// 1. Write a operator overloading code to overload all the arithmetic operators to

// add 2 complex no, 1 complex no and int value and one non member function to

// add int and complex no.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <cstdio>

#include <iostream>

#include <iomanip>

using namespace std;

struct Complex

{

    int real;

    int img;

    // setters

    void setReal(int real)

    {

        this->real = real;

    }

    void setImg(int img)

    {

        this->img = img;

    }

    // getters

    int getReal()

    {

        return this->real;

    }

    int getImg()

    {

        return this->img;

    }

    Complex()

    {

        this->real = 0;

        this->img = 0;

    }

    Complex(int real, int img)

    {

        this->real = real;

        this->img = img;

    }

    void display()

    {

        cout << this->real << "+" << this->img << "i";

    }

    // Operator Overloading

    Complex operator+(Complex c2)

    {

        Complex temp;

        temp.real = this->real + c2.real;

        temp.img = this->img + c2.img;

        return temp;

    }

    Complex operator+(int real)

    {

        Complex temp;

        temp.real = this->real + real;

        temp.img = this->img;

        return temp;

    }

    Complex operator-(Complex c1)

    {

        Complex temp;

        temp.real = this->real - c1.real;

        temp.img = this->img - c1.img;

        return temp;

    }

    Complex operator\*(Complex c2)

    {

        Complex temp;

        temp.real = this->real \* c2.real;

        temp.img = this->img \* c2.img;

        return temp;

    }

};

Complex operator+(int real, Complex c1)

{

    Complex temp;

    temp.real = c1.real + real;

    temp.img = c1.img;

    return temp;

}

int main()

{

    Complex c1, c2(20, 30), c3, c4, c5, c6, c7;

    int real, img;

    cout << "\nEnter C1 Values\nreal = ";

    cin >> real;

    cout << "\nimginary = ";

    cin >> img;

    c1.setReal(real);

    c1.setImg(img);

    cout << "\nC1 values\t";

    c1.display();

    cout << "\n\nC2 values\t";

    c2.display();

    c3 = c1 + c2; // c1.operator+(c2);

    c4 = c3 - c1; // c3.operator-(c1);

    c5 = c1 \* c2; // c1.operator\*(c2);

    cout << "\n\nc1+c2\t";

    c3.display();

    cout << "\n\nc3-c1\t";

    c4.display();

    cout << "\n\nc1\*c2\t";

    c5.display();

    c6 = c1 + 10;

    c7 = 10 + c2;

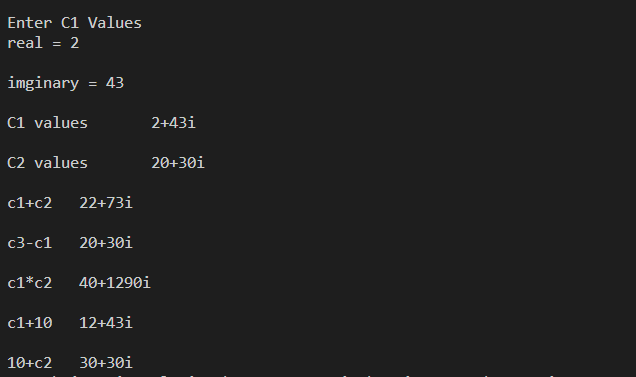
    cout << "\n\nc1+10\t";

    c6.display();

    cout << "\n\n10+c2\t";

    c7.display();

}



// 2. Write a operator overloading code to overload all the arithmetic operators to

// add 2 distances, 1 distance and int value and one non member function to add

// int and distance.

#include <stdio.h>

#include <iostream>

using namespace std;

struct Distance

{

    int inch;

    int feet;

    void setInch(int inch)

    {

        this->inch = inch;

    }

    void setFeet(int feet)

    {

        this->feet = feet;

    }

    // getter

    int getInch()

    {

        return this->inch;

    }

    int getFeet()

    {

        return this->feet;

    }

    // default

    Distance()

    {

        this->inch = 0;

        this->feet = 0;

    }

    Distance(int inch, int feet)

    {

        this->inch = inch;

        this->feet = feet;

    }

    Distance operator+(Distance d2)

    {

        Distance temp;

        temp.feet = this->feet + d2.feet;

        temp.inch = this->inch + d2.inch;

        return temp;

    }

    Distance operator+(int feet)

    {

        Distance temp;

        temp.feet = this->feet + feet;

        temp.inch = this->inch;

        return temp;

    }

    Distance operator\*(Distance d2)

    {

        Distance temp;

        temp.feet = this->feet \* d2.feet;

        temp.inch = this->inch \* d2.inch;

        return temp;

    }

    void display()

    {

        cout << "\nDistance\nFeet&Inch = " << this->feet << "'" << this->inch;

    }

};

Distance operator+(int feet, Distance d2)

{

    Distance temp;

    temp.feet = feet + d2.getFeet();

    temp.inch = d2.getInch();

    return temp;

}

int main()

{

    Distance d1, d2(20, 5), d3, d4, d5, d6;

    d1.setFeet(6);

    d1.setInch(7);

    cout<<"\nd1\n";

    d1.display();

    cout<<"\nd2\n";

    d2.display();

    d3 = d1 + d2;

    cout<<"\n\nd1+d2\n";

    d3.display();

    d4 = d1 + 10;

    cout<<"\n\nd1+10";

    d4.display();

    d5 = 20 + d2;

    cout<<"\n\n20+d2";

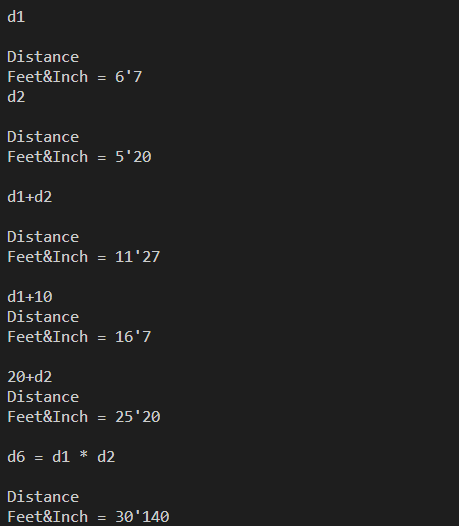
    d5.display();

    cout<<"\n\nd6 = d1 \* d2\n";

    d6 = d1 \* d2;

    d6.display();

}



// 3. Write a operator overloading code to overload logical operator for complex and distance.

#include <stdio.h>

#include <iostream>

using namespace std;

struct Complex

{

    int real;

    int img;

    Complex()

    {

        this->real = 0;

        this->img = 0;

    }

    Complex(int real, int img)

    {

        this->real = real;

        this->img = img;

    }

    void setReal(int real)

    {

        this->real = real;

    }

    void setImg(int img)

    {

        this->img = img;

    }

    int getReal()

    {

        return this->real;

    }

    int getImg()

    {

        return this->img;

    }

    void display()

    {

        cout << this->real <<"+"<< this->img << "i";

    }

    Complex operator&&(Complex c2)

    {

        Complex temp;

        temp.real = this->real && c2.real;

        temp.img = this->img && c2.img;

        return temp;

    }

    Complex operator||(Complex c2)

    {

        Complex temp;

        temp.real = this->real || c2.real;

        temp.img = this->img || c2.img;

        return temp;

    }

    int operator!()

    {

        if(this->real==0)

        {

            return 1;

        }

        else

        return 0;

    }

};

int main()

{

    Complex c1, c2(34, 9), c3, c4, c5;

    int real, img, ans;

    cout << "\nEnter C1 Values\nreal = ";

    cin >> real;

    cout << "\nimg = ";

    cin >> img;

    // Complex c1(real,img);

    c1.setReal(real);

    c1.setImg(img);

    c3 = c1 && c2;

    cout << "\nC1 Values\n";

    c1.display();

    cout << "\nC2 Values\n";

    c2.display();

    cout << "\nC1 && C2\n";

    if (c3.getReal())

    {

        cout << "\nC1.real & C2. Real both are nonZero";

    }

    else

    {

        cout << "\nC1.real or C2.real anyone of them is Zero or Both are zero";

    }

    cout << "\nC1 || C2\n";

    if (c3.getReal())

    {

        cout << "\nC1.real OR C2. Anyone of them is NonZero";

    }

    else

    {

        cout << "\nC1.real and C2.real Both are zero";

    }

    if(!c1)

    {

        cout<<"\n\n!c1.real has zero value";

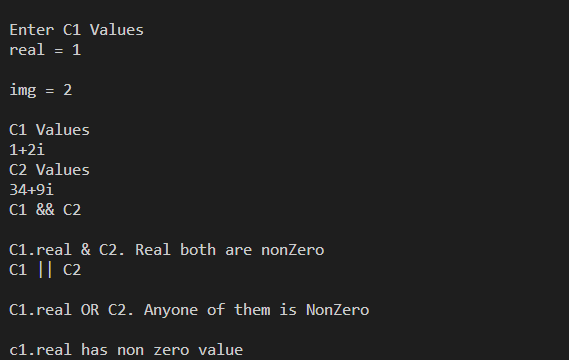
    }

    else{

        cout<<"\n\nc1.real has non zero value";

    }

}



// 3. Write a operator overloading code to overload logical operator for complex and

// distance.

#include <stdio.h>

#include <iostream>

using namespace std;

struct Distance

{

    int inch;

    int feet;

    void setInch(int inch)

    {

        this->inch = inch;

    }

    void setFeet(int feet)

    {

        this->feet = feet;

    }

    // getter

    int getInch()

    {

        return this->inch;

    }

    int getFeet()

    {

        return this->feet;

    }

    // default

    Distance()

    {

        this->inch = 0;

        this->feet = 0;

    }

    Distance(int inch, int feet)

    {

        this->inch = inch;

        this->feet = feet;

    }

    void display()

    {

        cout<<"\nFeet = "<<this->feet;

        cout<<"\nInch = "<<this->inch;

    }

    Distance operator&&(Distance d2)

    {

        Distance temp;

        temp.feet = this->feet && d2.feet;

        temp.inch = this->inch && d2.inch;

        return temp;

    }

    Distance operator||(Distance d2)

    {

        Distance temp;

        temp.feet = this->feet && d2.feet;

        temp.inch = this->inch && d2.inch;

        return temp;

    }

    int operator!()

    {

        if (this->feet == 0)

        {

            return 1;

        }

        else

        {

            return 0;

        }

    }

};

int main()

{

    Distance d1, d2(10, 20), d3, d4, d5;

    int feet, inch, ans;

    cout << "\nEnter D1 Values\nfeet = ";

    cin >> feet;

    cout << "\ninch = ";

    cin >> inch;

    d1.setFeet(feet);

    d1.setInch(inch);

    d3 = d1 && d2;

    cout << "\n\nD1 Values";

    d1.display();

    cout << "\n\nD2 Values";

    d2.display();

    cout << "\nD1 && D2\n";

    if (d3.getFeet())

    {

        cout << "\nD1.Feet & D2.Feet both are nonZero";

    }

    else

    {

        cout << "\nD1.Feet or D2.Feet anyone of them is Zero or Both are zero";

    }

    cout << "\nD1 || D2\n";

    d4 = d1 || d2;

    if (d4.getFeet())

    {

        cout << "\nD1.feet or D2.feet Anyone of them is NonZero or both are non zero";

    }

    else

    {

        cout << "\nD1.feet and D2.feet Both are zero";

    }

    if (!d1)

    {

        cout << "\n\n!D1.Feet has zero value";

    }

    else

    {

        cout << "\n\nD1.Feet has non zero value";

    }

}

