Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

Программирование на C++

Лабораторная работа №3, Перегрузка операторов.

Вариант 5

Выполнил: Гаджиев Саид M3115

Санкт-Петербург

2023 г.

Лабораторная работа №3. Перегрузка операторов.
Согласно варианту описать указанные типы данных и поместить их в отдельный заголовочный файл, в нем же описать операторы, указанные в варианте. Реализацию функций поместить с отдельный cpp файл.
Написать программу, проверяющую правильность работы – для наглядности и лучшего
усвоения материала использовать как явный, так и не явный метод вызова функций операторов (см. пример в конце задания).

Вариант 5

Тип данных:
1) Треугольник на плоскости.

2) Подмножество множества целых чисел от нуля до девяти: { 0, 1, 2, … 9 }.

Операции:
1) Равенство площадей треугольников (перегрузите операции ==, !=, <, >) – для вычисления площади можете использовать, например, формулу Герона (зависит от тех данных, что используются для хранения треугольника). Прибавление вектора (смещение треугольника на указанный вектор).

2) Объединение двух множеств (operator+). Сравнение (== и !=). Добавление числа в множество (operator+=). Изъятие числа из множества (operator-=).

Решение:

main.cpp

```
#include <iostream>
#include "triangle.h"
#include "subset.h"

int main() {
    // Пример использования структуры Triangle
    std::cout << "TRIANGLE:" << "\n";
    Point a = {0, 0};
    Point b = {0, 4};
    Point c = {3, 0};

    Triangle t1(a, b, c);
    std::cout << "Area of t1: " << t1.getArea() << "\n";
    Triangle t2 = t1 + Point{ 1, 1 };
    std::cout << "Area of t2: " << t2.getArea() << "\n";

    if (t1 == t2) {
        std::cout << "t1 = t2" << "\n";
    } else if (t1 != t2){
        if (t1 > t2) {
            std::cout << "t1 > t2" << "\n";
        } else if (t1 < t2) {
            std::cout << "t1 < t2" << "\n";
        }
    }
```

```
    // Пример использования класса Subset
    std::cout << "\nSUBSET:" << "\n";
    Subset s1, s2;

    // заполнение множеств
    for (int i = 0; i < 10; i++) {
        s1 += i;
        s2 += i * 2;
    }

    // вывод множеств
    std::cout << "subset1: ";
    s1.Println();

    std::cout << "subset2: ";
    s2.Println();

    // объединение множеств
    Subset s3 = s1 + s2;
    std::cout << "subset1 + subset2: ";
    s3.Println();

    // сравнение множеств
    if (s1 == s2) {
        std::cout << "subset1 == subset2" << "\n";
    } else {
        std::cout << "subset1 != subset2" << "\n";
    }

    // добавление элемента в множество
    s1 += 10;
    std::cout << "subset1 after adding 10: ";
    s1.Println();

    // удаление элемента из множества
    s2 -= 4;
    std::cout << "subset2 after removal 4: ";
    s2.Println();
    return 0;
}
```

Первая часть программы демонстрирует использование структуры Triangle. Создаются три точки на плоскости и с их помощью создаются два треугольника: t1 и t2. Выводятся площади этих треугольников, затем они сравниваются между собой, используя операторы сравнения.

Вторая часть программы демонстрирует использование класса Subset. Создаются два множества: s1 и s2, заполняются элементами и выводятся на экран. Затем производится объединение множеств в третье множество s3, выводятся элементы s3. Далее производится сравнение множеств и добавление/удаление элементов из множеств s1 и s2. В конце программы возвращается ноль.

triangle.h

```
//
// Created by Redmibook on 14.03.2023.
//
#ifndef LAB_3_TRIANGLE_H
#define LAB_3_TRIANGLE_H

#include <iostream>

struct Point {
    double x, y;
};

class Triangle {
    private:
        Point m_a;
        Point m_b;
        Point m_c;
    public:
        Triangle(Point a, Point b, Point c);
        Triangle(Triangle &other);

        double getArea() const;
        Triangle &operator=(const Triangle &other);
        Triangle operator+(const Point &vec);
        bool operator==(const Triangle &other);
        bool operator!=(const Triangle &other);
        bool operator<(const Triangle &other);
        bool operator>(const Triangle &other);
};

#endif //LAB_3_TRIANGLE_H
```

Здесь определяется класс Triangle и структура Point. В классе Triangle определены члены m_a, m_b, m_c, представляющие вершины треугольника. Они инициализируются в конструкторе класса, который принимает три параметра типа Point.

В классе Triangle определены различные операторы, такие как operator+, operator==, operator!=, operator< и operator>. Они позволяют выполнить арифметические и логические операции между треугольниками и точками.

Также в классе Triangle определена функция getArea(), которая возвращает площадь треугольника.

Класс Triangle определен с помощью ключевого слова class, а структура Point с помощью ключевого слова struct. Эти два ключевых слова похожи, но имеют различия в области видимости по умолчанию и доступа к членам структуры или класса.

triangle.cpp

```cpp
//
// Created by Redmibook on 14.03.2023.
//
#include <cmath>
#include "triangle.h"

Triangle::Triangle(Point a, Point b, Point c) {
    m_a = a;
    m_b = b;
    m_c = c;
}

Triangle::Triangle(Triangle &other) {
    m_a = other.m_a;
    m_b = other.m_b;
    m_c = other.m_c;
}

double Triangle::getArea() const {
    double a = std::sqrt(std::pow(m_b.x - m_a.x, 2) + std::pow(m_b.y - m_a.y,
2));
    double b = std::sqrt(std::pow(m_c.x - m_b.x, 2) + std::pow(m_c.y - m_b.y,
2));
    double c = std::sqrt(std::pow(m_a.x - m_c.x, 2) + std::pow(m_a.y - m_c.y,
2));
    double p = (a + b + c) / 2;
    double area = std::sqrt(p * (p - a) * (p - b) * (p - c));
    return area;
}

Triangle& Triangle::operator=(const Triangle &other) {
    if (this != &other) {
        m_a = other.m_a;
        m_b = other.m_b;
        m_c = other.m_c;
    }
    return *this;
}

Triangle Triangle::operator+(const Point &vec) {
    return Triangle({m_a.x + vec.x, m_a.y + vec.y}, {m_b.x + vec.x, m_b.y +
vec.y}, {m_c.x + vec.x, m_c.y + vec.y});
}

bool Triangle::operator==(const Triangle &other) {
    return getArea() == other.getArea();
}

bool Triangle::operator!=(const Triangle &other) {
    return getArea() != other.getArea();
}

bool Triangle::operator<(const Triangle &other) {
    return getArea() < other.getArea();
}

bool Triangle::operator>(const Triangle &other) {
    return getArea() > other.getArea();
}
```

В данном коде определены методы класса Triangle. В конструкторе Triangle(Point a, Point b, Point c) инициализируются координаты вершин треугольника m_a, m_b, m_c. Метод getArea() вычисляет площадь треугольника по формуле Герона. Оператор присваивания operator= копирует значения координат вершин из объекта other в текущий объект. Оператор сложения operator+ возвращает новый объект типа Triangle, координаты вершин которого сдвинуты на вектор vec. Операторы сравнения operator==, operator!=, operator< и operator> сравнивают площади двух треугольников.

subset.h

```
//
// Created by Redmibook on 14.03.2023.
//
#ifndef LAB_3_SUBSET_H
#define LAB_3_SUBSET_H

#include <set>

class Subset {
    private:
        std::set<int> subset;
    public:
        Subset operator+(Subset &other) ; // объединение множеств
        bool operator==(const Subset &other); // сравнение множеств
        bool operator!=(const Subset &other); // сравнение множеств
        void operator+=(int elem); // добавление элемента в множество
        void operator-=(int elem); // удаление элемента из множества
        void Println();
};

#endif //LAB_3_SUBSET_H
```

Этот код содержит объявление класса Subset, который представляет собой множество целых чисел. Он имеет закрытое поле subset, которое является стандартным множеством целых чисел из стандартной библиотеки C++. Класс Subset определяет несколько перегруженных операторов, таких как операторы +, ==, !=, += и -=.

Оператор + выполняет объединение двух множеств, операторы == и != сравнивают два множества на равенство и неравенство соответственно. Оператор += добавляет элемент в множество, а оператор -= удаляет элемент из множества.

Также в классе есть функция Println(), которая выводит содержимое множества на консоль. В целом, класс Subset может быть использован для работы с множествами целых чисел в программе.

subset.cpp

```cpp
//
// Created by Redmibook on 14.03.2023.
//
#include <iostream>
#include <set>
#include "subset.h"

Subset Subset::operator+(Subset &other) { // объединение множеств
    Subset result;
    result.subset = this->subset;
    result.subset.insert(other.subset.begin(), other.subset.end());
    return result;
}

bool Subset::operator==(const Subset &other) { // сравнение множеств
    return this->subset == other.subset;
}

bool Subset::operator!=(const Subset &other) { // сравнение множеств
    return !(*this == other);
}

void Subset::operator+=(int elem) { // добавление элемента в множество
    this->subset.insert(elem);
}

void Subset::operator-=(int elem) { // удаление элемента из множества
    this->subset.erase(elem);
}

void Subset::Println() {
    for (int elem : subset) {
        std::cout << elem << " ";
    }
    std::cout << "\n";
}
```

Здесь реализуется класс Subset, который содержит множество целых чисел типа std::set<int>. В данном файле определены методы класса, которые выполняют следующие действия:

Оператор +, который принимает в качестве аргумента ссылку на объект Subset, и возвращает новый объект Subset, который представляет объединение множеств текущего объекта и переданного объекта. Для этого создается новый объект Subset, который инициализируется текущим множеством, а затем в него добавляются элементы из переданного множества с помощью метода std::set::insert().

Операторы == и !=, которые сравнивают текущее множество с переданным множеством на равенство и неравенство соответственно.

Операторы += и -= для добавления и удаления элементов из множества. Они используют методы std::set::insert() и std::set::erase() соответственно.

Метод Println(), который выводит все элементы множества на экран, разделенные пробелом, и завершается символом новой строки.