МИНИСТЕРСТВО НАУКИ и высшего образования

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**Отчёт по лабораторной работе № 4**

**по учебной дисциплине «Системное программирование»**

**Тема: «Перегрузка операций»**

Выполнил студент

специальности 09.02.07

Информационные системы и

программирование

IV курса группы 42919/4

Марков Данил

Петрович

Преподаватель

Молькова Лолита Юрьевна

Санкт-Петербург

2023

# **Цели работы:**

Изучить механизм перегрузки операций.

# **Задание:**

Все классы следует наделить конструкторами, деструктором. Необходимо явно реализовать конструктор копирования и перегрузить оператор присваивания. Необходимо подготовить демонстрацию по работе перегруженных для класса операторов

Для пространства R 3 (выбрана правая система декартовых прямоугольных координат {0, i, j, k}) создать класс ВЕКТОР, предусмотрев для него несколько видов конструкторов, метод для вывода на экран его координат. Для данного класса перегрузить следующие бинарные операторы: суммы(+), разности (-),“векторное произведение” (\*).

# **Ход работы**:

Инициализация класса и создание объектов:

Были созданы 2 объекта класса RVector с помощью конструктора.



Рисунок 1 – Создание объектов

Вывод информации о секторах:

Используем метод printInfo для вывода информации о каждом секторе.

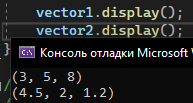


Рисунок 2 – Вывод информации с помощью метода display()

Скопируем vector2 в copied с помощью конструктора копирования.

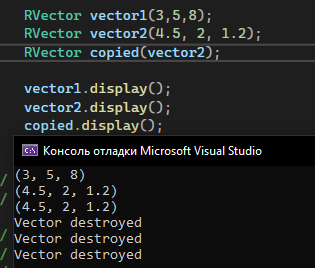


Рисунок 3 – Использование конструктора копирования

Далее на рисунке 4. Продемонстрирована работа операторов сложения, вычитания и умножения соответственно.

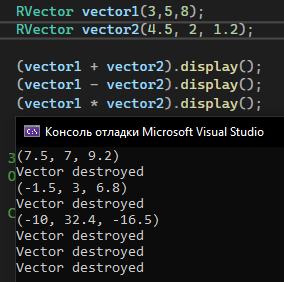


Рисунок 4 – Операции сложения, вычитания и умножения

Код файла RVector.h:

#include <iostream>

class RVector {

private:

double x, y, z;

public:

RVector();

RVector(double x, double y, double z);

~RVector();

RVector(const RVector& other);

void display() const;

RVector operator+(const RVector& other) const;

RVector operator-(const RVector& other) const;

RVector operator\*(const RVector& other) const;

}

Код файла RVector.cpp:

#include "RVector.h"

RVector::RVector() : x(0), y(0), z(0) {}

RVector::RVector(double x, double y, double z) : x(x), y(y), z(z) {}

RVector::~RVector() {

std::cout << "Vector destroyed\n";

}

RVector::RVector(const RVector& other) {

x = other.x;

y = other.y;

z = other.z;

}

void RVector::display() const {

std::cout << "(" << x << ", " << y << ", " << z << ")" << std::endl;

}

RVector RVector::operator+(const RVector& other) const {

return RVector(x + other.x, y + other.y, z + other.z);

}

RVector RVector::operator-(const RVector& other) const {

return RVector(x - other.x, y - other.y, z - other.z);

}

RVector RVector::operator\*(const RVector& other) const {

return RVector(y \* other.z - z \* other.y, z \* other.x - x \* other.z, x \* other.y - y \* other.x);

}

Код файла Main.cpp:

#include <iostream>

#include "RVector.h"

int main()

{

RVector vector1(3,5,8);

RVector vector2(4.5, 2, 1.2);

(vector1 + vector2).display();

(vector1 - vector2).display();

(vector1 \* vector2).display();

}

# **Контрольные вопросы**:

1. Для чего в С++ применяется перегрузка операций:

Перегрузка операций в C++ используется для предоставления пользовательских определений операций для пользовательских типов данных.

1. Истинно ли следующее утверждение: операция >= может быть перегружена?

Операция >= может быть перегружена для пользовательских типов данных.

1. Сколько аргументов требуется для определения перегруженной унарной операции?

Для унарной операции, такой как унарный минус - или унарный плюс +, требуется только один аргумент.

1. Сколько аргументов требуется для определения перегруженной бинарной операции?

Для бинарной операции, такой как сложение + или умножение \*, требуется два аргумента.

1. Чем отличается действие операции ++ в префиксной форме от её действия в постфиксной форме?

В префиксной форме ++ увеличивает значение переменной и возвращает новое значение, тогда как в постфиксной форме ++ возвращает текущее значение переменной и затем увеличивает её.

1. Истинно ли следующее утверждение: перегруженная операция всегда требует на один аргумент меньше, чем количество операндов?

Нет, это утверждение не всегда верно. Например, унарные операции требуют один аргумент, но они имеют только один операнд.

1. Когда перегружается операция арифметического присваивания, то результат:

b. Передается объекту слева от операции.

1. Истинно ли следующее утверждение: компилятор не выдаст сообщение об ошибке, если вы перегрузите операцию \* для выполнения деления?

Да, это истинно. Компилятор не обязательно выдаст ошибку при перегрузке операции \* для выполнения деления

1. Существуют ли операции, которые нельзя перегружать?

Да, есть несколько операций, которые нельзя перегружать, такие как операторы точки . и разрешения области ::.