Ques 1Create a class FLOAT that contains one float data member.Overload all the four arithmetic operators so that they operate on the objects of FLOAT.

Solution

#include<iostream>

using namespace std;

class FLOAT

{

    float a;

public:

            FLOAT(float a)

            {

                this->a=a;

            }

            float operator+(FLOAT F)

            {

                return(a+F.a);

            }

            float operator-(FLOAT F)

            {

                return(a-F.a);

            }

            float operator\*(FLOAT F)

            {

                return(a\*F.a);

            }

            float operator/(FLOAT F)

            {

                return(a/F.a);

            }

};

int main()

{

    FLOAT f1(10.10),f2(20.20);

    cout<<" Operator + : "<<f1+f2<<endl;

    cout<<" Operator - : "<<f1-f2<<endl;

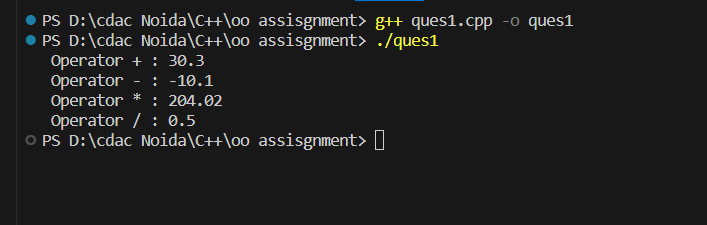
    cout<<" Operator \* : "<<f1\*f2<<endl;

    cout<<" Operator / : "<<f1/f2<<endl;

    return 0;

}

Output



Ques 2//Define a class string. Overlaod ==operator to compare 2 strings.

#include <iostream>

using namespace std;

#include <cstring>

class String {

private:

    char\* str;

public:

    String(const char\* s) {

        str = new char[strlen(s) + 1];

        strcpy(str, s);

    }

~String() {

        delete[] str;

    }

bool operator==(const String& other) const {

        return strcmp(str, other.str) == 0;

    }

const char\* getValue() const {

        return str;

    }

};

int main() {

    String str1("Hello");

    String str2("World");

    String str3("Hello");

if (str1 == str2) {

        cout << "str1 and str2 are equal." << endl;

    } else {

        cout << "str1 and str2 are not equal." << endl;

    }

    if (str1 == str3) {

        cout << "str1 and str3 are equal." << endl;

    } else {

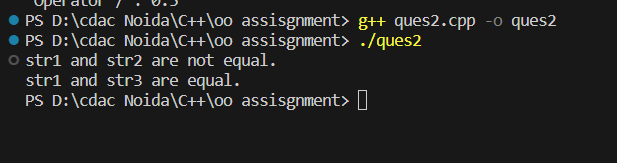
        cout << "str1 and str3 are not equal." << endl;

    }

    return 0;

}

Output



Ques 3 Create a Complex class that has real(int) and img(int) as member data, and has getData

and showData functions. Then also overload the following operators for Complex class. =,

==, +, ++, --,

#include <iostream>

using namespace std;

class Complex {

private:

    int real;

    int imag;

public:

    Complex(int r = 0, int i = 0) : real(r), imag(i) {}

int getReal() const {

        return real;

    }

    int getImag() const {

        return imag;

    }

Complex& operator=(const Complex& other) {

        real = other.real;

        imag = other.imag;

        return \*this;

    }

    bool operator==(const Complex& other) const {

        return (real == other.real) && (imag == other.imag);

    }

    Complex operator+(const Complex& other) const {

        return Complex(real + other.real, imag + other.imag);

    }

    Complex& operator++() {

        real++;

        imag++;

        return \*this;

    }

    Complex operator++(int) {

        Complex temp(\*this);

        ++(\*this);

        return temp;

    }

    Complex& operator--() {

        real--;

        imag--;

        return \*this;

    }

    Complex operator--(int) {

        Complex temp(\*this);

        --(\*this);

        return temp;

    }

    void showData() const {

        cout << real << " + " << imag << "i" << endl;

    }

};

int main() {

    Complex c1(3, 4);

    Complex c2(2, 6);

    Complex result;

    result = c1;

    if (c1 == c2) {

        cout << "c1 and c2 are equal." << endl;

    } else {

        cout << "c1 and c2 are not equal." << endl;

    }

    result = c1 + c2;

    cout << "c1 + c2 = ";

    result.showData();

    ++c1;

    cout << "c1 after pre-increment: ";

    c1.showData();

    c2++;

    cout << "c2 after post-increment: ";

    c2.showData();

    --c1;

    cout << "c1 after pre-decrement: ";

    c1.showData();

    c2--;

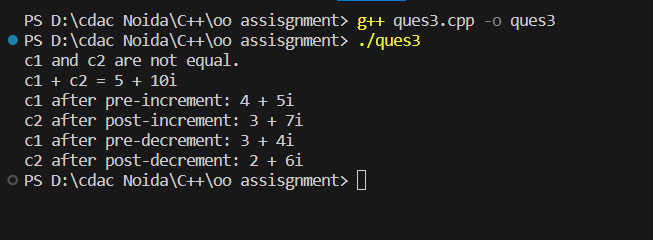
    cout << "c2 after post-decrement: ";

    c2.showData();

    return 0;

}

Output



ques 4. Write a C++ program to overload ‘!’ operator using friend function

#include <iostream>

using namespace std;

class MyBoolean {

private:

bool value;

public:

MyBoolean(bool val){

value=val;

}

friend bool operator!(const MyBoolean& obj);

bool getValue() const {

return value;

}

};

bool operator!(const MyBoolean& obj) {

return !obj.value;

}

int main() {

MyBoolean myBool(true);

MyBoolean result = !myBool;

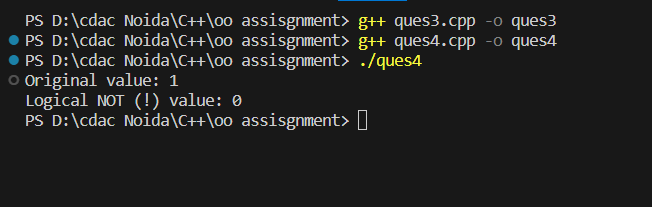
cout << "Original value: " << myBool.getValue() << endl;

cout << "Logical NOT (!) value: " << result.getValue() << endl;

return 0;

}

Output



Ques 5 Read a value of distance from one object and add with a value in

another object using friend function.

#include <iostream>

using namespace std;

class Distance {

private:

    int meters;

public:

    Distance(int m) : meters(m) {}

    friend Distance addDistances(const Distance& d1, const Distance& d2);

    int getMeters() const {

        return meters;

    }

};

Distance addDistances(const Distance& d1, const Distance& d2) {

    int totalMeters = d1.meters + d2.meters;

    return Distance(totalMeters);

}

int main() {

    Distance distance1(10);

    Distance distance2(5);

Distance totalDistance = addDistances(distance1, distance2);

    cout << "Total Distance: " << totalDistance.getMeters() << " meters" << endl;

    return 0;

}

Output

