TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

DHARAN

LABSHEET 5 : [FRIEND FUCNTION / CLASS AND OPERATOR OVERLOADING]

BY

Rabin Poudel [PUR079BCT056]

DEPARTMENT OF COMPUTER & ELECTRONICS ENGINEERING

PURWANCHAL CAMPUS

DHARAN, NEPAL

JULY, 2024

0.1 Question 1 :

Write a class for instantiating the objects that represent the two-dimensional Cartesian coordinate system.

A. Make a particular member function of one class as a friend function of another class for addition.

B. Make the other three functions to work as a bridge between the classes for multiplication, division, and subtraction.

C. Also write a small program to demonstrate that all the member functions of one class are the friend functions of another class if the former class is made friend to the latter.

Make least possible classes to demonstrate all the above in a single program without conflict.

Code:

```
#include <iostream>
#include <cmath>

using namespace std;
class Coordinate;
class Vector;
class Polar
{
private:
    float radius, theta;

public:
    Polar(float r, float angle) : radius(r), theta(angle) {};
    friend class Vector;
    Coordinate toCartesian();
};
class Vector
{
public:
    Coordinate add(Coordinate a, Coordinate b);
```

```
        Coordinate sub(Coordinate a, Coordinate b);
        Coordinate mul(Coordinate a, Coordinate b);
        Coordinate div(Coordinate a, Coordinate b);
};

class Coordinate
{
private:
        float x, y;

public:
        Coordinate(float a, float b) : x(a), y(b) {};
        friend Coordinate Vector::add(Coordinate a, Coordinate b);
        friend Coordinate Vector::sub(Coordinate a, Coordinate b);
        Polar toPolar()
        {
                Polar temp(sqrt(x      *  x + y   *  y), atanf(y / x));
                return temp;
        }
        void display()
        {
                cout << "(" << x << "," << y << ")";
        }
};

Coordinate Polar::toCartesian()
{
        Coordinate C(radius      *  cos(theta), radius       *  sin(theta));
        return C;
}
Coordinate Vector::add(Coordinate a, Coordinate b)
{
        Coordinate temp(a.x + b.x, a.y + b.y);
        return temp;
}
Coordinate Vector::sub(Coordinate a, Coordinate b)
{
        Coordinate temp(a.x − b.x, a.y − b.y);
        return temp;
```

```
}
Coordinate Vector::mul(Coordinate a, Coordinate b)
{
	Polar pa = a.toPolar();
	Polar pb = b.toPolar();
	Polar p(pa.radius	*	pb.radius, pa.theta + pb.theta);
	return p.toCartesian();
}
Coordinate Vector::div(Coordinate a, Coordinate b)
{
	Polar pa = a.toPolar();
	Polar pb = b.toPolar();
	Polar p(pa.radius / pb.radius, pa.theta − pb.theta);
	return p.toCartesian();
}

int main()
{
	int x, y;
	char temp;
	cout << "Enter cordinate x y in format x,y";
	cin >> x >> temp >> y;
	Coordinate a(x, y);
	cout << "Enter cordinate x y in format x,y";
	cin >> x >> temp >> y;
	Coordinate b(x, y);
	Vector v;
	Coordinate c = v.add(a, b);
	cout << "The sum is ";
	c.display();
	cout << endl;
	Coordinate d = v.sub(a, b);
	cout << "The difference is";
	d.display();
	cout << endl;
	Coordinate p = v.mul(a, b);
	cout << "The product is";
	p.display();
	cout << endl;
```

```
    Coordinate q = v.div(a, b);
    cout << "The quotient is";
    q.display();
    cout << endl;
    return 0;
}
```

0.2 Question 2:

Write a class to store x, y, and z coordinates of a point in three-dimensional space. Overload addition and subtraction operators for addition and subtraction of two coordinate objects. Implement the operator functions as non-member functions (friend operator functions).

Code:

```
#include <iostream>

using namespace std;

class Coordinate
{
private:
    float x, y, z;

public:
    Coordinate(float a, float b, float c) : x(a), y(b), z(c) {};
    friend Coordinate operator+(Coordinate a, Coordinate b);
    friend Coordinate operator−(Coordinate a, Coordinate b);
    void display()
    {
        cout << "(" << x << "," << y << "," << z << ")";
    }
};
Coordinate operator+(Coordinate a, Coordinate b)
{
    Coordinate temp(a.x + b.x, a.y + b.y, a.z + b.z);
    return temp;
}
Coordinate operator−(Coordinate a, Coordinate b)
{
    Coordinate temp(a.x − b.x, a.y − b.y, a.z − b.z);
    return temp;
}
int main()
{
```

```
    Coordinate a(1, 2, 3), b(4, 5, 6);
    Coordinate c = a + b;
    Coordinate d = a − b;
    cout << "a: ";
    a.display();
    cout << endl;
    cout << "b: ";
    b.display();
    cout << endl;
    cout << "a + b: ";
    c.display();
    cout << endl;
    cout << "a − b: ";
    d.display();
    cout << endl;
    return 0;
}
```

0.3 Question 3:

Write a program to compare two objects of a class that contains an integer value as its data member. Make overloading functions to overload equality(==), less than(<), greater than(>), not equal (!=), greater than or equal to (>=), and less than or equal to(<=) operators using member operator functions.

Code:

```
#include <iostream>
using namespace std;

class Integer
{
private:
    int value;

public:
    Integer(int val) : value(val) {}

    // Getter for the value (so we can access it in main)
    int getValue() const { return value; }

    // Comparison operators
    bool operator==(const Integer &obj) const
    {
        return value == obj.value;
    }

    bool operator<(const Integer &obj) const
    {
        return value < obj.value;
    }

    bool operator>(const Integer &obj) const
    {
        return value > obj.value;
    }
```

```
    bool operator!=(const Integer &obj) const
    {
        return value != obj.value;
    }

    bool operator>=(const Integer &obj) const
    {
        return value >= obj.value;
    }

    bool operator<=(const Integer &obj) const
    {
        return value <= obj.value;
    }
};

int main()
{
    Integer a(5), b(5), c(10);

    cout << "a: " << a.getValue() << endl;
    cout << "b: " << b.getValue() << endl;
    cout << "c: " << c.getValue() << endl;

    cout << "a == b: " << (a == b) << endl;
    cout << "a < b: " << (a < b) << endl;
    cout << "a > b: " << (a > b) << endl;
    cout << "a != b: " << (a != b) << endl;
    cout << "a >= b: " << (a >= b) << endl;
    cout << "a <= b: " << (a <= b) << endl;

    cout << "a == c: " << (a == c) << endl;
    cout << "a < c: " << (a < c) << endl;
    cout << "a > c: " << (a > c) << endl;
    cout << "a != c: " << (a != c) << endl;
    cout << "a >= c: " << (a >= c) << endl;
    cout << "a <= c: " << (a <= c) << endl;
```

```
    return 0;
}
```

0.4 Question 4:

Write a class Date that overloads prefix and postfix operators to increase the Date object by one day, while causing appropriate increments to the month and year (use the appropriate condition for leap year). The prefix and postfix operators in the Date class should behave exactly like the built-in increment operators.

Code:

```
#include <iostream>
using namespace std;
class date
{
    int y, m, d;

public:
    void get_data()
    {
        cout << "Enter valid date.";
        cout << endl
             << "Enter year: ";
        cin >> y;
        cout << endl
             << "Enter month: ";
        cin >> m;
        cout << endl
             << "Enter day: ";
        cin >> d;
    }
    void operator++(int)
    {
        cout << y << ":" << m << ":" << d++ << endl;
    }
    void operator++()
    {
        ++d;
        if (((y % 4 == 0) && (y % 100 == 0) && (y % 400 == 0)) ||
            ((y % 4 == 0) && (y % 100 != 0)))
        {
```

```
            if ((m / 2 == 1) && (29 < d))
            {
                m++;
                d = d - 29;
            }
        }
        else
        {
            if ((m / 2 == 1) && (28 < d))
            {
                m++;
                d = d - 28;
            }
        }
        if ((m % 2 == 1) && (31 < d))
        {
            m++;
            d = d - 31;
        }
        if ((m % 2 == 0) && (m / 2 != 1) && (30 < d))
        {
            m++;
            d = d - 30;
        }
        if (12 < m)
        {
            m = 1;
            y++;
        }
        cout << y << ":" << m << ":" << d << endl;
    }
};
int main()
{
    date yyyy;
    yyyy.get_data();
    cout << endl
         << "Prefix Operator Overloaded." << endl;
    yyyy++;
```

```
        cout << endl
                << "Postfix Operator Overloaded." << endl;
        ++yyyy;
        return 0;
}
```