**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Айрапетова Евгения Ашотовна

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

.**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.

Операции сложения, вычитания, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов.

Необходимо реализовать пользовательский литерал для работы с константами типа **vector3D**.

1. Описание программы

На вход программе поступают координаты x, y, z двух векторов, которые хранятся в классе Vector3D, а также скаляр n.

С помощью перегрузки операторов осуществляются операции сложения векторов, вычитания b из a, векторное и скалярное произведение векторов, умножение вектора a на скаляр n, вычисляется длина векторов a и b, угол между ними и сравниваются их длины. Результат этих действий выводится на экран пользователя.

1. Набор тестов

Тест 1:

a{ 10.0\_ft, 12.0\_ft, 9.0\_ft }

b{ 5.0\_m, 2.55\_m, 3.14\_m }

n = 3.5

Тест 2:

a{ 3.28\_ft, 6.56\_ft, 6.56\_ft }

b{ 2.0\_m, 2.0\_m, 1.0\_m }

n = 2

Тест 3:

a{ 1.0\_ft, 2.0\_ft, 1.5\_ft }

b{ 3.0\_m, 3.0\_m, 4.0\_m }

n = 7

1. Результаты выполнения тестов

Тест 1:

Вектор a:

x = 3.04878

y = 3.65854

z = 2.7439

Вектор b:

x = 5

y = 2.55

z = 3.14

Сумма a + b:

x = 8.04878

y = 6.20854

z = 5.8839

Разность a - b:

x = -1.95122

y = 1.10854

z = -0.396098

Векторное произведение a \* b:

x = 4.49085

y = 4.14634

z = 6.43293

Скалярное произведение a & b: 33.189

Умножение вектора a на скаляр n:

x = 10.6707

y = 12.8049

z = 9.60366

Длина вектора a: 5.49627

Длина вектора b: 6.43134

Угол между векторами a и b: 0.00613201

a < b

Тест 2:

Вектор a:

x = 1

y = 2

z = 2

Вектор b:

x = 2

y = 2

z = 1

Сумма a + b:

x = 3

y = 4

z = 3

Разность a - b:

x = -1

y = 0

z = 1

Векторное произведение a \* b:

x = -2

y = 3

z = 2

Скалярное произведение a & b:8

Умножение вектора a на скаляр n:

x = 2

y = 4

z = 4

Длина вектора a: 3

Длина вектора b: 3

Угол между векторами a и b: 0.00830571

a = b

Тест 3:

Вектор a:

x = 0.304878

y = 0.609756

z = 0.457317

Вектор b:

x = 3

y = 3

z = 4

Сумма a + b:

x = 3.30488

y = 3.60976

z = 4.45732

Разность a - b:

x = -2.69512

y = -2.39024

z = -3.54268

Векторное произведение a \* b:

x = 1.06707

y = 0.152439

z = -1.47561

Скалярное произведение a & b:4.57317

Умножение вектора a на скаляр n:

x = 2.13415

y = 4.26829

z = 3.20122

Длина вектора a: 0.820909

Длина вектора b: 5.83095

Угол между векторами a и b: 0.00523255

a < b

1. Листинг программы

// Айрапетова Евгения М8о-206Б-19

#include <iostream>

#define PI 3.1415926535

using namespace std;

class Vector3D {

double x\_, y\_, z\_;

public:

Vector3D() { x\_ = 0; y\_ = 0; z\_ = 0; };

Vector3D(double x, double y, double z) {

x\_ = x;

y\_ = y;

z\_ = z;

}

double& x() { return x\_; }

double& y() { return y\_; }

double& z() { return z\_; }

double x() const { return x\_; }

double y() const { return y\_; }

double z() const { return z\_; }

};

Vector3D operator+ (const Vector3D& a, const Vector3D& b) {

return { a.x() + b.x(), a.y() + b.y(), a.z() + b.z() };

}

Vector3D operator- (const Vector3D& a, const Vector3D& b) {

return { a.x() - b.x(), a.y() - b.y(), a.z() - b.z() };

}

Vector3D operator\* (const Vector3D& a, const Vector3D& b) { // Векторное произведение векторов

return { (a.y() \* b.z()) - (b.y() \* a.z()), (b.x() \* a.z()) - (a.x() \* b.z()), (a.x() \* b.y()) - (b.x() - a.y()) };

}

Vector3D operator\* (const Vector3D& x, const double& a) { // Умножение вектора на скаляр

return { x.x() \* a, x.y() \* a, x.z() \* a };

}

double ScalarProduct (const Vector3D& a, const Vector3D& b) { // Скалярное произведение векторов

return { a.x() \* b.x() + a.y() \* b.y() + a.z() \* b.z() };

}

double length(Vector3D a) { // Длина вектора

return (sqrt((a.x() \* a.x()) + (a.y() \* a.y()) + (a.z() \* a.z())));

}

bool operator== (const Vector3D& a, const Vector3D& b) {

return { (a.x() == b.x()) && (a.y() == b.y()) && (a.z() == b.z()) };

}

bool operator!= (const Vector3D& a, const Vector3D& b) {

return { !(a == b) };

}

bool operator< (const Vector3D& a, const Vector3D& b) {

return { length(a) < length(b) };

}

bool operator<= (const Vector3D& a, const Vector3D& b) {

return { length(a) <= length(b) };

}

bool operator> (const Vector3D& a, const Vector3D& b) {

return { length(a) > length(b) };

}

bool operator>= (const Vector3D& a, const Vector3D& b) {

return { length(a) >= length(b) };

}

double corner(const Vector3D& a, const Vector3D& b) { // Угол между двумя векторами

double c = ScalarProduct(a, b) / (length(a) \* length(b));

return (acos(c) \* PI) / 180;

}

istream& operator>> (istream& input, Vector3D& v) {

input >> v.x() >> v.y() >> v.z();

return input;

}

ostream& operator<< (ostream& output, const Vector3D& v) {

output << "x = " << v.x() << '\n';

output << "y = " << v.y() << '\n';

output << "z = " << v.z() << '\n';

return output;

}

double operator"" \_m(const long double val) { // Пользовательский литерал для метров

return val;

}

double operator"" \_ft(const long double val) { // Пользовательский литерал для футов

return (val / 3.28);

}

int main() {

setlocale(LC\_ALL, "rus");

Vector3D a{ 1.0\_ft, 2.0\_ft, 1.5\_ft }, b{ 3.0\_m, 3.0\_m, 4.0\_m };

double n = 3.5;

try {

cout << "Вектор a:\n" << a << "\nВектор b:\n" << b;

cout << "\nСумма a + b:\n" << a + b;

cout << "\nРазность a - b:\n" << a - b;

cout << "\nВекторное произведение a \* b:\n" << a \* b;

cout << "\nСкалярное произведение a & b:" << ScalarProduct(a, b);

cout << "\nУмножение вектора a на скаляр n:\n" << a \* n;

cout << "\nДлина вектора a: " << length(a);

cout << "\nДлина вектора b: " << length(b) << '\n';

if (a > b) { cout << "a > b"; }

else if (a < b) { cout << "a < b"; }

else { cout << "a = b"; }

}

catch (const exception & ex) {

cout << ex.what() << endl;

}

}

1. Вывод:

В результате выполнения лабораторной работы можно сделать вывод, что использование пользовательских литералов может облегчить написание программы и помогает приводить типы к одному виду.

1. Список литературы:

Пользовательские литералы: https://habr.com/ru/post/140357/