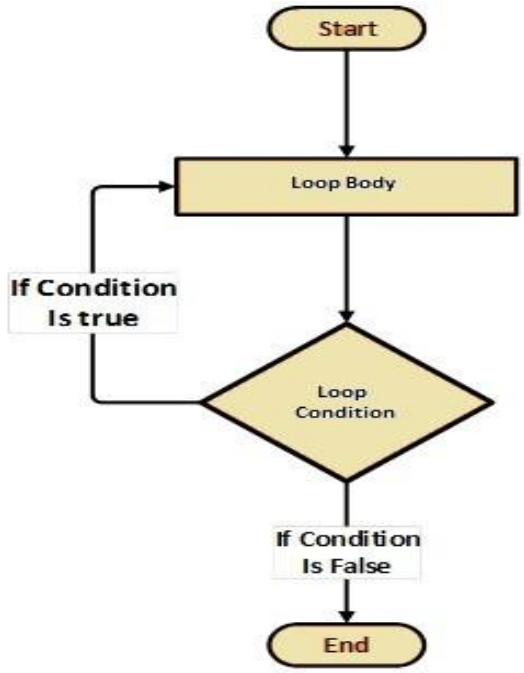
do while loop

The do…while statement executes loop statements and then test the condition for next iteration and executes next only if condition is true.

Syntax:-

```
do{
   // loop body
} while(condition);
```

do while Loop Flow Diagram



Jump Statements

Jump statements are used to alter or transfer the control to other section or statements in your program from the current section.
In C, we have following types of jump statements –

Break Statement
Continue Statement
goto Statement

Continue Statement

The Continue statement is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.

jump-statement;
continue;

Continue Statement Example

```
    for(int i=1;i<=10;i++){
        if(i==5){
            continue;
        }
}
```

Break Statement

The break is used to break loop or switch statement. It breaks the current flow of the program at the given condition. In case of inner loop, it breaks only inner loop.

jump-statement ;
break;

Break Statement Example

```
    for (int i = 1; i <= 10; i++)
    {
        if (i == 5)
      {
        break;
      }
```

Goto Statement

The goto statement is also known as jump statement. It is used to transfer control to the other part of the program. It unconditionally jumps to the specified label.

It can be used to transfer control from deeply nested loop or switch case label.

Goto Statement Example

```
void main()
{
int age;
noteligible:        // goto Label
       printf("You are not eligible to vote!\n");
     printf("Enter your age:\n");
     scanf(“%d”,&age);
     if (age < 18){
           goto noteligible;  // Call goto Lable
     }
   else
     {
           printf("You are eligible to vote!");
     }
}
```

==========