**CONTROL/DECISION MAKING STATEMENT**

**Conditional Statements in C** programming are used to make decisions based on the conditions. **Conditional statements** execute sequentially when there is no **condition** around the **statements**. ... It is also called as branching as a program decides which **statement** to execute based on the result of the evaluated **condition**.

In 'C' programming conditional statements are possible with the help of the following two constructs:

**1. If statement**

**2. If-else statement**

It is also called as branching as a program decides which statement to execute based on the result of the evaluated condition

**If statement**

It is one of the powerful conditional statement. If statement is responsible for modifying the flow of execution of a program. If statement is always used with a condition. The condition is evaluated first before executing any statement inside the body of If. The syntax for if statement is as follows:

How to write a TEST CASE Software Testing Tutorial

if (condition)

instruction;

The condition evaluates to either true or false. True is always a non-zero value, and false is a value that contains zero. Instructions can be a single instruction or a code block enclosed by curly braces { }.

Following program illustrates the use of if construct in 'C' programming:

#include<stdio.h>

int main()

{

int num1=1;

int num2=2;

if(num1<num2) //test-condition

{

printf("num1 is smaller than num2");

}

return 0;

}

Output:

num1 is smaller than num2

**DEMO OF IF STATEMENT**

**Program1**

#include<stdio.h>

Int main()

{

Int number;

Printf(“enter a number less than 10=”);

Scanf(“%d”,&number);

If(number<10)

Printf(“this number is less than 10 \n”);

Return 0;

}

**Output**

Enter a number less than 10 =3

This number is less than 10

**Program2**

#include<stdio.h>

main()

{

int number;

printf("enter an integer \t");

scanf("%d",&number);

if(number>0)

{

printf("we have entered a positive number");

}

printf("\n the number you have enteredis %d",number);

}

**Output 1:**

Enter an integer 6

We have entered a positive integer

The number you have entered is 6

**Output 2:**

Enter an integer -9

The number you have entered is -9

**IF-ELSE STATEMENT**

The if-else is statement is an extended version of If. The general form of if-else is as follows:

if (test-expression)

{

True block of statements

}

Else

{

False block of statements

}

Statements;

n this type of a construct, if the value of test-expression is true, then the true block of statements will be executed. If the value of test-expression is false, then the false block of statements will be executed. In any case, after the execution, the control will be automatically transferred to the statements appearing outside the block of If.

Following programs illustrate the use of the if-else construct:

We will initialize a variable with some value and write a program to determine if the value is less than ten or greater than ten.

#include<stdio.h>

int main()

{

int num=19;

if(num<10)

{

printf("The value is less than 10");

}

else

{

printf("The value is greater than 10");

}

return 0;

}

**Output**:The value is greater than 10

**Nested If-else Statements**

When a series of decision is required, nested if-else is used. Nesting means using one if-else construct within another one.

Let's write a program to illustrate the use of nested if-else.

#include<stdio.h>

int main()

{

int num=1;

if(num<10)

{

if(num==1)

{

printf("The value is:%d\n",num);

}

else

{

printf("The value is greater than 1");

}

}

else

{

printf("The value is greater than 10");

}

return 0;

}

Output:

The value is:1

The above program checks if a number is less or greater than 10 and prints the result using nested if-else construct.

**DEMO OF NESTED IF-ELSE**

//write a c program to find the greatest among three numbers

#include<stdio.h>

int main()

{

int a,b,c;

printf("enter three values\t");

scanf("%d %d %d",&a,&b,&c);

printf("largest value is\t");

if(a>b)

{

if(a>c)

printf("%d\n",a);

else

printf("%d\n",c);

}

else

{

if(c>b)

printf("%d\n",c);

else

printf("%d\n",b);

}

}

**output**

## enter three values 34 67 21

## largest value is 67

## Nested Else-if statements

Nested else-if is used when multipath decisions are required.

The general syntax of how else-if ladders are constructed in 'C' programming is as follows:

if (test - expression 1) {

statement1;

} else if (test - expression 2) {

Statement2;

} else if (test - expression 3) {

Statement3;

} else if (test - expression n) {

Statement n;

} else {

default;

}

Statement x;

This type of structure is known as the else-if ladder. This chain generally looks like a ladder hence it is also called as an else-if ladder. The test-expressions are evaluated from top to bottom. Whenever a true test-expression is found, statement associated with it is executed. When all the n test-expressions becomes false, then the default else statement is executed.

Let us see the actual working with the help of a program.

#include<stdio.h>

int main()

{

int marks=83;

if(marks>75){

printf("First class");

}

else if(marks>65){

printf("Second class");

}

else if(marks>55){

printf("Third class");

}

else{

printf("Fourth class");

}

return 0;

}

Output:

First class

**Decision using Switch statement**

The switch statement test the value of a given variable(or expression)against a list of case values and when a match is found,a block of statement associated with that case is executed.

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each **switch case**.

Syntax

The syntax for a **switch** statement in C programming language is as follows −

switch(expression) {

case constant-expression :

statement(s);

break; /\* optional \*/

case constant-expression :

statement(s);

break; /\* optional \*/

/\* you can have any number of case statements \*/

default : /\* Optional \*/

statement(s);

}

**Example 1**

#include<stdio.h>

Int main()

{

int i=2;

switch(i)

{

case 1:

printf(“i am in case1\n”);

break;

case 2:

printf(“i am in case 2 \n”);

break;

case 3:

printf(“i am in case 3 \n”);

Break;

default:

printf(“i am in default\n”);

}

return 0;

}

**Output**

I am in case 2

**EXAMPLE 2**

#include <stdio.h>

int main ()

{

/\* local variable definition \*/

char grade = 'C';

switch(grade)

{

case 'A' :

printf("Excellent!\n" );

break;

case 'B' :

case 'C' :

printf("Well done\n" );

break;

case 'D' :

printf("You passed\n" );

break;

case 'F' :

printf("Better try again\n" );

break;

default :

printf("Invalid grade\n" );

}

printf("Your grade is %c\n", grade );

return 0;

}

**Output**

well done

Your grade is C

**Summary**

* **Decision making or branching statements are used to select one path based on the result of the evaluated expression.**
* **It is also called as control statements because it controls the flow of execution of a program.**
* **'C' provides if, if-else constructs for decision-making statements.**
* **We can also nest if-else within one another when multiple paths have to be tested.**
* **The else-if ladder is used when we have to check various ways based upon the result of the expression**.
* **A switch statement allows a variable to be tested for equality against a list of values**.