

Name >> JANZAID KHALIQ

ID >> 15386

Paper >> Programming Fundamentals

Term >> Summer Final

Teacher >> Sir Fazal e Malik

Date >> 29/09/2020

**Q1(a) : What is the purpose of *if statement*? Discuss its two different forms with examples.**

Ans : The IF statement is a powerful programming tool that allows selective branching within a program. Using the IF statement, the programmer can set up code that will only be processed under specified circumstances. The use of the IF statement is closely tied TO the use of logic tools like flowcharting because of the complexity of the problems that the IF statement allows. The flowchart provides an excellent vehicle to think through the logical problem prior to writing the COBOL code.

**Different forms of If Statement with examples :**

**Else If Statement:**

Introduces a Boolean expression that is evaluated if none of the preceding conditions returned TRUE.

**Example:**

int time = 22;  
if (time < 10) {  
  cout << "Good morning.";  
} else if (time < 20) {  
  cout << "Good day.";  
} else {  
  cout << "Good evening.";  
}  
// Outputs "Good evening."

**Else Statement:**

If control reaches this keyword, the sequence of statements that follows it is executed. This occurs when none of the previous conditional tests returned TRUE.

**Example:**

int time = 20;  
if (time < 18) {  
  cout << "Good day.";  
} else {  
  cout << "Good evening.";  
}  
// Outputs "Good evening."

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Q1 (b): Write a C++ program to read two numbers from keyboard and then find the   
LARGEST number of them.**

#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

{

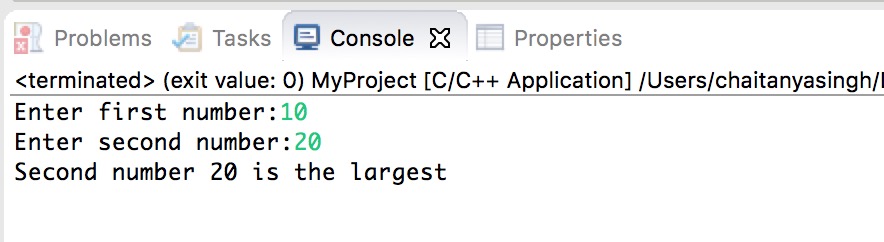
cout<<"Second number "<<num2<<" is the largest";

}

return 0;

}

**Output:**



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q2(1) : What are the Logical Operators? Explain them.**

Ans:

* **&& (logical AND)**
  + Used to combine two conditions
  + true if both conditions are true

if ( gender == 1 && age >= 65 )  
 senior++;

* **|| (logical OR)**
  + true if either of condition is true

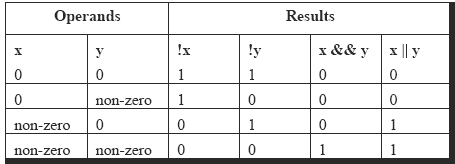
if (semesterAvg >= 90 || finalExam >=90 )  
 cout<<("Student grade is A“);

* **!** (logical **NOT**, logical negation)
  + Returns true when its condition is false, & vice versa

if ( !( grade == 20 ) )  
 cout<<“hello world“);

Alternative:

if ( grade != 20 )  
 cout<<“hello world“);



**Using Logical Operators:**

* A company insures its drivers in the following cases:
  + If the driver is married
  + If the driver is unmarried, male & above 30 years of age
  + If the driver is unmarried, female & above 25 years of age

if ( ms == 'M' )

cout<<"Driver is insured";

else

{

if ( sex == 'M' )

{

if ( age > 30 )

cout<<"Driver is insured" ;

else

cout<<"Driver is not insured" ;

}

else

{

if ( age > 25 )

cout<<"Driver is insured" ) ;

else

cout<<"Driver is not insured" ) ;

}

}

if ( ( ms == 'M') || ( ms == 'U' && sex == 'M' && age > 30 ) ||

( ms == 'U' && sex == 'F' && age > 25 ) )

cout<<"Driver is insured" ) ;

else

cout<<"Driver is not insured" ) ;

**Q2(b): Write a c++ program tp 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 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.**

**Ans:**

#include <iostream>

using namespace std;

main()

{

int temp;

cout<<"enter temperature"<<endl;

cin>>temp;

if(temp>40)

{

cout<<"very hot";

}

else if(temp>35 && temp<=40)

{

cout<<"tolerable";

}

else if(temp>=30 && temp<=35)

{

cout<<"warm";

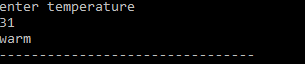
}

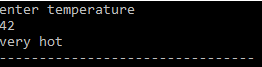
else

cout<<"cool";

}

**Output:**

**cool.PNG **

**tol.PNG **

**Q3(a): What does *Looping* mean? Explain different loops in C++.**

Ans: A **Loop** executes the sequence of statements many times until the stated condition becomes false. A loop consists of two parts, a body of a loop and a control statement. The control statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false. The purpose of the loop is to repeat the same code a number of times.

In C++ there are three types of basic loops

For loop

[While](https://beginnersbook.com/2017/08/cpp-while-loop/) loop

[Do-While](https://beginnersbook.com/2017/08/cpp-do-while-loop/) Loop

**For Loop:**

* The initial value of the for loop is performed only once.
* The condition is a Boolean expression that tests and compares the counter to a fixed value after each iteration, stopping the for loop when false is returned.
* The incrementation/decrementation increases (or decreases) the counter by a set value.

**Example:**

#include<stdio.h>

int main()

{

int number;

for(number=1;number<=8;number++) //for loop to print 1-8 numbers

{

printf("%d\n",number); //to print the number

}

return 0;

}

**Output:**

1

2

3

4

5

6

7

8

**While Loop:**

In while loop, a condition is evaluated before processing a body of the loop. If a condition is true then and only then the body of a loop is executed. After the body of a loop is executed then control again goes back at the beginning, and the condition is checked if it is true, the same process is executed until the condition becomes false. Once the condition becomes false, the control goes out of the loop.

**Example:**

#include<stdio.h>

#include<conio.h>

int main()

{

int num=1; //initializing the variable

while(num<=8) //while loop with condition

{

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

num++; //incrementing operation

}

return 0;

}

**Output:**

1

2

3

4

5

6

7

8

**Do-While loop:**

In the do-while loop, the body of a loop is always executed at least once. After the body is executed, then it checks the condition. If the condition is true, then it will again execute the body of a loop otherwise control is transferred out of the loop.

**Example:**

#include<stdio.h>

#include<conio.h>

int main()

{

int num=1; //initializing the variable

do //do-while loop

{

printf("%d\n",2\*num);

num++; //incrementing operation

}while(num<=10);

return 0;

}

**Output:**

2

4

6

8

10

12

14

16

18

20

**Q3(b): Write a C++ program toread a number from keyboard and then determine whether it is *Even or Odd* number?**

**Ans:**

1. #include<iostream>
2. using namespace std;
4. int main()
5. {
6. int number, remainder;
8. cout << "Enter the number : ";
9. cin >> number;
10. remainder = number % 2;
11. if (remainder == 0)
12. cout << number << " is an even " << endl;
13. else
14. cout << number << " is an odd " << endl;
16. return 0;
17. }

**Output:**

Enter the number : 3

3 is an odd

Enter the number : 10

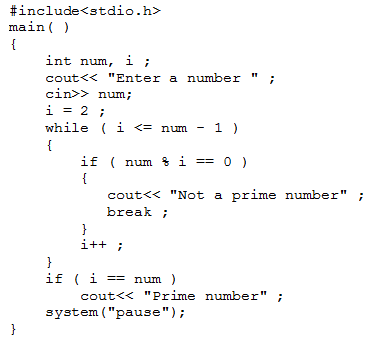
10 is an even

**Q4(a): What is the purpose of using *break and continue statements*?**

Ans:

* There are situations where we want to jump out of a loop instantly, without waiting to get back to the conditional test.
* The keyword breakallows us to do this
* When breakis encountered inside any loop, control automatically passes to the first statement after the loop.
* A break is usually associated with an if.
* The keyword break,breaks the control only from the loop in which it is placed

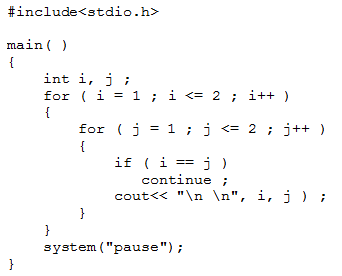
**Example: Write a program to determine whether a number is prime or not. A prime number is one, which is divisible only by 1 or itself.**

****

**Continue statements:**

* continue statement allows to take the control to the beginning of the loop, bypassing the statements inside the loop, which have not yet been executed
* A continue is usually associated with an if.

**Example:**

****

**Q4(b): Write a C++ program to find the sum of the following numbers:**

**1+2+3+………+10**

Ans:

#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;

}

**Output:**

Find the first 10 natural numbers:

---------------------------------------

The natural numbers are:

1 2 3 4 5 6 7 8 9 10

The sum of first 10 natural numbers: 55

**Q5: Expain the the following with proper examples**

1. **C++ Character set**
2. **Constants**
3. **Variables**
4. **Keywords**
5. **Relational Operators**

**Ans:**

**C++ Character set:**

Character set is a set of valid characters that a language can recognise. A character represents any letter, digits, or any other sign.

C++ has the following character set :

* Letters : A-Z, a-z
* Digits : 0-9
* Special Symbols : Space + - ∗ ⁄ ^ \ ( ) [ ] { } = != < > . ′ ″ $ , ; : % ! & \_ # <= >= @
* White Spaces : Blank space, Horizontal tab (→), Carriage return (↵), Newline, Form feed
* Other Characters : C++ can process any of the 256 ASCII characters as data or as literals.

**Example:**

#include<iostream.h>

#include<conio.h>

void main()

{

clrscr();

char letter, digit, special, white;

cout<<"Enter a Letter : ";

cin>>letter;

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

cout<<"Enter a Digit : ";

cin>>digit;

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

cout<<"Enter a special character : ";

cin>>special;

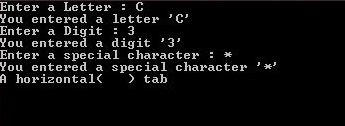
cout<<"You entered a special character '"<<special<<"'"<<"\n";

cout<<"A horizontal(\t) tab";

getch();

}

**Output:**

****

**Constant:**

In C++ program we can define constants in two ways

1. Using *#define* preprocessor directive
2. Using a *const* keyword

**Literals:** The values assigned to each constant variables are referred to as the *literals*. Generally, both terms, constants and literals are used interchangeably. For eg, “*const int = 5;*“, is a constant expression and the value 5 is refered to as constant integer literal.  
Refer here for various [Types of Literals in C++](https://www.geeksforgeeks.org/types-of-literals-in-c-c-with-examples/).

**Using *#define* preprocessor directive:** This directive is used to declare an alias name for existing variable or any value. We can use this to declare a constant

#define identifierName value

* **identifierName:** It is the name given to constant.
* **value:** This refers to any value assigned to identifierName.

**Example:**

#include<stdio.h>

#define val 10

#define floatVal 4.5

#define charVal 'G'

int main()

{

    printf("Integer Constant: %d\n",val);

    printf("Floating point Constant: %.1f\n",floatVal);

    printf("Character Constant: %c\n",charVal);

    return 0;

}

**Output:**

Integer Constant: 10

Floating point Constant: 4.5

Character Constant: G

**Using a const keyword**: Using const keyword to define constants is as simple as defining variables, the difference is you will have to precede the definition with a const keyword.

**Example:**

#include <stdio.h>

int main()

{

    // int constant

    const int intVal = 10;

    // Real constant

    const float floatVal = 4.14;

    // char constant

    const char charVal = 'A';

    // string constant

    const char stringVal[10] = "ABC";

    printf("Integer constant:%d \n", intVal );

    printf("Floating point constant: %.2f\n", floatVal );

    printf("Character constant: %c\n", charVal );

    printf("String constant: %s\n", stringVal);

    return 0;

}

**Output:**

Integer constant: 10

Floating point constant: 4.14

Character constant: A

String constant: ABC

**Variables:**

A variable is a name given to a memory location. It is the basic unit of storage in a program.

* The value stored in a variable can be changed during program execution.
* A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
* In C++, all the variables must be declared before use.

**How to declare variables?**

// Declaring a single variable

type variable\_name;

// Declaring multiple variables:

type variable1\_name, variable2\_name, variable3\_name;

A variable name can consist of alphabets (both upper and lower case), numbers and the underscore ‘\_’ character. However, the name must not start with a number.

**Example:**

#include <iostream>

using namespace std;

int main()

{

    // declaration and definition

    // of variable 'a123'

    char a123 = 'a';

    // This is also both declaration and definition

    // as 'b' is allocated memory and

    // assigned some garbage value.

    float b;

    // multiple declarations and definitions

    int \_c, \_d45, e;

    // Let us print a variable

    cout << a123 << endl;

    return 0;

}

**Output:**

a

**Keywords:**

Keywords are the reserved keywords that are defined by the compiler to perform the internal operation, written in lowercase. Keywords have some meaning which is defined by the compiler to accomplish a task in code, they cannot be used as a variable in programming. C++ provides 64 keywords – for, break, continue, switch, int float, double, char, try, catch, while, etc.

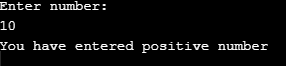
**List of Keywords:**

Auto, double, int, struct, Break, else, long, switch, Case, [enum](https://www.educba.com/enum-in-c-sharp/), register, typedef, Char, extern, return, [union](https://www.educba.com/c-union/), Const, float, short, unsigned, Continue, for, signed, void, Default, goto, sizeof, volatile, Do, if, static, while, Asm, dynamic\_cast, namespace, reinterpret\_cast, Bool, explicit, new, static\_cast, Catch false, operator, template, Class, friend, private, this, Const\_cast, inline, public, throw, Delete, mutable, protected, true, Try, typeid, typename, using, virtual, wchar\_t)

**Example:**

#include<iostream>  
using namespace std;  
int main()  
{  
int n;  
cout << "Enter number:" << endl;  
cin >> n;  
if(n > 0)  
{  
cout << "You have entered positive number";  
}  
return 0;  
}

**Output:**

****

**Relational Operators:**

Relational operators are also known for comparison operators. Relational operators are used to relating the condition, that is it compares the two values and prints the result.

Various C++ relational operators available are-

Operator Meaning

> Greater than  
> = Greater than or equal to  
= = Is equal to  
!= Is not equal to  
< Less than or equal to <= , Less than [/table]

**Example:**

#include <iostream>

using namespace std;

main() {

int a = 21;

int b = 10;

int c ;

if( a == b ) {

cout << "Line 1 - a is equal to b" << endl ;

} else {

cout << "Line 1 - a is not equal to b" << endl ;

}

if( a < b ) {

cout << "Line 2 - a is less than b" << endl ;

} else {

cout << "Line 2 - a is not less than b" << endl ;

}

if( a > b ) {

cout << "Line 3 - a is greater than b" << endl ;

} else {

cout << "Line 3 - a is not greater than b" << endl ;

}

/\* Let's change the values of a and b \*/

a = 5;

b = 20;

if( a <= b ) {

cout << "Line 4 - a is either less than \ or equal to b" << endl ;

}

if( b >= a ) {

cout << "Line 5 - b is either greater than \ or equal to b" << endl ;

}

return 0;

}

**Output:**

Line 1 - a is not equal to b

Line 2 - a is not less than b

Line 3 - a is greater than b

Line 4 - a is either less than or euqal to b

Line 5 - b is either greater than or equal to b

**>> END <<**