***Name - Akriti Choudhary***

***Roll number - 2005776***

***Lab 10***

***Subject - OOP lab***

***Class - B14***

***Branch - CSE***

***Date- 28/10/2021***

**Question 1)** **WAP to overload following operators for class distance, which stores the distance in**

**feet and inches.**

**a) Binary + to**

**-add two objects (D3=D1+D2)**

**-Add an object to an integer, where the integer should be added to the inches value ( D2=4+D1)**

**-Multiply an object to an integer, where the integer should be multiplied to the inches value ( D2=D1 \* 4)**

**b) Unary -**

#include <iostream>

using namespace std;

class Distance

{

int foot;

int inches;

public:

Distance()

{

foot = 0;

inches = 0;

}

Distance(int a, int b)

{

foot = a;

inches = b;

}

void display()

{

cout << "Foot = " << foot << endl;

cout << "Inches = " << inches << endl;

}

friend Distance operator+(Distance &, Distance &);

friend Distance operator+(int d, Distance &);

friend Distance operator\*(Distance &b, int d);

friend Distance operator-(Distance &);

};

Distance operator+(Distance &obj1, Distance &obj2)

{

Distance obj;

obj.foot = obj1.foot + obj2.foot;

obj.inches = obj1.inches + obj2.inches;

if (obj.inches > 12)

{

obj.foot += obj.inches / 12;

obj.inches = obj.inches % 12;

}

return obj;

}

Distance operator+(int d, Distance &b)

{

Distance ob;

ob.foot = b.foot;

ob.inches = d + b.inches;

if (ob.inches > 12)

{

ob.foot += ob.inches / 12;

ob.inches = ob.inches % 12;

}

return ob;

}

Distance operator\*(Distance &b, int d)

{

Distance ob;

ob.foot = b.foot ;

ob.inches = b.inches \* d;

if (ob.inches > 12)

{

ob.foot += ob.inches / 12;

ob.inches = ob.inches % 12;

}

return ob;

}

Distance operator-(Distance &b)

{

Distance ob;

ob.foot = -(b.foot);

ob.inches = -(b.inches);

return ob;

}

int main()

{

int f1, f2, i1, i2;

cout << "Enter distance1 in feet and inches\n";

cin >> f1 >> i1;

Distance obj1(f1, i1);

cout << "Enter distance2 in feet and inches\n";

cin >> f2 >> i2;

Distance obj2(f2, i2);

Distance obj;

obj = obj1 + obj2;

obj.display();

Distance ob;

int d;

cout << "Enter an integer to add in the resulting inches\n";

cin >> d;

ob = d + obj;

cout << "Displaying the resulting(ob = d + obj) distance\n";

ob.display();

ob = ob \* d;

cout << "Displaying the resulting(ob = obj \* d) distance\n";

ob.display();

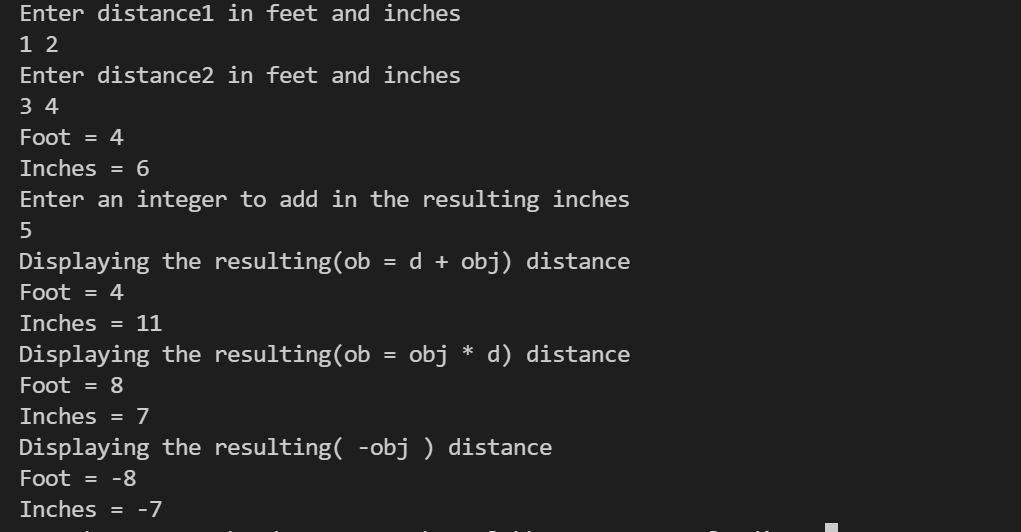
cout << "Displaying the resulting( -obj ) distance\n";

ob = -ob;

ob.display();

return 0;

}



**Question 2) Create a class to store an integer array. Overload insertion and extraction operator to input and display the array elements.**

#include <iostream>

using namespace std;

class Array

{

int arr[5];

public:

Array()

{

for (int i = 0; i < 5; ++i)

{

arr[i] = 0;

}

}

friend istream &operator>>(istream &, Array &);

friend ostream &operator<<(ostream &, Array &);

};

istream &operator>>(istream &cin, Array &ob)

{

int value = 0;

for (int i = 0; i < 5; ++i)

{

cout << "Enter the element\n";

cin >> value;

ob.arr[i] = value;

}

return cin;

}

ostream &operator<<(ostream &cout, Array &ob)

{

for (int i = 0; i < 5; ++i)

{

cout << ob.arr[i] << " ";

}

return cout;

}

int main()

{

Array ob;

cout<<"Entering The elements of the object using the overloaded extraction operator\n";

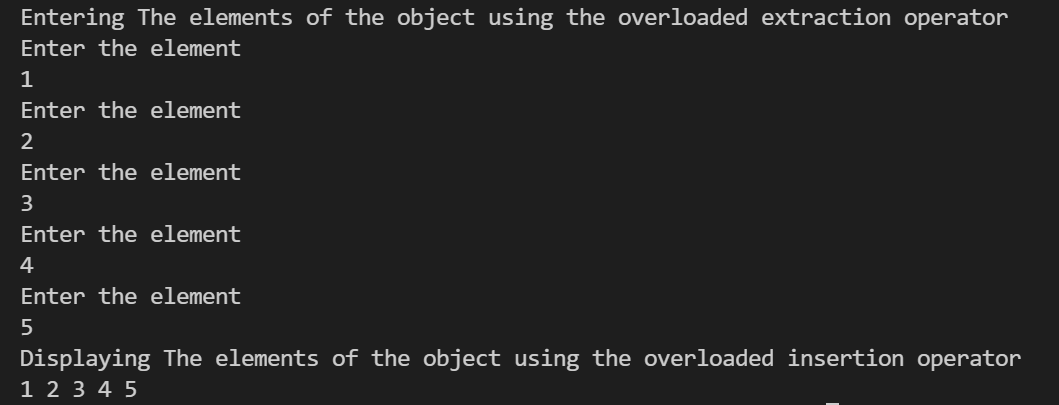
cin>>ob;

cout<<"Displaying The elements of the object using the overloaded insertion operator\n";

cout<<ob;

return 0;

}



**Question 3) Create a class which a complex number. Add two objects and display the resultant object .Overload the ++ (post and pre) operator for the class.**

#include <iostream>

using namespace std;

class Complex

{

int a;

int b;

public:

Complex()

{

a = 0;

b = 0;

}

Complex(int p, int q)

{

a = p;

b = q;

}

void display()

{

if (b >= 0)

{

cout << a << " "

<< "+ i" << b << "\n";

}

else

{

cout << a << " "

<< "- i" << -b << "\n";

}

}

Complex operator++(int)

{

Complex obj;

obj.a = a++;

obj.b = b++;

return obj;

}

Complex operator++()

{

Complex obj;

obj.a = ++a;

obj.b = ++b;

return obj;

}

friend Complex operator+(Complex obj1, Complex obj2);

};

Complex operator+(Complex obj1, Complex obj2)

{

Complex obj;

obj.a = obj1.a + obj2.a;

obj.b = obj1.b + obj2.b;

return obj;

}

int main()

{

int a1, b1;

int a2, b2;

cout << "Enter the real part of complex number1 :";

cin >> a1;

cout << "Enter the imaginary part of complex number1 :";

cin >> b1;

cout << "Enter the real part of complex number2 :";

cin >> a2;

cout << "Enter the imaginary part of complex number2 :";

cin >> b2;

Complex obj1(a1, b1), obj2(a2, b2);

Complex obj , obj3 , obj4;

obj = obj1 + obj2;

obj.display();

cout << "Displaying obj++ \n";

obj3 = obj++;

obj3.display();

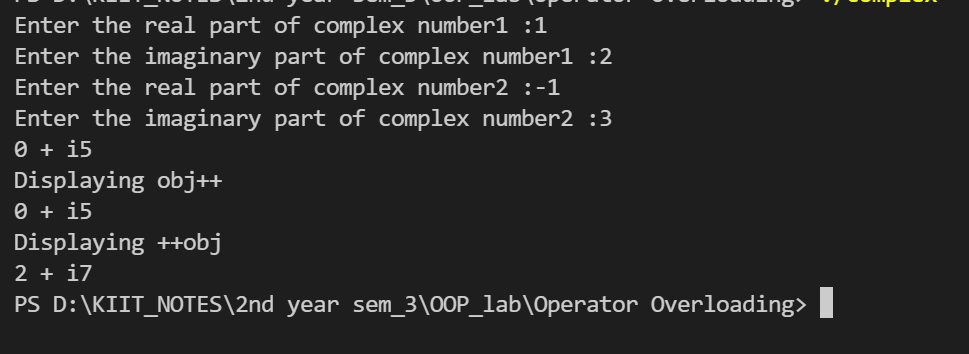
cout << "Displaying ++obj \n";

obj4 = ++obj;

obj4.display();

return 0;

}



**Question 4) Create a class which allocates the memory for a string through dynamic constructor. Overload the binary + to concatenate two strings and display it.**

#include <iostream>

#include <string>

using namespace std;

class abc

{

string \*s = NULL;

public:

abc()

{

s = new string();

\*s = " ";

}

abc(string value)

{

s = new string();

\*s = value;

}

~abc()

{

free(s);

}

friend abc operator+(abc &obj1, abc &obj2);

};

abc operator+ (abc &obj1, abc &obj2)

{

abc obj;

(\*(obj.s)) = (\*(obj1.s)).append(\*(obj2.s));

cout<<"The resulting string is : "<<\*(obj.s)<<endl;

return obj;

}

int main()

{

string v1, v2;

cout << "Enter string 1 : \n";

cin >> v1;

cout << "Enter string 2 : \n";

cin >> v2;

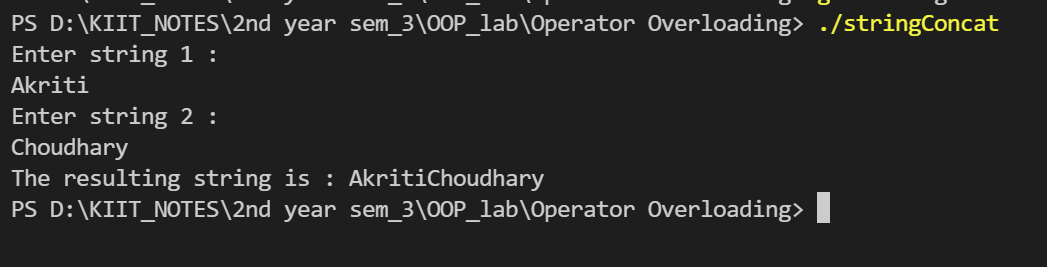
abc obj;

abc obj1(v1), obj2(v2);

obj = obj1 + obj2;

return 0;

}



**Question 5)WAP to add two objects of time class. Overload the operator ‘==’ to compare two objects and display whether they are equal or not. Overload the assignment operator.**

#include <iostream>

using namespace std;

class Time

{

int h;

int m;

public:

Time()

{

h = 0;

m = 0;

}

Time(int hh, int mm)

{

h = hh;

m = mm;

}

void operator=(Time obj1)

{

h = obj1.h;

m = obj1.m;

}

void display()

{

cout << "hours : " << h << " minutes : " << m << "\n";

}

friend void operator==(Time obj1, Time obj2);

friend Time operator+(Time obj1, Time obj2);

};

void operator==(Time obj1, Time obj2)

{

if (obj1.h == obj2.h && obj1.m == obj2.m)

{

cout << "Equal\n";

}

else

{

cout << "Unequal\n";

}

}

Time operator+(Time obj1, Time obj2)

{

Time obj;

obj.h = obj1.h + obj2.h;

obj.m = obj1.m + obj2.m;

if (obj.m >= 60)

{

obj.h += 1;

obj.m -= 60;

}

return obj;

}

int main()

{

int h1, m1, h2, m2;

Time obj;

cout << "Enter time1 in hours and minutes\n";

cin >> h1 >> m1;

cout << "Enter time2 in hours and minutes\n";

cin >> h2 >> m2;

Time obj1(h1, m1);

Time obj2;

Time obj3(h2, m2);

cout << "Displaying obj1\n";

obj1.display();

cout << "Displaying obj2\n";

obj2.display();

cout << "Displaying obj3\n";

obj3.display();

cout << "\n";

cout << "Copying obj3 in obj2\n";

obj2 = obj3;

cout << "Displaying obj2\n";

obj2.display();

cout << "\n";

cout << "Verifying obj1 == obj2 \n";

obj1 == obj2;

cout << "\n";

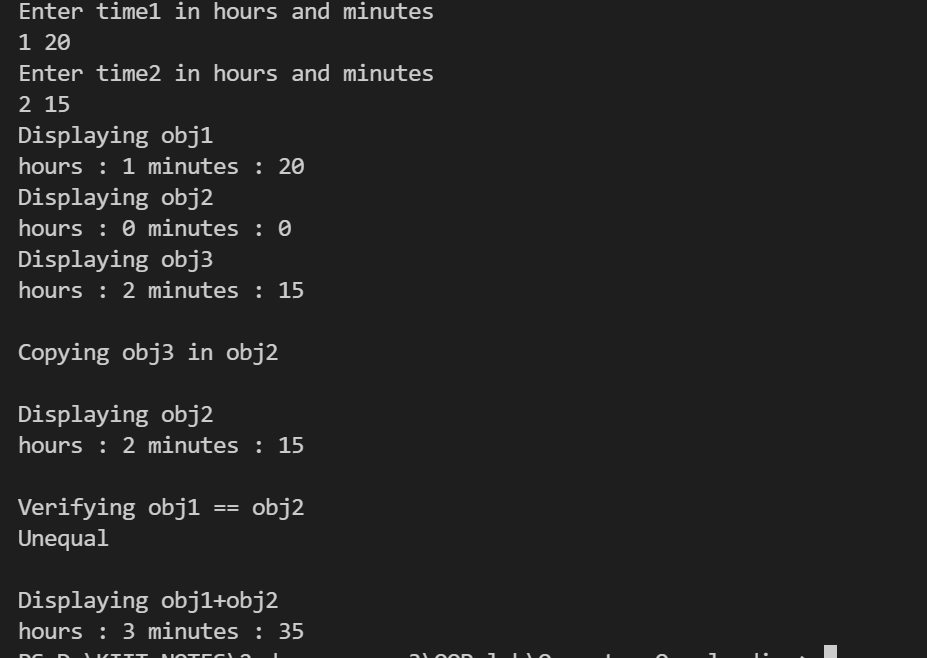
obj = obj1 + obj2;

cout << "Displaying obj1+obj2\n";

obj.display();

return 0;

}



**Question 6) WAP to add two objects of distance class. Overload the operator ‘>’ to compare two.**

**objects and return the object with larger time value and display it. Overload the ‘==’operator to compare and display whether two given objects contain same distance**

**value.**

#include <iostream>

using namespace std;

class Distance

{

int distKm;

int distM;

public:

Distance()

{

distKm = 0;

distM = 0;

}

Distance(int km, int m)

{

distKm = km;

if (m < 1000)

{

distM = m;

}

else

{

distM = 0;

}

}

void display()

{

cout << distKm << " km " << distM << " m \n";

}

friend Distance operator+(Distance obj1, Distance obj2);

friend Distance operator>(Distance obj1, Distance obj2);

friend int operator==(Distance obj1, Distance obj2);

};

Distance operator+(Distance obj1, Distance obj2)

{

Distance obj;

obj.distKm = obj1.distKm + obj2.distKm;

obj.distM = obj1.distM + obj2.distM;

if (obj.distM > 1000)

{

obj.distM -= 1000;

obj.distKm += 1;

}

return obj;

}

Distance operator>(Distance obj1, Distance obj2)

{

if (obj1.distKm > obj2.distKm)

{

return obj1;

}

else if (obj1.distM > obj2.distM)

{

return obj1;

}

else

{

return obj2;

}

}

int operator==(Distance obj1, Distance obj2)

{

if ((obj1.distKm == obj2.distKm) && (obj1.distM == obj2.distM))

{

return 1;

}

else

{

return 0;

}

}

int main()

{

int km1, m1, km2, m2;

cout << "Enter the distance1 in km and m(< 1000)\n";

cin >> km1 >> m1;

cout << "Enter the distance2 in km and m(< 1000)\n";

cin >> km2 >> m2;

Distance obj;

Distance ob;

Distance obj1(km1, m1);

Distance obj2(km2, m2);

ob = obj1 > obj2;

cout<<"Displaying the greater value object\n";

ob.display();

int p = obj1 == obj2;

if (p == 1)

{

cout << "Distance 1 and Distance 2 are equal\n";

}

else

{

cout << "Distance 1 and Distance 2 are not equal\n";

}

obj = obj1 + obj2;

cout << "Addition of both the distances = ";

obj.display();

return 0;

}

