**C++ 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.

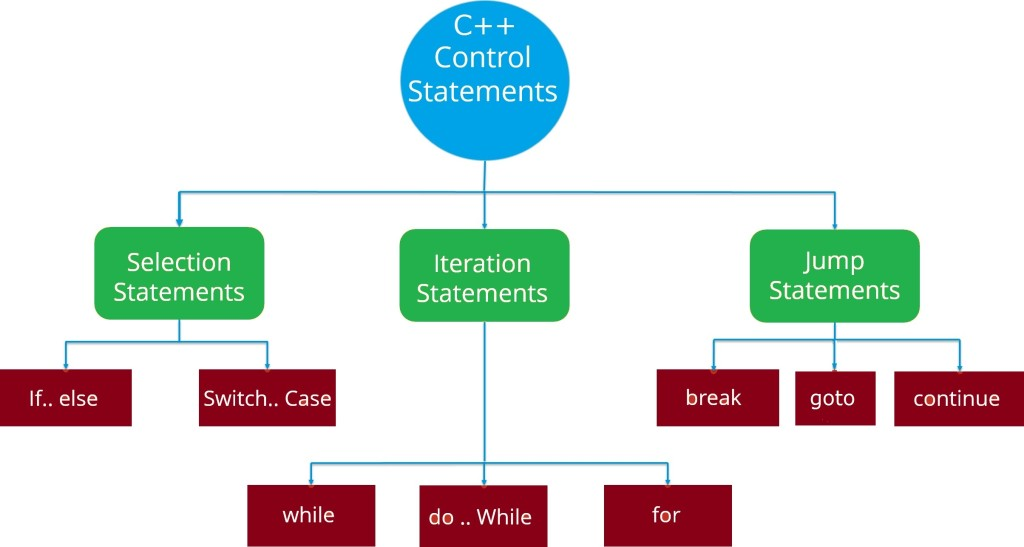
C++ Control Flow Statement Types

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 evaluates 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 executed if the condition is “TRUE” or “FALSE” otherwise.

In C++, we have following decision making 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

}

Here, Condition is a Boolean expression that results in either True or False, if it results in True then statements inside if body are executed, if it results in False then execution is skipped from if body.

**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

}

Here, Condition is a Boolean expression that results in either True or False, if it results in True then statements inside if body are executed, if it results in False then statements inside else body are executed.

#include<iostream>

using namespace std;

main()

{

int choice;

double temp,conv\_temp;

cout<<"Temperature Conversion Menu:"<<"\n";

cout<<"1.Fahrenheit to Celsius"<<"\n";

cout<<"2.Celsius to Fahrenheit"<<"\n";

cout<<"Enter choice(1-2):";

cin>>choice;

if(choice==1)

{

cout<<"\n"<<"Enter temperature in Fahrenheit:";

cin>>temp;

conv\_temp=(temp-32)/1.8;

cout<<"The temperature in Celsius is"<<conv\_temp<<"\n";

}

else

{

cout<<"\n"<<"Enter temperature in Celsius:";

cin>>temp;

conv\_temp=(1.8\*temp)+32;

cout<<"The temperature in Fahrenheit is"<<conv\_temp<<"\n";

}

return 0;

}

**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)

}

Example:

#include<iostream>

using namespace std;

main()

{

char ch;

cout<<"\n Enter a character";

cin>>ch;

if((ch>=48)&&(ch<=57))

cout<<"\n"<<"You entered a digit";

else if((ch>=65)&&(ch<=90))

cout<<"\n"<<"You entered an uppercase letter";

else if((ch>=97)&&(ch<=122))

cout<<"\n"<<"You entered a lowercase letter";

else cout<<"\n"<<"You entered a special character";

return 0;

}

**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 C++ Switch Case Statement.

Syntax:-

if(condition1)

{

if(condition2)

{

// statements

} else

{

// statements

}

}

else {

if(condition3){

// statements

} else {

// statements

}

}

Example:

#include<iostream>

using namespace std;

main()

{

char ch;

float a,b,result;

cout<<"Enter two numbers:";

cin>>a>>b;

cout<<"\n"<<"Enter the operator(+,-,\*,/):";

cin>>ch;

if(ch=='+')

result=a+b;

else

if(ch=='-')

result=a-b;

else

if(ch=='\*')

result=a\*b;

else

if(ch=='/')

result=a/b;

else

cout<<"Wrong operator\n";

cout<<"\n"<<"The calculated result is:"<<result<<"\n";

return 0;

}

**Switch Case Statement**

In C++, switch case statement is simplified form of the C++ 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;

}

Example:

#include<iostream>

using namespace std;

main()

{

int dow;

cout<<"Enter number for day of the week(1-7):";

cin>>dow;

switch(dow)

{

case 1:cout<<"\n"<<"Sunday";

break;

case 2:cout<<"\n"<<"Monday";

break;

case 3:cout<<"\n"<<"Tuesday";

break;

case 4:cout<<"\n"<<"Wednesday";

break;

case 5:cout<<"\n"<<"Thursday";

break;

case 6:cout<<"\n"<<"Friday";

break;

case 7:cout<<"\n"<<"Saturday";

break;

default:cout<<"\n"<<"You have entered wrong number of day";

break;

}

return 0;

}