OBJECT ORIENTED PROGRAMMING

LAB 8-OPERATOR OVERLOADING

• Name :HITU RAJ

• Roll no. :2005025

• Branch :CSE

/\*Q1A>.WAP to overload following operators for class distance, which stores the

distance in feet and inches.

Binary + to

-add two objects (D3=D1+D2)

-

Unary -\*/

#include <bits/stdc++.h>

using namespace std;

class distance\_025

{

    int feet\_025;

    int inches\_025;

public:

    void input()

    {

        cin >> feet\_025 >> inches\_025;

    }

    void display()

    {

        cout << "sum is---->" << feet\_025 << "'' " << inches\_025 << "'\n";

    }

    distance\_025 operator+(distance\_025);

};

distance\_025 distance\_025 ::operator+(distance\_025 b)

{

    distance\_025 t;

    t.feet\_025 = feet\_025 + b.feet\_025;

    t.inches\_025 = inches\_025 + b.inches\_025;

    if (t.inches\_025 > 11)

    {

        t.feet\_025 = t.feet\_025 + t.inches\_025 / 12;

        t.inches\_025 = t.inches\_025 % 12;

    }

    return t;

}

int main()

{

    distance\_025 a[2], sum;

    for (int i = 0; i < 2; i++)

    {

        cout << "enter" << i + 1 << " distance in feet and inches ";

        a[i].input();

    }

    sum = a[0] + a[1];

    sum.display();

    return 0;

}

OUTPUT -1



/\*Q1B>Add an object to an integer, where the integer should be added

to the inches value ( D2=4+D1)\*/

#include <bits/stdc++.h>

using namespace std;

class distance\_025

{

    int feet\_025;

    int inches\_025;

public:

    void input()

    {

        cin >> feet\_025 >> inches\_025;

    }

    void display()

    {

        cout << "sum is---->" << feet\_025 << "'' " << inches\_025 << "'\n";

    }

 friend distance\_025 operator +(int,distance\_025);

};

distance\_025 operator +(int a, distance\_025 b)

{

    distance\_025 t;

    t.feet\_025 = a + b.feet\_025;

    t.inches\_025 =  b.inches\_025;

    if (t.inches\_025 > 11)

    {

        t.feet\_025 = t.feet\_025 + t.inches\_025 / 12;

        t.inches\_025 = t.inches\_025 % 12;

    }

    return t;

}

int main()

{

    distance\_025 a, sum;

    int x;

        cout << "enter distance in feet and inches ";

        a.input();

        cout<<"\nEnter a number which u want to add: ";

        cin>>x;

    sum = x + a;

    sum.display();

    return 0;

}

OUTPUT -1-b



/\*2Create a class to store an integer array. Overload insertion and extraction

operator to input and display the array elements.n\*/

#include <iostream>

using namespace std;

class array

{

    int arr[5];

    friend void operator>>(istream &in, array &c)

    {

        cout << "\n Enter the 5 elements in array ";

        for (int i = 0; i < 5; i++)

        {

            in >> c.arr[i];

        }

    }

    friend void operator<<(ostream &out, const array &c)

    {

        cout << "\narray is ";

        for (int i = 0; i < 5; i++)

        {

            out << c.arr[i] << "\t";

        }

    }

};

int main()

{

    array a;

    cin >> a;

    cout << a;

    return 0;

}

OUTPUT -2



/\*Q3.Create a class time 1 which has three data members: hours, minute and

second. Overload the ++ (post and pre) and -- (post and pre )operator for the

class. Using friend and member function.

 Create a class which stores time1 in hh:mm format. Include all the constructors.

The parameterized constructor should initialize the minute value to zero, if it

is not provided.\*/

//using friend function

#include <iostream>

using namespace std;

class time1

{

    int hr\_025;

    int min\_025;

    int sec;

public:

    time1()

    {

        hr\_025 = 0;

        min\_025 = 0;

        sec = 0;

    }

    void input(int t, int j, int k) // parameterized

    {

        hr\_025 = t;

        min\_025 = j;

        sec = k;

    }

    friend void operator++(time1 &a, int)

    {

        a.hr\_025++;

        a.min\_025++;

        a.sec++;

    }

    friend void operator++(time1 &a)

    {

        ++a.hr\_025;

        ++a.min\_025;

        ++a.sec;

    }

    friend void operator--(time1 &a, int)

    {

        a.hr\_025--;

        a.min\_025--;

        a.sec--;

    }

    friend void operator--(time1 &a)

    {

        --a.hr\_025;

        --a.min\_025;

        --a.sec;

    }

    void display()

    {

        cout << hr\_025 << ":" << min\_025 << ":" << sec << endl;

    }

} a[2], t;

int main()

{

    int hr, min, sec,c;

    cout << "enter  time\n";

    for (int i = 0; i < 1; i++)

    {

        cout << "enter" << i + 1 << " time in  hr min sec";

        cin >> hr >> min >> sec;

        a[i].input(hr, min, sec);

    }

    cout<<"press1 to post ++\n";

    cout<<"press2 to pre ++\n";

    cout<<"press3 to post --\n";

    cout<<"press4 to pre --\n";

    cin>>c;

    switch (c)

    {

    case 1:

        a[0].display();

        a[0]++;

        a[0].display();

        break;

    case 2:

        a[0].display();

        ++a[0];

        a[0].display();

        break;

    case 3:

        a[0].display();

        a[0]--;

        a[0].display();

        break;

    case 4:

        a[0].display();

        --a[0];

        a[0].display();

        break;

    default:

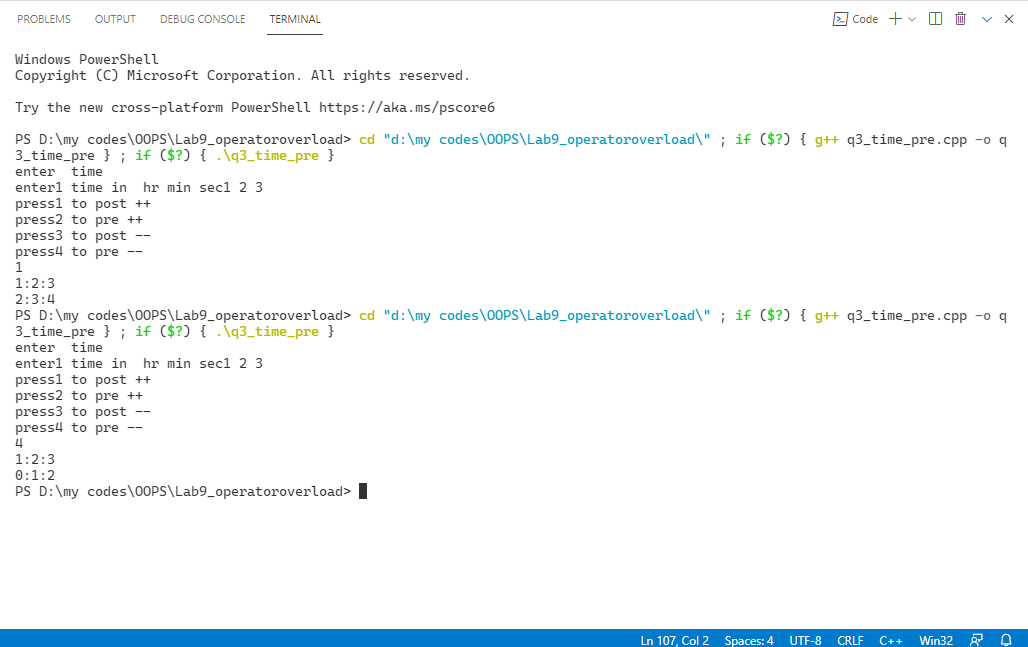
        break;

    }

    return 0;

}

OUTPUT -3



/\*Q4Create a class which allocates the memory for a string through dynamic

constructor. Overload the binary + to concatenate two strings and display it.

Overload the assignment operator.\*/

#include <string.h>

#include <iostream>

using namespace std;

class stringadd

{

    char \*s;

public:

    stringadd(char \*a)

    {

        s = new char[10];

        strcpy(s, a);

    }

    char\* operator+(stringadd a)

    {

        return strcat(s, a.s);

    }

};

int main()

{

    char aa[10], be[10];

     char \*res;//=new char[100];

    cout << "Enter 2 string u want to add ";

    gets(aa);

    gets(be);

    stringadd a(aa), b(be);

    res = a + b;

    cout<<"concatinated string is : "<<res;

    return 0;

}

OUTPUT -4



/\*Q5 WAP to Overload the operator ‘==’ to compare two objects of complex class

and display whether they are equal or not. Overload the assignment operator.\*/

#include <iostream>

using namespace std;

class Complex

{

    int real\_025;

    int img\_025;

    ;

public:

    void input()

    {

        cin >> real\_025 >> img\_025;

        ;

    }

    bool operator==(Complex s2)

    {

        if (s2.real\_025 == real\_025 && img\_025 == s2.img\_025)

            return true;

        else

            return false;

    }

    Complex operator =(Complex s2)

    {

        real\_025=s2.real\_025;

        img\_025=s2.img\_025;

        return \*this;

    }

    void display()

    {

        cout << "\nThe complex no. is " << real\_025 << "+i" << img\_025<<"\n";

        ;

    }

};

int main()

{

    int n\_025, a\_025;

    Complex s[2], c\_025;

    for (int i = 0; i < 2; i++)

    {

        cout << "enter" << i + 1 << " complex no.";

        s[i].input();

    }

    cout

        << "\npress 1 if you want to compare both the complex no.";

      cout  << "\npress 2 if you want to asign  2nd complex no.to the first\n";

    cin >> n\_025;

    switch (n\_025)

    {

    case 1:

        if (s[0] == s[1])

            cout << "both the complex no. are same\n";

        else

            cout << "both are different \n";

        break;

        case 2:

        s[0]=s[1];

        cout<<"value of the 2 comple no. is\n";

        s[0].display();

        s[1].display();

        break;

    default:

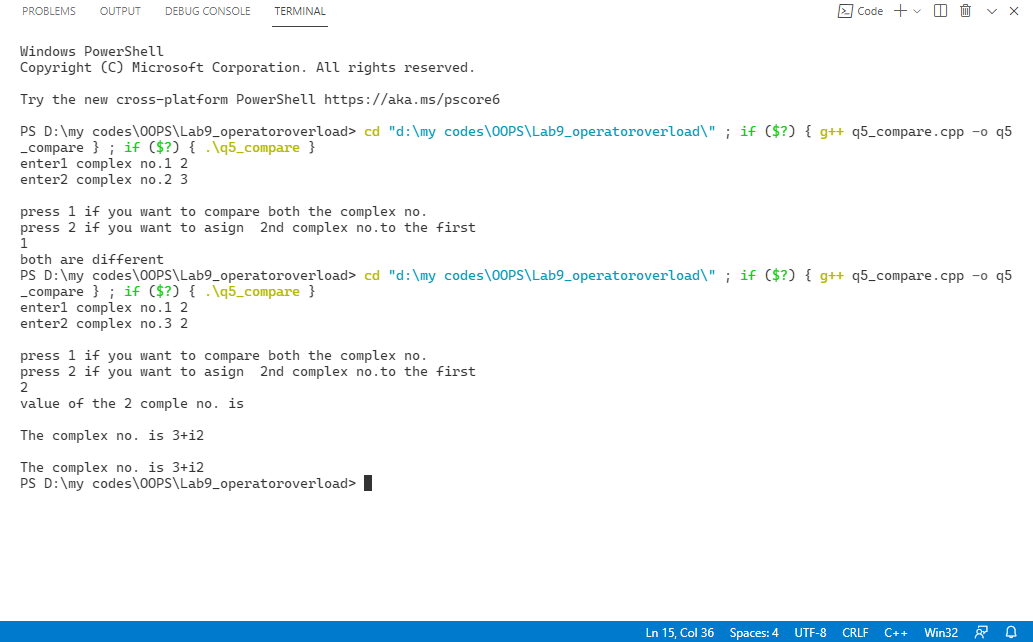
        break;

    }

    return 0;

}

OUTPUT -5



/\*Q6.WAP to add two objects of distance class. Overload the operator ‘>’ to

compare two objects and return the object with larger distance value and

display it. Overload the ‘==’ operator to compare and display whether two

given objects contain same distance value.\*/

#include <bits/stdc++.h>

using namespace std;

class distance\_025

{

    int feet\_025;

    int inches\_025;

public:

    void input()

    {

        cin >> feet\_025 >> inches\_025;

    }

    void display()

    {

        cout << feet\_025 << "'' " << inches\_025 << "'\n";

    }

    bool operator>(distance\_025);

    bool operator==(distance\_025);

};

bool distance\_025 ::operator>(distance\_025 b)

{

    float ta, tb;

    ta = feet\_025 + inches\_025 / 12.0;

    tb = b.feet\_025 + b.inches\_025 / 12.0;

    return (ta > tb)

               ? true

               : false;

}

bool distance\_025 ::operator==(distance\_025 b)

{

    float ta, tb;

    ta = feet\_025 + inches\_025 / 12.0;

    tb = b.feet\_025 + b.inches\_025 / 12.0;

    return (ta == tb)

               ? true

               : false;

}

int main()

{

    distance\_025 a[2];

    for (int i = 0; i < 2; i++)

    {

        cout << "enter" << i + 1 << " distance in feet and inches ";

        a[i].input();

    }

cout << "greatet distance is: \t";

    if (a[0] > a[1])

    {

        a[0].display();

    }

    else

        a[1].display();

    if (a[0] == a[1])

        cout << "both the distannce are equal \n ";

    else

        cout << "both are different\n";

    return 0;

}

OUTPUT -6

