МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

# ЛАБОРАТОРНАЯ РАБОТА №2 по курсу

объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Медведев Данила Андреевич, группа М80-208Б-20*
Преподаватель *Дорохов Евгений Павлович*

**Цель:**

- Изучение основ работы с классами в C++;
- Перегрузка операций и создание литералов

**Требования к программе(Вариант11)**

Создать класс vector3D, задаваемый тройкой координат. Обязательно должны быть реализованы: операции сложения и вычитания векторов, векторное произведение векторов, скалярное произведение векторов, умножения на скаляр, сравнение векторов на совпадение, вычисление длины вектора, сравнение длины векторов, вычисление угла между векторами.

Реализовать над объектами реализовать в виде перегрузки операторов.

Реализовать пользовательский литерал для работы с константами объектов созданного класса.

## Дневник отладки

## Недочёты

## Выводы

Изучил основы работы с классами.

## Исходный код

main.cpp

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

class Vector{

public:
```

```
    Vector(): vx(0), vy(0), vz(0), vlength(0) {}

    Vector(double x, double y, double z): vx(x), vy(y), vz(z) {
        vlength = sqrt(x * x + y * y + z * z);
    }

    double x()      const { return vx; }
    double y()      const { return vy; }
    double z()      const { return vz; }
    double length() const { return vlength; }

    bool is_match(const Vector& v) const{
        return vx == v.x() && vy == v.y() && vz == v.z();
    }

    bool is_equal(const Vector& v) const{
        return vlength == v.length();
    }

    friend std::istream& operator>> (std::istream& in, Vector& v);

private:

    double vx, vy, vz;
    double vlength;

};

bool is_number(const std::string& s){
    bool point = false;
    for (int i = 0; i < s.length(); ++i){
        if (s[i] == '-' && i == 0){
            continue;
        } else if (s[i] == '.'){
            if ((i == 0 || i == s.length() - 1) || point){
                return false;
            } else {
                point = true;
            }
        } else if (s[i] < '0' || s[i] > '9'){ return false; }
    }
    return true;
}

std::istream& operator>> (std::istream& in, Vector& v){

    std::string x, y, z; std::cin >> x >> y >> z;

    if (!is_number(x) || !is_number(y) || !is_number(z)){
        std::cout << "Некорректный ввод.\n";
        exit(1);
    }

    v.vx = stod(x); v.vy = stod(y); v.vz = stod(z);
    v.vlength = sqrt(v.x() * v.x() + v.y() * v.y() + v.z() * v.z());
    return in;
}

std::ostream& operator<< (std::ostream& out, const Vector& v){
```

```
    std::cout << "(" << v.x() << ", " << v.y() << ", " << v.z() << "); length = " <<
v.length() << '\n';
    return out;
}

Vector operator+ (const Vector& v1, const Vector& v2){
    return {v1.x() + v2.x(), v1.y() + v2.y(), v1.z() + v2.z()};
}

Vector operator- (const Vector& v1, const Vector& v2){
    return {v1.x() - v2.x(), v1.y() - v2.y(), v1.z() - v2.z()};
}

double operator* (const Vector& v1, const Vector& v2){
    return v1.x() * v2.x() + v1.y() * v2.y() + v1.z() * v2.z();
}

Vector operator* (const int s, const Vector& v){
    return {v.x() * s, v.y() * s, v.z() * s};
}

Vector operator* (const Vector& v, const int s){
    return s * v;
}

Vector operator& (const Vector& v1, const Vector& v2){
    return {v1.y() * v2.z() - v1.z() * v2.y(),
            v1.z() * v2.x() - v1.x() * v2.z(),
            v1.x() * v2.y() - v1.y() * v2.x()};
}

double cos_angle(const Vector& v1, const Vector& v2){
    return (v1 * v2)/(v1.length() * v2.length());
}

double angle(const Vector& v1, const Vector& v2){
    double ca = cos_angle(v1, v2);
    if (ca >= 1 && round(ca) == 1){ return 0; }
    if (ca <= -1 && round(ca) == -1){ return acos(-1); }
    return acos(cos_angle(v1, v2));
}

int main(){

    Vector v1, v2; std::cin >> v1 >> v2;

    std::cout << "v1: " << v1 << "v2: " << v2;
    std::cout << "v1 + v2: " << v1 + v2 << "v1 - v2: " << v1 - v2;

    std::cin >> v1 >> v2;
    std::cout << "v1: " << v1 << "v2: " << v2;
    std::cout << "(v1, v2): " << v1 * v2 << '\n';
    std::cout << "[v1, v2]: " << (v1 & v2);
    std::cout << "-2 * v1: " << -2 * v1;

    std::cin >> v1 >> v2;
    std::cout << "v1: " << v1 << "v2: " << v2;
    std::cout << "v1 == v2: " << (v1.is_match(v2) ? "yes\n" : "no\n");
    std::cout << "|v1| == |v2|: " << (v1.is_equal(v2) ? "yes\n" : "no\n");
```

```
    std::cout << "cos(v1 ^ v2): " << cos_angle(v1, v2) << "\n";
    std::cout << "v1 ^ v2: " << angle(v1, v2) << "\n";


}
```