Assignment operator overloading

1. Create a class FLOAT that contains one float data member .Overload all the four

arithmetic operators so that they operate on the objects of FLOAT.

#include<iostream>

using namespace std;

class Float

{

float num;

public:

void accept()

{

cout<<"\n Enter Number : ";

cin>>num;

}

Float operator+(Float &a)

{

Float t;

t.num=num+a.num;

return t;

}

Float operator-(Float &a)

{

Float t;

t.num=num-a.num;

return t;

}

Float operator\*(Float &a)

{

Float t;

t.num=num\*a.num;

return t;

}

Float operator/(Float &a)

{

Float t;

t.num=num/a.num;

return t;

}

void display()

{

cout<<num;

}

};

int main()

{

Float a1, a2, a3;

a1.accept();

a2.accept();

a3=a1+a2;

cout<<"\n\n Addition of Two Numbers : ";

a3.display();

a3=a1-a2;

cout<<"\n\n Subtraction of Two Numbers : ";

a3.display();

a3=a1\*a2;

cout<<"\n\n Multiplication of Two Numbers : ";

a3.display();

a3=a1/a2;

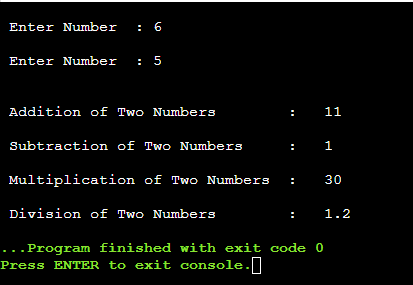
cout<<"\n\n Division of Two Numbers : ";

a3.display();

return 0;

}

OUTPUT:



2. Define a class string. Overload =operator to compare 2 strings.

#include<iostream>

#include<stdio.h>

#include<string.h>

using namespace std;

class String

{

char str[20];

public:

void getdata()

{

gets(str);

}

int operator ==(String s)

{

if(!strcmp(str,s.str))

return 1;

return 0;

}

};

int main()

{

String s1,s2;

cout<<"Enter first string :: ";

s1.getdata();

cout<<"\nEnter second string :: ";

s2.getdata();

if(s1==s2)

{

cout<<"\nStrigs are Equal\n";

}

else

{

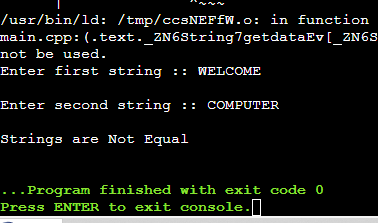
cout<<"\nStrings are Not Equal\n";

}

return 0;

}

OUTPUT:



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 complex1

{

private:

int real, imag;

public:

complex1(int r = 0, int i =0)

{

real = r;

imag = i;

}

complex1 operator+(complex1 const& obj)

{

complex1 res;

res.real = real + obj.real;

res.imag = imag + obj.imag;

return res;

}

void operator-()

{

real=-real;

imag=-imag;

}

void operator++()

{

++real;

++imag;

}

void operator++(int)

{

++real;

++imag;

}

friend complex1 operator+(int i,complex1 c)

{

complex1 t;

t.real=i+c.real;

t.imag=i+c.imag;

return t;

}

friend ostream& operator<<(ostream &out , const complex1 &c)

{

out << c.real;

out << "+i" << c.imag << endl;

return out;

}

friend istream& operator>>(istream &in, complex1 &c)

{

cout<< "Enter Real Part ";

in>> c.real;

cout<< "Enter Imaginary Part ";

in>>c.imag;

return in;

}

void display()

{

cout << real << " + i" << imag << '\n';

}

};

int main()

{

complex1 c1(10, 5), c2(8, 4);

complex1 c3 = c1 + c2;

c3.display();

-c2;

c2.display();

++c2;

c2.display();

c2++;

c2.display();

c2=8+c2;

c2.display();

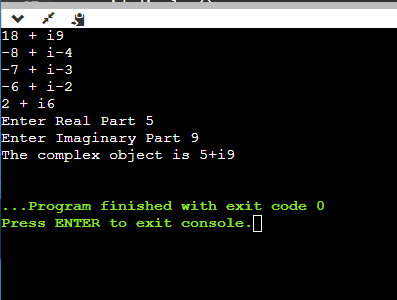
cin>>c1;

cout<<"The complex object is ";

cout<<c1;

return 0;

}



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

#include <iostream>

using namespace std;

class logical

{

private:

int value;

public:

logical(int val)

{

value=val;

}

friend logical operator!(const logical& obj);

void display()

{

cout <<"Value:"<< value <<endl;

}

};

logical operator!(const logical& obj)

{

return logical(!obj.value);

}

int main()

{

logical obj(5);

logical result = !obj;

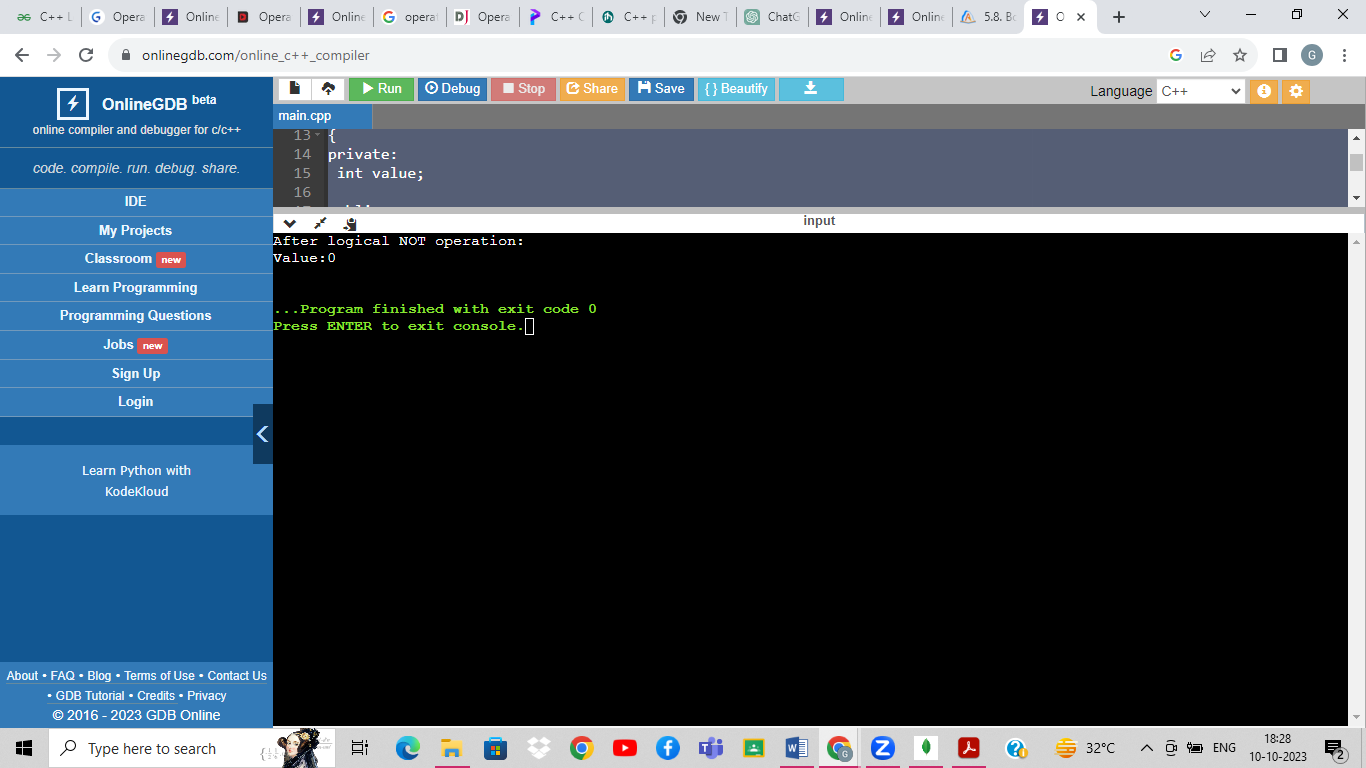
cout << "After logical NOT operation:" <<endl;

result.display();

return 0;

}

o/p:



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 meter;

public:

Distance()

{

meter=5;

}

friend int add(Distance);

};

int add(Distance d)

{

d.meter=d.meter+5;

return d.meter;

}

int main()

{

Distance d;

cout<<"Distance: "<<add(d);

return 0;

}

o/p:

