Nithin S
221IT085

IT150 Lab Assignment 5

1. 1. Write a program in c++ to perform operator overloading on unary operator on unary operator ++ and -- , use the following specification to write the program.

a) Create Class temp

b) Class temp has private access modifier and variable count.

c) In public section of class temp create a constructor of class temp

d) Assign value 5 to variable count

e) Overload operator ++ which will increase the value of count by 1.

f) Overload operator -- which will decrease the value of count by 1.

CODE

```cpp
#include <iostream>

class temp {
private:
    int count;
public:
    // Constructor
    temp() : count(5) {}

    // Overloading the unary operator ++
    void operator++() {
        count++;
    }

    // Overloading the unary operator --
    void operator--() {
        count--;
    }

    // Function to get the value of count
    int getCount() const {
        return count;
    }
};

int main() {
    temp obj;

    std::cout << "Initial count: " << obj.getCount() << std::endl;

    char choice;
    std::cout << "Do you want to increment (++), decrement (--), or exit (E)? ";
    std::cin >> choice;

    while (choice != 'E') {
        if (choice == '+') {
            ++obj; // Using overloaded ++
            std::cout << "After increment: " << obj.getCount() << std::endl;
        } else if (choice == '-') {
            --obj; // Using overloaded --
            std::cout << "After decrement: " << obj.getCount() << std::endl;
        } else {
            std::cout << "Invalid choice!" << std::endl;
        }

        std::cout << "Do you want to increment (++), decrement (--), or exit (E)? ";
        std::cin >> choice;
    }

    return 0;
}
```

OUTPUT

```
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$ g++ 1.cpp
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$ ./a.out
Initial count: 5
Do you want to increment (++), decrement (--), or exit (E)? ++
After increment: 6
Do you want to increment (++), decrement (--), or exit (E)? After increment: 7
Do you want to increment (++), decrement (--), or exit (E)? --
After decrement: 6
Do you want to increment (++), decrement (--), or exit (E)? After decrement: 5
Do you want to increment (++), decrement (--), or exit (E)? E
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$
```

2.Write a C++ program to perform the addition and subtraction of two complex numbers using the binary (+) and (-) operator.

CODE

```cpp
#include <iostream>

class Complex {
private:
    double real;
    double imaginary;

public:

    Complex(double r = 0.0, double i = 0.0) : real(r), imaginary(i) {}


    Complex operator+(const Complex& other) const {
        return Complex(real + other.real, imaginary + other.imaginary);
    }


    Complex operator-(const Complex& other) const {
        return Complex(real - other.real, imaginary - other.imaginary);
    }


    void display() const {
        if (imaginary >= 0)
            std::cout << real << " + " << imaginary << "i";
        else
            std::cout << real << " - " << -imaginary << "i";
    }
};
```

```cpp
int main() {
    Complex num1, num2;
    double real, imaginary;


    std::cout << "Enter real and imaginary parts of first complex number: ";
    std::cin >> real >> imaginary;
    num1 = Complex(real, imaginary);


    std::cout << "Enter real and imaginary parts of second complex number: ";
    std::cin >> real >> imaginary;
    num2 = Complex(real, imaginary);


    Complex sum = num1 + num2;
    std::cout << "Sum: ";
    sum.display();
    std::cout << std::endl;


    Complex diff = num1 - num2;
    std::cout << "Difference: ";
    diff.display();
    std::cout << std::endl;

    return 0;
}
```

OUTPUT

```
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$ g++ 2.cpp
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$ ./a.out
Enter real and imaginary parts of first complex number: 4 5
Enter real and imaginary parts of second complex number: 6 8
Sum: 10 + 13i
Difference: -2 - 3i
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$
```

3. Run the below code and write your observation w.r.t virtual functions.

Given CODE

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


class base_class {
public:

    virtual void printit() {
        cout << endl << "Print from base class" << endl;
    }
};


class first_class : public base_class  {
public:
    void printit() override {
        cout << endl << "Print from first class" << endl;
    }
};


class second_class : public base_class  {
};


class coach_class : public base_class  {
public:
    void printit() override {
        cout << endl << "Print from coach class" << endl;
    }
};


class tourist_class : public coach_class  {
};
```

```
int main() {
    base_class base;
    base.printit();

    first_class first1;
    first1.printit();

    second_class second1;
    second1.printit();

    coach_class coach1;
    coach1.printit();

    tourist_class tour1;
    tour1.printit();

    return 0;
}
```

OUTPUT

```
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$ g++ 3.cpp
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$ ./a.out

Print from base class

Print from first class

Print from base class

Print from coach class

Print from coach class
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$
```

Observations:

Use of virtual Keyword: In the base_class, the printit() function is declared as virtual. This indicates that this function is intended to be overridden in derived classes.

Function Overriding in Derived Classes:

In first_class, coach_class, and tourist_class, the printit() function is redefined. This is function overriding, where the derived classes provide their own implementation of the printit() function.

In second_class, there's no explicit redefinition of the printit() function. In such a case, the printit() function of the base class will be invoked.

Dynamic Binding:

When an object of the base class (base_class) is created and its printit() function is called (base.printit()), the actual function called depends on the type of the object being referred to at runtime.

Similarly, for objects of derived classes (first_class, second_class, coach_class, tourist_class), dynamic binding ensures that the correct overridden version of printit() function is called based on the actual type of the object.

Main Function Usage:

In the main() function, objects of various classes are created and their printit() functions are called to demonstrate polymorphic behavior due to virtual functions.

This helps in illustrating how virtual functions work in a hierarchy of classes, ensuring that the appropriate function is called based on the object's dynamic type.

Overall, the code demonstrates the use of virtual functions and polymorphism in C++, allowing for more flexible and extensible class hierarchies.

4. Write a C++ program to :

a) perform function overriding by calling the overridden function of a member function from the child class

b) access overridden function using pointer of Base type that points to an object of Derived class

c) Access of Overridden Function to the Base Class

d) Access to Overridden Function

CODE

```cpp
#include <iostream>
#include <string>

class Base {
public:

    virtual void display() {
        std::cout << "Base class display()" << std::endl;
    }
};

class Derived : public Base {
public:

    void display() override {
        std::cout << "Derived class display()" << std::endl;
    }
};
```

```cpp
int main() {

    Derived derivedObj;
    derivedObj.display();


    Base *basePtr = &derivedObj;
    basePtr->display();


    Base &baseRef = derivedObj;
    baseRef.display();


    std::string choice;
    std::cout << "Do you want to call the overridden function? (yes/no): ";
    std::cin >> choice;

    if (choice == "yes") {

        Base *userChoicePtr;
        std::cout << "Enter '1' to create a Base object or '2' to create a Derived object: ";
        int type;
        std::cin >> type;

        if (type == 1) {
            userChoicePtr = new Base();
        } else {
            userChoicePtr = new Derived();
        }

        userChoicePtr->display();

        delete userChoicePtr;
    }

    return 0;
```

OUTPUT

```
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$ g++ 4.cpp
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$ ./a.out
Derived class display()
Derived class display()
Derived class display()
Do you want to call the overridden function? (yes/no): yes
Enter '1' to create a Base object or '2' to create a Derived object: 1
Base class display()
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_5$
```