Conditional Statements

* In C++, conditional statements is a type of control structure that allows the program to make decisions based on certain conditionas.

* The most common types of conditional statement in C++ are -

i) if statement -

The if statement is used to execute a block of code if a condition is true.

Syntax :

```
if (condition)
{
        // code to execute
              if condition is true
}
```

Eg:-
```
#include <iostream>
using namespace std;
int main()
{
    int x = 10;
    if (x > 5)
    {
        cout<< "x is greater than 5"<< endl;
    }
    return 0;
}
```

ii) if... else statement -

<u>Syntax</u> :

```
if (condition)
{
    // code to execute if condition is true
}
else
{
    // code to execute if condition is false
}
```

Q. To check a number is even or odd -

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout << "Enter a number";
     cin >> n;
    if ( n % 2 = = 0)
        cout << " Even number";
    else
        cout << " Odd number";
}
```

iii) else if ladder (if ... else if ... else) -

Syntax :

```
if (condition)
{
    // code1
}
else if (condition)
{
    // code2
}
else if (condition)
{
    // code 3
}
else
{
    // code 4
}
```

Q. Program to check given no. is +ve, -ve or zero -

```
#include <iostream>
using namespace std;
int main()
{
    int a = 9;
    if (a< 0)
        cout<< " Negative no.";
    else if (a> 0)
        cout << " Positive no.";
    else
        cout<< "zero";
}
```

iv) Nesting of if... else -

When there are another if-else statement in if-block, then it is called nesting of if-else statement.

```
if(condition)
{
    if (condition)
        code1;
    else
        code 2;
}
else
{
    if (condition)
        code 3;
    else
        code 4;
}
```

Q. Greater value in two numbers -

```
#include <iostream>
using namespace std;
int main()
{
    int a = 10, b = 20;
    if (a<b)
        cout << "b is greater";
    if (a>b)
        cout << "a is greater";
    if (a==b)
        cout << "Both are equal";
}
```

* Switch statement -

Switch statement allows us to execute one statement from many statement based on the value of an expression.

```
switch(expression)
{
    case value1 :
      // code to execute if expression equals value1
       break;
    Case value 2 :
       // code to execute if expression equals value2
       break;
      - - - - - - - -
      - - - - - - - -
      - -   -   - - -
     case value n :
        // code to execute if expression equal value n
        break ;
     default:
        // code to execute if expression does not
                         match any of the cases.
        break ;
}
```

Q. Program to check, of alphabet is consonent or vowel.

```
#include<iostream>
using namespace std;
int main()
 {
     char ch;
     cout<< "Enter any alphabet"<<endl;
      cin>> ch;
      switch (ch)
       {
           case 'a':
              cout<<"vowel";
              break;
           case 'e':
              cout<<"vowel";
              break;
            case 'i':
              cout<<"vowel";
              break;
            case 'o':
              cout<<"vowel";
              break;
            case 'u':
              cout<< "vowel;
              break;
            default:
              cout<<"Consonent";
       }
 }
```

* goto statement -

'goto' is an unconditional control transfer statement, which is used to transfer control from one point to another in the program.

```
#include <iostream>
using namespace std;

int main()                              o/p -
{     int i = 0;                             0
 start:                                      1
     cout << i << endl;                      2
                                             3
      i++;                                   4
     if (i < 5)
         goto start;
 }
```

Q. Program to input five no. and print sum using goto statement.

```
#include <iostream>
using namespace std;
int main()
 {
     int count = 0, sum = 0, n;
     start :
         cout << " Enter a number" ;
           cin >> n;
         sum = sum+n;  count++;
         if (count < 5)
              goto start;
       cout << " Sum of numbers :" << sum ;
 }
```

* Break statement -

The 'break' statement is used to exit a loop or switch statement. When the 'break' statement is encountered, the program will immediately exit the loop or switch statement and continue executing the next line of code after the loop or switch.

```
#include <iostream>
 using namespace std;
 int main()
  {
     int i;
     for( i=0; i<10; i++)
       {
          if (i==4)
             break;
          cout << i << endl
       }
  }
```

```
o/p -  0
       1
       2
       3
```

* Continue Statement -

The 'continue' statement is used to skip the remaining code in a loop iteration and start the next iteration.

When the 'continue' statement is encountered, the program will immediatly exit the current iteration of the loop and start the next iteration.

```
#include <iostream>
 using namespace std;
 int main()
 {
      for( int i=0; i<10; i++)
      {
           if (i = = 5)
                continue;

           cout << i << endl;
      }
 }
```

o/p -

```
0
1
2
3
4
6
7
8
9
```

* Break v/s Return v/s exit(0) -

→ Break statement will stop the loop or switch case.
→ Return statement will stop the function.
→ exit(0) will stop the program.

* Program to demonstrate short circuit -

```
#include <iostream>
using namespace std;
int main()
{
    int a = 10, b = 5, i = 5;
    if (a>b && ++i <= b)
    {

    }
    cout << i << endl;    ⟶ 6

    if(a<b || ++i <= b)
    {

    }
    cout << i << endl;    ⟶ 7
}
```

Note:
* A short circuit in c++ occurs when a boolean expression is evaluated and the result can be determined without evaluating the entire expression.

Loops in C++

* It is a block of statement that performs set of instructions.

  " The looping is a process of repeating a single statement or a group of statements untill some condition for termination of loop is satisfied".

* Types of loops -

i) while loop -

When we want to do something a fixed no. of time but not known about the no. of iteration in a program then while loop is used

In this loop, first condition is checked, if it is true, body of the loop is executed otherwise control will be come out of the loop.

Syntax :

```
while (condition)
{
    // code to be executed repeatedly
}
```

Q Program to print, table of one using while-loop -

```
#include <iostream>
using namespace std;

int main()
{
    int i=1;
    while (i<=10)
     {
        cout<< i << endl
         i++;
     }
}
```

ii) do-while loop -

The do-while loop is similar to the 'while' loop, but the code inside the loop will be executed at least once before the condition is checked.

Syntax:

```
do
 {
    // code to be executed repeatedly

 } while (condition)
```

Q Table of one, using do-while loop-

```
# include <iostream>
using namespace std;
int main()
{
    int i=1;
    do
    {
        cout<< i<< endl;
        i++;
    }
    while(i<=10);
}
```

iii) for loop -

This loop is generally used when the no. of iteration are known in advance.

Syntax:

```
for(initialization ; condition; incre/decre)
{
    // code to be executed
}
```

Q Table of one using for loop -

```
#include <iostream>
using namespace std;

int main()
 {
   for( int i=1;  i<=10 ; i++)
    {
          cout<< i << endl;

    }
 }
```

Q Table of any number -

```
#include <iostream>
using namespace std;

int main()
{
   int n;
   cout << " Enter a number " << endl;
    cin>>n;
    for( int i=1;  i<=10;  i++)
     {
          cout<< i*n << endl;

     }
  }
```

iv) for-each loop - (Range-based for loop).

This loop is work with collection of elements means Array or Vector type container.

```
for(type&var : container)
{
    // code to be executed for each element
}
```

Eg:-

```
int arr[] = {1, 2, 3, 4, 5};
for( int&x : arr)
{
    cout << x << " ";
}
```

} It will print all elements of array 'A'.