

UEE 1303(1067): Object-Oriented Programming

Lab #7: Operator Overloading

In this laboratory session you will:

- Review the operator overloading in C++

Lab 7-1: Operator Overloading

- ✓ Operator can be overloaded to define the operator on the object.

```
// lab7-1.cpp
#include <iostream>

class Point2D
{
public:
    Point2D();
    Point2D(int n1, int n2);
    Point2D(int n1, int n2, double v);
    Point2D operator + (const Point2D &);
    Point2D operator - ();
    void assignPoint2D(int n1, int n2);
    void assignPoint2D(int n1, int n2, double v);
    void displayPoint2D() const;
    friend double distPoint2D(const Point2D &, const Point2D &);
    friend double distPoint2D(const Point2D &, const Point2D &, const
Point2D &);
    friend bool operator == (const Point2D &, const Point2D &);
    friend bool operator != (const Point2D &, const Point2D &);
private:
    int x;
    int y;
    double value;
};

Point2D Point2D::operator + (const Point2D &pt)
{
    return Point2D(x + pt.x, y + pt.y, value + pt.value);
}
```

```
Point2D Point2D::operator - ()
{
    return Point2D(-x, -y, -value);
}

bool operator == (const Point2D &pt1, const Point2D &pt2)
{
    if (pt1.x != pt2.x || pt1.y != pt2.y || pt1.value != pt2.value)
        return false;
    return true;
}

bool operator != (const Point2D &pt1, const Point2D &pt2)
{
    return !(pt1 == pt2);
}

double distPoint2D(const Point2D &pt1, const Point2D &pt2)
{
    return sqrt((pt1.x - pt2.x)*(pt1.x - pt2.x) + (pt1.y -
pt2.y)*(pt1.y - pt2.y));
}

double distPoint2D(const Point2D &pt1, const Point2D &pt2, const
Point2D &pt3)
{
    double n1 = distPoint2D(pt1, pt2);
    double n2 = distPoint2D(pt1, pt3);
    double n3 = distPoint2D(pt2, pt3);
    return (n1 + n2 + n3);
}

Point2D::Point2D()
{
    x = 0;
    y = 0;
    value = 0;
}

Point2D::Point2D(int n1, int n2)
{
    assignPoint2D(n1, n2, 0.0);
}
```

```
Point2D::Point2D(int n1, int n2, double v)
{
    assignPoint2D(n1, n2, v);
}
void Point2D::assignPoint2D(int n1, int n2)
{
    assignPoint2D(n1, n2, value);
}
void Point2D::assignPoint2D(int n1, int n2, double v)
{
    x = n1;
    y = n2;
    value = v;
}
void Point2D::displayPoint2D() const
{
    std::cout << "(" << x << "," << y << ") = ";
    std::cout << value << std::endl;
}
int main()
{
    Point2D pt1(3,4,4.1);
    Point2D pt2(3,2,4.5);
    if (pt1 == pt2)
        std::cout << "pt1 is equal to pt2 " << std::endl;
    else
        std::cout << "pt1 is not equal to pt2 " << std::endl;
    pt1.displayPoint2D();
    pt2.displayPoint2D();
    pt2.displayPoint2D();
    Point2D pt3;
    pt3 = pt1 + pt2;
    pt3.displayPoint2D();
    Point2D pt4 = -pt1;
    pt4.displayPoint2D();
    return 0;
}
```

Exercise 7-1

- ✓ Please modify the class Complex you defined in ex5-1 which make the file ex7-1 work.

```
// ex7-1.cpp
#include <iostream>
using std::cout;
using std::endl;
#include "Complex.h"
int main()
{
    Complex a(1.0, 7.0), b(9.0, 2.0), c; // create three Complex
objects
    printMeg(a, b, '+'); // output (1.0, 7.0) + (9.0, 2.0) =
    c = a + b; // invoke operator + and assign to object c
    cout << c << endl; // output object c

    printMeg(a, b, '-'); // output (1.0, 7.0) - (9.0, 2.0) =
    c = a - b; // invoke operator - function and assign to object c
    cout << c << endl; // output object c

    printMeg(a, b, '*'); // output (1.0, 7.0) * (9.0, 2.0) =
    c = a * b; // invoke operator * function and assign to object c
    cout << c << endl; // output object c

    printMeg(a, b, '/'); // output (1.0, 7.0) / (9.0, 2.0) =
    c = a / b; // invoke operator / function and assign to object c
    cout << c << endl; // output object c

    a.setComplexNumber(10.0, 1.0); // reset object a
    b = -a;
    printMeg(a, b, '-');
    c = a - b; // invoke operator - function and assign to object c
    cout << c << endl; // output object c

    return 0;
}
```