

Name : Sufyan Ahmad

ID : 13062

Program : BS ,SE

Paper : Programming fundamental

Date : 26 /6 /2020

Q1

- a) **What is the purpose of *if statement*? Discuss its two different forms with examples.**

Answer :

Statement Purpose:

The various forms of if statements are Fortran's main branching tool. They give Fortran an ability to make decisions in a program. The different forms of if statements that can be used include the simple logical if, the if-then-else structure, and the arithmetic if.

Syntax of if statement:

The statements inside the body of “if” only execute if the given condition returns true. If the condition returns false then the statements inside “if” are skipped.

```
if (condition)
{
    //Block of C statements here
    //These statements will only execute if the condition is true
}
```

Example of if statement

```
#include <stdio.h>

int main()
{
    int x = 20;
    int y = 22;
```

```
if (x<y)
{
    printf("Variable x is less than y");
}
return 0;
}
```

Output:

Variable x is less than y

Explanation: The condition (x<y) specified in the “if” returns true for the value of x and y, so the statement inside the body of if is executed.

Example of multiple if statements

We can use multiple if statements to check more than one conditions.

```
#include <stdio.h>
int main()
{
    int x, y;
    printf("enter the value of x:");
    scanf("%d", &x);
    printf("enter the value of y:");
    scanf("%d", &y);
    if (x>y)
```

```
{  
    printf("x is greater than y\n");  
}  
if (x<y)  
{  
    printf("x is less than y\n");  
}  
if (x==y)  
{  
    printf("x is equal to y\n");  
}  
printf("End of Program");  
return 0;  
}
```

In the above example the output depends on the user input.

Output:

enter the value of x:20

enter the value of y:20

x is equal to y

End of Program

b) Write a C++ program to read two numbers from keyboard and then find the LARGEST number of them.

Answer :

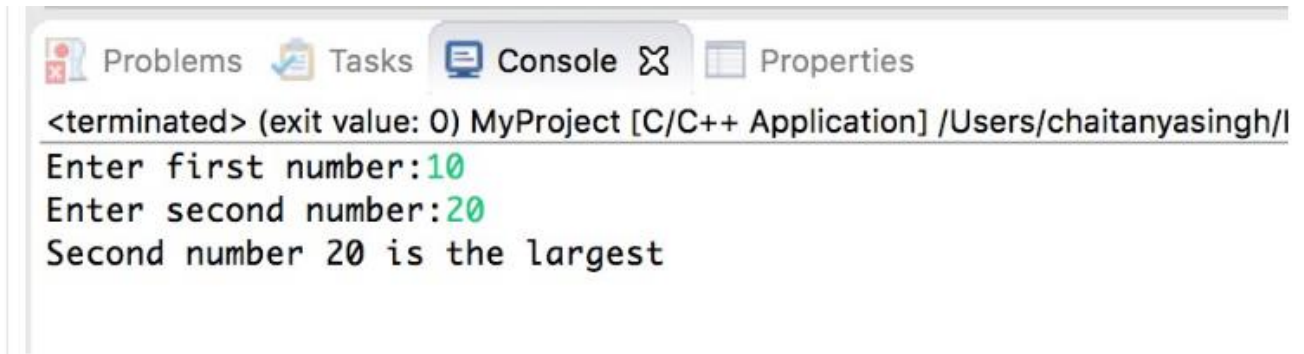
In this program we are using if..else statement to find out the largest of two numbers entered by user.

Example: Program to find the largest of two entered numbers

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

int main()
{
    int num1, num2;
    cout<<"Enter first number:";
    cin>>num1;
    cout<<"Enter second number:";
    cin>>num2;
    if(num1>num2)
    {
        cout<<"First number "<<num1<<" is the largest";
    }
    else
    {
        cout<<"Second number "<<num2<<" is the largest";
    }
    return 0;
}
```

Output:

A screenshot of a C++ application's console window. The window title is "MyProject [C/C++ Application] /Users/chaitanyasingh/". The console shows the following output: "<terminated> (exit value: 0) MyProject [C/C++ Application] /Users/chaitanyasingh/". Below this, the program prompts for input: "Enter first number:10" and "Enter second number:20". The final output is "Second number 20 is the largest". The console window has tabs for "Problems", "Tasks", "Console", and "Properties".

```
<terminated> (exit value: 0) MyProject [C/C++ Application] /Users/chaitanyasingh/  
Enter first number:10  
Enter second number:20  
Second number 20 is the largest
```

Q.2 a) What are the Logical Operators? Explain them

Answer :

The Logical Operators

Logical operators are mainly used to control program flow. Usually, you will find them as part of an if, a while, or some other control statement.

The Logical operators are:

op1 && op2

-- Performs a logical AND of the two operands.

op1 || op2

-- Performs a logical OR of the two operands.

!op1

-- Performs a logical NOT of the operand.

The concept of logical operators is simple. They allow a program to make a decision based on multiple conditions. Each operand is considered a condition that can be evaluated to a true or false value. Then the value of the conditions is used to determine the overall value of the op1 operator

op2 or !op1 grouping. The following examples demonstrate different ways that logical conditions can be used.

The && operator is used to determine whether both operands or conditions are true and.pl.

For example:

```
if ($firstVar == 10 && $secondVar == 9) {
```

```
    print("Error!");
```

```
};
```

If either of the two conditions is false or incorrect, then the print command is bypassed.

The || operator is used to determine whether either of the conditions is true.

b) Write a C++ program to get Temperature in Fahrenheit F and then find the Atmosphere according to the below rules:

- If temperature F is above 40 degree Fahrenheit then display.....Very Hot.
- If temperature F is between 35 & 40 degree Fahrenheit then display.....Tolerable.
- If temperature F is between 30 & 35 degree Fahrenheit then display.....Warm.
- If temperature F is less than 30 degree Fahrenheit then display.....Cool.

Answer :

```
//C++ program for converting degree Celsius into Fahrenheit and vice versa
#include<iostream>
using namespace std;

int main()
{
    float fahr, cel;
    char option;

    cout << "Choose from following option:" << endl;
    cout << "1. Celsius to Fahrenheit." << endl;
    cout << "2. Fahrenheit to Celsius." << endl;
    cin >> option;

    //option for converting celsius into fahrenheit
    if (option == '1')

    {
        cout << "Enter the temperature in Celsius: ";
        cin >> cel;

        fahr = (1.8 * cel) + 32.0; //temperature conversion formula
        cout << "\nTemperature in degree Fahrenheit: " << fahr << " F" << endl;
    }
    //option for converting Fahrenheit into Celsius
    else if (option == '2')
    {
        cout << "Enter the temperature in Fahrenheit: ";
        cin >> fahr;

        cel = (fahr - 32) / 1.8; //temperature conversion formula
        cout << "\nTemperature in degree Celsius: " << cel << " C" << endl;
    }
    else
        cout << "Error Wrong Input." << endl;
```



```
    return 0;  
}
```

Q.3 a) What does Looping mean? Explain different loops in C++.

Answer :

A loop is used for executing a block of statements repeatedly until a particular condition is satisfied. For example, when you are displaying number from 1 to 100 you may want set the value of a variable to 1 and display it 100 times, increasing its value by 1 on each loop iteration.

In C++ we have three types of basic loops: for, while and do-while.

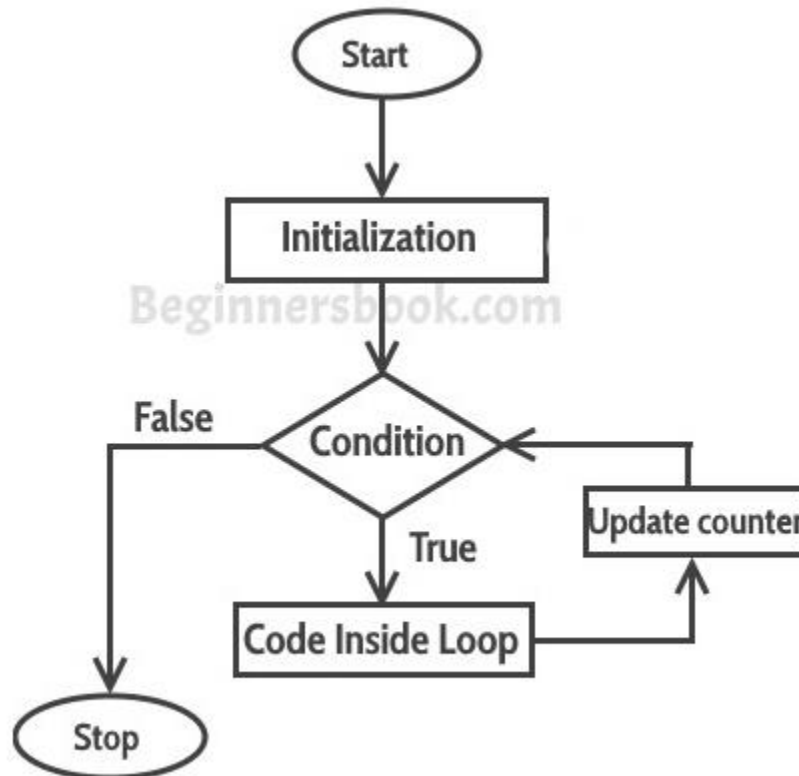
Syntax of for loop

```
for(initialization; condition ; increment/decrement)  
{  
    C++ statement(s);  
}
```

Flow of Execution of the for Loop

As a program executes, the interpreter always keeps track of which statement is about to be executed. We call this the control flow, or the flow of execution of the program.

C++ for loop flow diagram



Example of a Simple For loop in C++

Here in the loop initialization part I have set the value of variable i to 1, condition is $i \leq 6$ and on each loop iteration the value of i increments by 1.

```
#include <iostream>
using namespace std;
int main(){
    for(int i=1; i<=6; i++){
        /* This statement would be executed
        * repeatedly until the condition
        * i<=6 returns false.
        */
        cout<<"Value of variable i is: "<<i<<endl;
    }
    return 0;
}
```

Output:

Value of variable i is: 1

Value of variable i is: 2

Value of variable i is: 3

Value of variable i is: 4

Value of variable i is: 5

Value of variable i is: 6

B . Write a C++ program to read a number from keyboard and then determine whether it is *Even or Odd* number?

Answer :

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

int main()
{
    int n;

    cout << "Enter an integer: ";
    cin >> n;

    if ( n % 2 == 0)
        cout << n << " is even.";
    else
        cout << n << " is odd.";

    return 0;
}
```

Q.4 a) What is the purpose of using break and continue statements?

Break :

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

This example jumps out of the loop when i is equal to 4:

Example

```
for (int i = 0; i < 10; i++) {  
    if (i == 4) {  
        break;  
    }  
    cout << i << "\n";  
}
```

C++ Continue

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 4:

Example

```
for (int i = 0; i < 10; i++) {  
    if (i == 4) {  
        continue;  
    }  
}
```

```
}  
cout << i << "\n";  
}
```

b) Write a C++ program to find the sum of the following numbers:

1+2+3+.....+10

Answer :

Code to add 1+2+3....10

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int i,sum=0;  
    cout << "\n\n Find the first 10 natural numbers:\n";  
    cout << "-----\n";  
    cout << " The natural numbers are: \n";  
    for (i = 1; i <= 10; i++)  
    {  
        cout << i << " ";  
        sum=sum+i;  
    }  
}
```

```
    cout << "\n The sum of first 10 natural numbers: "<<sum << endl;
}
```

Q.5 What is an array? Explain On-Dimensional and Two-Dimensional Arrays with examples.

Answer :

Array :

An array is a group (or collection) of same data types. For example an int array holds the elements of int types while a float array holds the elements of float types.

How to declare Array in C

```
int num[35]; /* An integer array of 35 elements */
```

```
char ch[10]; /* An array of characters for 10 elements */
```

Similarly an array can be of any data type such as double, float, short etc.

One dimensional array :

A one-dimensional array is a structured collection of components (often called array elements) that can be accessed individually by specifying the position of a component with a single index value.

Here is the syntax template of a one-dimensional array declaration

Two dimensional array :

An array of arrays is known as 2D array. The two dimensional (2D) array in C programming is also known as matrix. A matrix can be represented as a table of rows and columns.

Simple Two dimensional(2D) Array Example

This program demonstrates how to store the elements entered by user in a 2d array and how to display the elements of a two dimensional array.

```
#include<stdio.h>
```

```
int main(){
```

```
    /* 2D array declaration*/
```

```
    int disp[2][3];
```

```
    /*Counter variables for the loop*/
```

```
    int i, j;
```

```
    for(i=0; i<2; i++) {
```

```
        for(j=0;j<3;j++) {
```

```
            printf("Enter value for disp[%d][%d]:", i, j);
```

```
            scanf("%d", &disp[i][j]);
```

```
        }
```

```
    }
```

```
    //Displaying array elements
```

```
    printf("Two Dimensional array elements:\n");
```



```
for(i=0; i<2; i++) {  
    for(j=0; j<3; j++) {  
        printf("%d ", disp[i][j]);  
        if(j==2){  
            printf("\n");  
        }  
    }  
}  
return 0;  
}
```

Output:

Enter value for disp[0][0]:1

Enter value for disp[0][1]:2

Enter value for disp[0][2]:3

Enter value for disp[1][0]:4

Enter value for disp[1][1]:5

Enter value for disp[1][2]:6

Two Dimensional array elements:

1 2 3

4 5 6