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 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(14.14),f2(20.20);

cout<<"\n\n Operator + : "<<f1+f2;

cout<<"\n\n Operator - : "<<f1-f2;

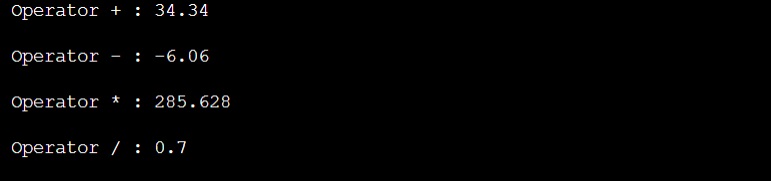
cout<<"\n\n Operator \* : "<<f1\*f2;

cout<<"\n\n Operator / : "<<f1/f2;

return 0;

}

O/p:



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

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

#include <iostream>

#include <string.h>

using namespace std;

class Comparesing

{

public:

char str[25];

Comparesing(char str1[])

{

strcpy(this->str,str1);

}

int operator==(Comparesing s2)

{

if(strcmp(str,s2.str)==0)return 1;

else

return 0;

}

int operator<=(Comparesing s3)

{

if(strlen(str)<=strlen(s3.str))

return 1;

else

return 0;

}

int operator>=(Comparesing s3)

{

if(strlen(str)>=strlen(s3.str))

return 1;

else

return 0;

}

};

void compare(Comparesing s1 ,Comparesing s2)

{

if(s1==s2)

cout<<s1.str<<" is equal to "<<s2.str<<endl;

else

{

cout<<s1.str<<" is not equal to "<<s2.str<<endl;

if(s1>=s2)

cout<<s1.str<<" is greater than "<<s2.str<<endl;

else

cout<<s2.str<<" is greater than "<<s1.str<<endl;

}

}

void testcase1()

{

char str1[]="Computer";

char str2[]="Laptop";

Comparesing s1(str1);

Comparesing s2(str2);

cout<<"Comparing \""<<s1.str<<"\" and \""<<s2.str<<"\""<<endl;

compare(s1,s2);

}

void testcase2()

{

char str1[]="Mobile";

char str2[]="Mobile";

Comparesing s1(str1);

Comparesing s2(str2);

cout<<"\n\nComparing \""<<s1.str<<"\" and \""<<s2.str<<"\""<<endl;

compare(s1,s2);

}

int main()

{

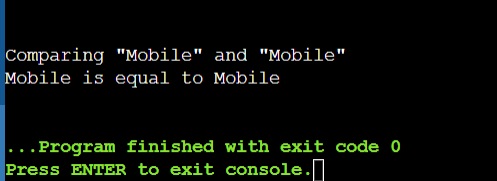
testcase1();

testcase2();

return 0;

}

O/p:



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;

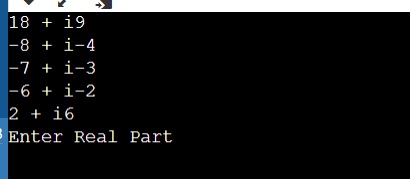
cout<<"The complex object is ";

cout<<c1;

return 0;

}

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;

// friend function

friend int addFive(Distance);

public:

Distance() : meter(0) {}

};

// friend function definition

int addFive(Distance d) {

//accessing private members from the friend function

d.meter += 5;

return d.meter;

}

int main() {

Distance D;

cout << "Distance: " << addFive(D);

return 0;

}

O/p:

