Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»
Дисциплина «Объектно-ориентированное программирование»

Лабораторная работа №2

Тема: Операторы, литералы

Студент: Инютин М. А.
Группа: М8О-207Б-19
Преподаватель: Чернышев Л. Н.
Дата:
Оценка:

Москва, 2020

1. Постановка задачи

Изучить механизм перегрузки операторов и механизм работы с пользовательскими литералами.

Вариант 11. Создать класс Vector3D, задаваемый тройкой координат. Обязательно должны быть реализованы: операции сложения и вычитания векторов, векторное произведение векторов, скалярное произведение векторов, умножение на скаляр, сравнение векторов на совпадение, вычисление длины вектора, сравнение длин векторов, вычисление угла между векторами. Операции сложения, вычитания, сравнений (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа Vector3D.

2. Описание программы

Для класса Vector3D создадим три приватных члена - координаты в ортонормированном базисе $\vec{i}$, $\vec{j}$, $\vec{k}$. Реализуем конструктор с параметрами и без, перегрузим операции сложения и вычитания векторов, умножения на число, отношения меньше, больше и равно по длинам векторов. Создадим статические методы для вычисления векторного и скалярного произведения, угла между векторами. Наконец, реализуем пользовательский литер для констант типа Vector3D с функции std::stold. Чтобы было проще писать и читать код, вынесем реализацию методов класса в отдельный файл. В основном файле реализуем ввод и вывод данных, взаимодействие с методами класса.

3. Набор тестов

На ввод программе поступает два вектора. Каждый вектор описан тремя координатами в базисе.

Тест №1
1 2 0
0 0 2

Тест №2
1.0 2.0 0.0
-2.0 1.0 0.0

Тест №3
3.5 4.5 0
4.2 4.3 2.718281828

Тест №4
8 -7 -2
7 -11 8

Тест №5
1.23 1.23 1.23
1.23 1.23 1.23

Тест №6
1.0 1.0 1.0
-1.0 -1.0 -1.0

4. Результаты выполнения тестов

Программа выводит введённые вектора, вектор с меньшей и большей длинной, скалярное произведение векторов, угол между ними и векторное произведение.

Тест №1

```
cat test1.txt | ./oop_exercise_02
Your input:
a = (1, 2, 0)
b = (0, 0, 2)
Shortest vector: (0, 0, 2)
Longest vector: (1, 2, 0)
Dot product of your vectors: 0
Angle between your vectors in radians: 1.5708
Cross product of your vectors: (4, -2, 0)
```

Тест №2

```
cat test2.txt | ./oop_exercise_02
Your input:
a = (1, 2, 0)
b = (-2, 1, 0)
Shortest vector: (-2, 1, 0)
Longest vector: (-2, 1, 0)
Dot product of your vectors: 0
Angle between your vectors in radians: 1.5708
Cross product of your vectors: (0, -0, 5)
```

Тест №3

```
cat test3.txt | ./oop_exercise_02
Your input:
a = (3.5, 4.5, 0)
b = (4.2, 4.3, 2.71828)
Shortest vector: (3.5, 4.5, 0)
Longest vector: (4.2, 4.3, 2.71828)
Dot product of your vectors: 34.05
Angle between your vectors in radians: 0.438499
Cross product of your vectors: (12.2323, -9.51399, -3.85)
```

Тест №4
cat test4.txt | ./oop_exercise_02
Your input:
a = (8, -7, -2)
b = (7, -11, 8)
Shortest vector: (8, -7, -2)
Longest vector: (7, -11, 8)
Dot product of your vectors: 117
Angle between your vectors in radians: 0.785398
Cross product of your vectors: (-78, -78, -39)

Тест №5
cat test5.txt | ./oop_exercise_02
Your input:
a = (1.23, 1.23, 1.23)
b = (1.23, 1.23, 1.23)
Shortest vector: (1.23, 1.23, 1.23)
Longest vector: (1.23, 1.23, 1.23)
Dot product of your vectors: 4.5387
Angle between your vectors in radians: 0
Cross product of your vectors: (0, 0, 0)

Тест №6
cat test6.txt | ./oop_exercise_02
Your input:
a = (1, 1, 1)
b = (-1, -1, -1)
Shortest vector: (-1, -1, -1)
Longest vector: (-1, -1, -1)
Dot product of your vectors: -3
Angle between your vectors in radians: 3.14159
Cross product of your vectors: (0, 0, 0)

5. Листинг программы

Программа представлена в трёх файлах: объявление класса в adress.hpp, реализация методов в adress.cpp и взаимодействие с объектами класса в main.cpp.

vector3d.hpp

```
#ifndef VECTOR3D_HPP
#define VECTOR3D_HPP

#include <bits/stdc++.h>

const long double EPS = 1e-6;

class Vector3D {
private:
     long double X, Y, Z;
     friend Vector3D operator + (const Vector3D &lhs,
                                 const Vector3D &rhs);
     friend Vector3D operator * (const long double &num,
                                 const Vector3D &vec);
     friend Vector3D operator - (const Vector3D &lhs,
                                 const Vector3D &rhs);
     friend std::ostream & operator << (std::ostream &out,
                                        const Vector3D &vec);
    friend std::istream & operator >> (std::istream &in,
                                       Vector3D &vec);
     friend bool operator == (Vector3D &lhs, Vector3D &rhs);
     friend bool operator < (Vector3D &lhs, Vector3D &rhs);
     friend bool operator > (Vector3D &lhs, Vector3D &rhs);
public:
     Vector3D() : X(NAN), Y(NAN), Z(NAN) {};
     Vector3D(long double x, long double y,
              long double z) : X(x), Y(y), Z(z) {};
     ~Vector3D() {};
     long double Length();
     bool IsNaN();
     static bool Equal(const Vector3D &lhs, const Vector3D &rhs);
     static Vector3D CrossProduct(const Vector3D &lhs,
                                  const Vector3D &rhs);
     static long double DotProduct(const Vector3D &lhs,
                                   const Vector3D &rhs);
     static long double Angle(Vector3D &lhs, Vector3D &rhs);
};

Vector3D operator"" _vector3d(const char *str, size_t n);

#endif // VECTOR3D_HPP
```

vector3d.cpp

```
#include "vector3d.hpp"

Vector3D operator + (const Vector3D &lhs, const Vector3D &rhs) {
     return Vector3D(lhs.X + rhs.X, lhs.Y + rhs.Y, lhs.Z + rhs.Z);
}

Vector3D operator * (const long double &num, const Vector3D &vec)
{
     return Vector3D(num * vec.X, num * vec.Y, num * vec.Z);
}

Vector3D operator - (const Vector3D &lhs, const Vector3D &rhs) {
     return lhs + (-1.0) * rhs;
}

std::ostream & operator << (std::ostream &out, const Vector3D
&vec) {
     out << "("
               << vec.X << ", "
               << vec.Y << ", "
               << vec.Z << ")";
     return out;
}

std::istream & operator >> (std::istream &in,  Vector3D &vec) {
     in >> vec.X >> vec.Y >> vec.Z;
     return in;
}

bool operator == (Vector3D &lhs, Vector3D &rhs) {
     return std::abs(lhs.Length() - rhs.Length()) < EPS;
}

bool operator < (Vector3D &lhs, Vector3D &rhs) {
     return lhs.Length() < rhs.Length();
}

bool operator > (Vector3D &lhs, Vector3D &rhs) {
     return lhs.Length() > rhs.Length();
}

long double Vector3D::Length() {
     return std::sqrt(DotProduct(*this, *this));
}

bool Vector3D::IsNaN() {
     return std::isnan(this->X) or std::isnan(this->Y) or
std::isnan(this->Z);
}
```

```
bool Vector3D::Equal(const Vector3D &lhs, const Vector3D &rhs) {
     return (std::abs(lhs.X - rhs.X) < EPS) && (std::abs(lhs.Y -
rhs.Y) < EPS) && (std::abs(lhs.Z - rhs.Z) < EPS);
}

Vector3D Vector3D::CrossProduct(const Vector3D &lhs,
                                const Vector3D &rhs) {
     return Vector3D(lhs.Y * rhs.Z - lhs.Z * rhs.Y,
                     lhs.Z * rhs.X - lhs.X * rhs.Z,
                     lhs.X * rhs.Y - lhs.Y * rhs.X);
}

long double Vector3D::DotProduct(const Vector3D &lhs,
                                 const Vector3D &rhs) {
     return lhs.X * rhs.X + lhs.Y * rhs.Y + lhs.Z * rhs.Z;
}

long double Vector3D::Angle(Vector3D &lhs, Vector3D &rhs) {
     long double res = std::acos(DotProduct(lhs, rhs)
                           / (lhs.Length() * rhs.Length()));
     return std::isnan(res) ? 0 : res;
}

Vector3D operator"" _vector3d(const char *str, size_t n) {
     long double *cords = new long double[3];
     for (int i = 0; i < 3; ++i) {
          cords[i] = NAN;
     }
     std::string s(str);
     s = s + ',';
     size_t j = 0, last = 0;
     for (size_t i = 0; i < n + 1; ++i) {
          if (s[i] == ',' and j < 3) {
               try {
                    cords[j] = std::stold(s.substr(last,
                                                   i - last + 1));
               } catch (std::invalid_argument) {
                    ;
               }
               last = i + 1;
               ++j;
          }
     }
     return Vector3D(cords[0], cords[1], cords[2]);
}
```

main.cpp

```cpp
#include "vector3d.hpp"

/*
 * Инютин М А М8О-207Б-19
 * Создать класс Vector3D, задаваемый тройкой координат.
 * Обязательно должны быть реализованы: операции сложения и
 * вычитания векторов, векторное произведение векторов,
 * скалярное произведение векторов, умножение на скаляр, сравнение
 * векторов на совпадение, вычисление длины вектора, сравнение
 * длин векторов, вычисление угла между векторами. Операции
 * сложения, вычитания, сравнений (на равенство, больше и меньше)
 * должны быть выполнены в виде перегрузки операторов.
 * Необходимо реализовать пользовательский литерал для работы
 * с константами типа Vector3D.
 */

signed main() {
     std::cout << "Ortonormal basis:" << "\n";
     std::cout << "i = " << "1,0,0"_vector3d << "\n";
     std::cout << "j = " << "0,1,0"_vector3d << "\n";
     std::cout << "k = " << "0,0,1"_vector3d << "\n";
     Vector3D a, b;
     std::cout << "Input three coordinates of first vector
separated with \' \': ";
     std::cin >> a;
     std::cout << "Input three coordinates of second vector
separated with \' \': ";
     std::cin >> b;
     if (a.IsNaN() or b.IsNaN()) {
          std::cout << "Invalid input! Programm terminates with
exit code -1\n";
          return -1;
     }
     std::cout << "Your input:\n";
     std::cout << "a = " << a << "\n";
     std::cout << "b = " << b << "\n";
     std::cout << "Shortest vector: " << (a < b ? a : b) << "\n";
     std::cout << "Longest vector: " << (a > b ? a : b) << "\n";
     std::cout << "Dot product of your vectors: ";
     std::cout << Vector3D::DotProduct(a, b) << "\n";
     std::cout << "Angle between your vectors in radians: ";
     std::cout << Vector3D::Angle(a, b) << "\n";
     std::cout << "Cross product of your vectors: ";
     std::cout << Vector3D::CrossProduct(a, b) << "\n";
     return 0;
}
```

6. Выводы

Я научился перегружать операторы для классов на языке C++, создавать пользовательские литеры для работы с константами. Чтобы реализовать проверку корректности введённых данных, изучил конструкцию try/catch, которая спасла от аварийного завершения программу. Во время выполнения лабораторной работы вспомнил половину курса аналитической геометрии. Теперь буду использовать эту программу для решения геометрических задач.

7. Список литературы

1. Перегрузка операторов в C++ / Хабр — Habr
   URL: https://habr.com/ru/post/132014/ (дата обращения: 30.09.2020)
2. Пользовательские литералы в C++11 / Хабр — Habr
   URL: https://habr.com/ru/post/140357/ (дата обращения: 30.09.2020)