Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа №3

Вариант 5

Перегрузка операторов

Выполнил студент группы № M3111
Соловьев Михаил Александрович.

Санкт-Петербург
2023

В данном задании необходимо согласно варианту, описать указанные типы данных и поместить их в отдельный заголовочный файл, в нем же описать операторы. Реализацию функций поместить в отдельный cpp файл.
Типы данных:
- Треугольник на плоскости
- Подмножество множества целых чисел от 0 до 9

Код:

class.cpp

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include "class.h"

Triangle::Triangle(std::vector<Point> points) {
    this->points = points;
}

Point Triangle::getVertex1() const { return points[0]; }
Point Triangle::getVertex2() const { return points[1]; }
Point Triangle::getVertex3() const { return points[2]; }

double Triangle::area() const {
    double a = distance(points[0], points[1]);
    double b = distance(points[1], points[2]);
    double c = distance(points[2], points[0]);
    double p = (a + b + c) / 2.0;

    return std::sqrt(p * (p - a) * (p - b) * (p - c));
}

double Triangle::distance(const Point &point1, const Point &point2) {
    double dx = point2.x - point1.x;
    double dy = point2.y - point1.y;
    return std::sqrt(dx * dx + dy * dy);
}

void Triangle::move(std::vector<double> &vector) {
    for (int i = 0; i < 3; i++) {
        points[i].x += vector[0];
        points[i].y += vector[1];
    }
}

bool Triangle::operator==(const Triangle &other) const {
    if (area() == other.area()) { return true; }
    return false;
}
```

```
bool Triangle::operator!=(const Triangle &other) const {
    if (area() != other.area()) { return true; }
    return false;
}

bool Triangle::operator>(const Triangle &other) const {
    if (area() > other.area()) { return true; }
    return false;
}

bool Triangle::operator<(const Triangle &other) const {
    if (area() < other.area()) { return true; }
    return false;
}

void Triangle::operator+(std::vector<double> &vector) {
    move(vector);
}

/* SUBSET */
Subset::Subset() {
    std::fill(std::begin(elements), std::end(elements), false);
}

Subset::Subset(const std::vector<int> &elements_add) {
    for (int i = 0; i < 10; i++) {
        elements[i] = false;
    }

    for (auto element : elements_add) {
        add(element);
    }

}

void Subset::add(int element) {
    elements[element] = true;
}

void Subset::remove(int element) {
    elements[element] = false;
}

Subset Subset::operator+(const Subset &other) const {
    Subset result;
    for (int i = 0; i < 10; i++) {
        result.elements[i] = elements[i] || other.elements[i];
    }
```

```
    return result;
}

bool Subset::operator==(const Subset &other) const {
    for (int i = 0; i < 10; i++) {
        if (elements[i] != other.elements[i]) {
            return false;
        }
    }
    return true;
}

bool Subset::operator!=(const Subset &other) const {
    for (int i = 0; i < 10; i++) {
        if (elements[i] != other.elements[i]) {
            return true;
        }
    }
    return false;
}

void Subset::operator+=(int element) {
    add(element);
}

void Subset::operator-=(int element) {
    remove(element);
}

std::vector <int> Subset::To_vector() const {
    std::vector <int> result;
    for (int i = 0; i < 10; i++) {
        if (elements[i]) {
            result.push_back(i);
        }
    }
    return result;
}
```

class.h

```
#ifndef COURSE_C___CLASS_H
#define COURSE_C___CLASS_H

#include <iostream>
#include <vector>

struct Point {
    double x;
```

```
    double y;
};

/* TRIANGLE */
class Triangle {
private:
    std::vector <Point> points; // points representing the triangle vertices

public:
    Triangle(std::vector <Point> points);

    Point getVertex1() const;
    Point getVertex2() const;
    Point getVertex3() const;

    double area() const;
    void move(std::vector <double> &vector);

    bool operator==(const Triangle &other) const;
    bool operator!=(const Triangle &other) const;
    bool operator<(const Triangle &other) const;
    bool operator>(const Triangle &other) const;
    void operator+(std::vector <double> &vector);

private:
    static double distance(const Point &point1, const Point &point2) ;
};

/* SUBSET */
class Subset {
private:
    bool elements[10];

public:
    Subset();
    Subset(const std::vector <int> &elements);

    void add(int element);
    void remove(int element);

    Subset operator+(const Subset &other) const;

    bool operator==(const Subset &other) const;
    bool operator!=(const Subset &other) const;

    void operator+=(int element);
    void operator-=(int element);

    std::vector <int> To_vector() const;
```

```
};

#endif //COURSE_C___CLASS_H
```

main.cpp

```
#include <iostream>
#include "class.h"

int main() {
    Point a = {0.0, 4.0};
    Point b = {-2.0, 0.0};
    Point c = {2.0, 0.0};
    Point a1 = {2.0, 3.0};
    std::vector <Point> vertices {a, b, c};
    std::vector <Point> vertices1 {a1, b, c};
    std::vector <double> to_move {1.0, 0.0};

    Triangle triangle(vertices);
    Triangle triangle1(vertices1);

    Subset subset({1, 2, 3});
    Subset subset1({4, 5, 6});

    std::cout << "Diff " << (triangle != triangle1) << "\n";
    std::cout << "Same  " << (triangle == triangle) << "\n";
    std::cout << "More  " << (triangle > triangle1) << "\n";
    std::cout << "Less  " << (triangle1 < triangle) << "\n";

    triangle + to_move;

    std::cout << "New: \n";
    std::cout << triangle.getVertex1().x << " " << triangle.getVertex1().y << "\n";
    std::cout << triangle.getVertex2().x << " " << triangle.getVertex2().y << "\n";
    std::cout << triangle.getVertex3().x << " " << triangle.getVertex3().y << "\n";

    std::cout << "Subsets: \n";
    for (auto i : (subset + subset1).To_vector()) {
        std::cout << i << " ";
    }
    std::cout << "\n";
    std::cout << "Same " << (subset == subset) << "\n";
    std::cout << "Diff " << (subset != subset1) << "\n";
    std::cout << "First add: 4 \n";
    subset += 4;
    for (auto i : subset.To_vector()) {
        std::cout << i << " ";
    }
    std::cout << "\n";
```

```
    std::cout << "First remove: 2 \n";
    subset -= 2;
    for (auto i : subset.To_vector()) {
        std::cout << i << " ";
    }
    std::cout << "\n";
}
```