Syeda Reeha Quasar

14114802719

4C7

Aim

Write a program to generate a Fibonacci series using Copy Constructor.

Experiment - 14

Object Oriented Programming Lab

# **EXPERIMENT – 14**

## **Aim:**

Write a program to generate a Fibonacci series using Copy Constructor.

## **Source Code:**

#include <iostream>

using namespace std;

class fibonacci{

    private:

        unsigned long int f0, f1, fib;

    public :

        fibonacci(){

            f0 = 0;

            f1 = 1;

            fib = f0 + f1;

        }

        void update(){

            f0 = f1;

            f1 = fib;

            fib = f0 + f1;

        }

        void displayFib(int upto){

            for (int i = 0; i <= upto; i++) {

                cout << fib << "  ";

                update();

            }

        }

}; //end of class construction

int main(){

    fibonacci fibObj;

    int upto;

    cout << "Enter the number uptill you want Fibobnacci to be listed: ";

    cin >> upto;

    cout << endl;

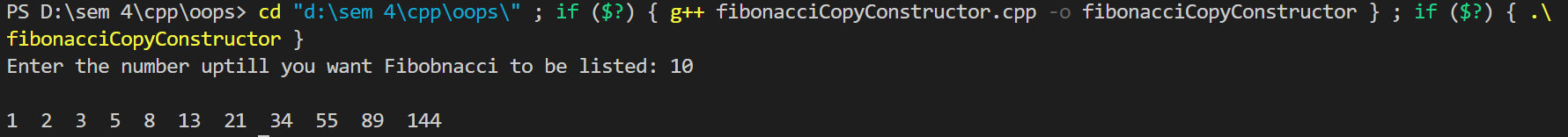
    fibObj.displayFib(upto);

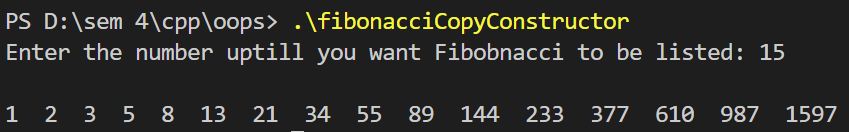
    cout << endl;

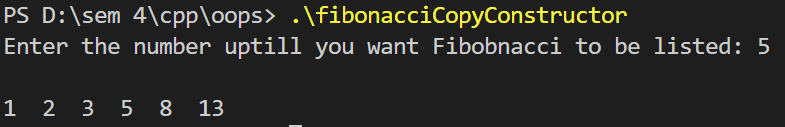
    return 0;

}

## **Output:**







# **Viva Questions**

#### Q1) Can a copy constructor accept an object of the same class as a parameter, in place of reference of the object? If No, why not possible?

Ans.

No. It is specified in the definition of the copy constructor itself. It should generate an error if a programmer specifies a copy constructor with a first argument that is an object and not a reference.

#### Q2) Are Constructors and destructors can declare as const?

Ans.

Constructors and destructors can’t be declared as const or volatile. They can, however, be invoked on const or volatile objects.

#### Q3) Can we make a copy constructor private?

Ans.

Yes, a copy constructor can be made private. When we make a copy constructor private in a class, objects of that class become non-copyable. This is particularly useful when our class has pointers or dynamically allocated resources.

#### Q4) Can you explain the order of execution in the constructor initialization list?

Ans.

When a class object is created using constructors, the execution order of constructors is:

* Constructors of Virtual base classes are executed, in the order that they appear in the base list.
* Constructors of nonvirtual base classes are executed, in the declaration order.
* Constructors of class members are executed in the declaration order (regardless of their order in the initialization list).
* The body of the constructor is executed.