**Лабораторная работа №1**

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студент: *Капичников Ярослав Андреевич, группа М8О-207Б-20*

Преподаватель: *Дорохов Евгений Павлович, каф. 806*

**Задание:(Вариант №11)**

**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.

**Описание программы:**

Исходный код разделён на несколько файлов:

* vector.h – описание класса вектора.
* vector.cpp – реализация функций класса вектор.

**Дневник отладки**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Дата | Событие | Действие по исправлению |
| 1 |  |  |  |
|  |  |  |  |

**Вывод:**  
 Проделав данную работу, я изучил основы принципов ооп, создал класс векторов, для которого реализовал множество арифметических операций. Создание класса в C++ очень схоже с созданием структур в си, с чем мы уже сталкивались в предыдущем семестре.

**Исходный код:**

**Vector.h**

|  |
| --- |
| #pragma once |
|  | #include <iostream> |
|  |  |
|  |  |
|  | class Vector { |
|  | public: |
|  | Vector(); |
|  | Vector(std::istream& is); |
|  | Vector(double x, double y, double z); |
|  |  |
|  | double dist(Vector& other); |
|  | double getX(); |
|  | double getY(); |
|  | double getZ(); |
|  |  |
|  | void setX(double a); |
|  | void setY(double a); |
|  | void setZ(double a); |
|  |  |
|  | friend std::istream& operator>>(std::istream& is, Vector& p); |
|  | friend std::ostream& operator<<(std::ostream& os, Vector& p); |
|  |  |
|  | friend bool operator== (Vector& p1, Vector& p2); |
|  | friend bool operator!= (Vector& p1, Vector& p2); |
|  |  |
|  | friend Vector operator+ (Vector& v1, Vector& v2); |
|  | friend Vector operator- (Vector& v1, Vector& v2); |
|  |  |
|  | friend Vector operator\* (Vector& v1,Vector& v2); |
|  | friend Vector operator\* (Vector& v1, double a); |
|  |  |
|  |  |
|  | double Len(); |
|  |  |
|  | double Angle( Vector& v2); |
|  | double SkalarUm(Vector& v2); |
|  |  |
|  | private: |
|  | double x\_; |
|  | double y\_; |
|  | double z\_; |
|  | }; |

**Vector.cpp**

|  |
| --- |
| #include "vector.h" |
|  |  |
|  | #include <cmath> |
|  |  |
|  | Vector::Vector() : x\_(0.0), y\_(0.0), z\_(0.0) {} |
|  |  |
|  | Vector::Vector(double x, double y, double z) : x\_(x), y\_(y), z\_(z) {} |
|  |  |
|  | Vector::Vector(std::istream& is) { |
|  | is >> x\_ >> y\_ >> z\_; |
|  | } |
|  |  |
|  | double Vector::dist(Vector& other) { |
|  | double dx = (other.x\_ - x\_); |
|  | double dy = (other.y\_ - y\_); |
|  | return std::sqrt(dx \* dx + dy \* dy); |
|  | } |
|  |  |
|  | double Vector::getX() |
|  | { |
|  | return x\_; |
|  | } |
|  |  |
|  | double Vector::getY() |
|  | { |
|  | return y\_; |
|  | } |
|  |  |
|  | double Vector::getZ() |
|  | { |
|  | return z\_; |
|  | } |
|  |  |
|  | void Vector::setX(double a) |
|  | { |
|  | x\_ = a; |
|  | } |
|  |  |
|  | void Vector::setY(double a) |
|  | { |
|  | y\_ = a; |
|  | } |
|  |  |
|  | void Vector::setZ(double a) |
|  | { |
|  | z\_ = a; |
|  | } |
|  |  |
|  | double Vector::Len() |
|  | { |
|  | double l = sqrt(x\_ \* x\_ + y\_ \* y\_ + z\_ \* z\_); |
|  | return l; |
|  | } |
|  |  |
|  | double Vector:: Angle( Vector& v2) |
|  | { |
|  | double cos = (x\_ \* v2.x\_ + y\_ \* v2.y\_ + z\_ \* v2.z\_) / (Len() \* v2.Len()); |
|  | return acos(cos) \* 180/3.1415; |
|  | } |
|  |  |
|  | std::istream& operator>>(std::istream& is, Vector& p) { |
|  | is >> p.x\_ >> p.y\_ >> p.z\_; |
|  | return is; |
|  | } |
|  |  |
|  | std::ostream& operator<<(std::ostream& os, Vector& p) { |
|  | os << "(" << p.x\_ << ", " << p.y\_ << ", " << p.z\_ << ")"; |
|  | return os; |
|  | } |
|  |  |
|  | bool operator== (Vector& p1, Vector& p2) |
|  | { |
|  | return (p1.getX() == p2.getX() && |
|  | p1.getY() == p2.getY() && |
|  | p1.getZ() == p2.getZ()); |
|  | } |
|  |  |
|  | bool operator!= (Vector& p1, Vector& p2) |
|  | { |
|  | return !(p1 == p2); |
|  | } |
|  |  |
|  | Vector operator+ (Vector& v1, Vector& v2) |
|  | { |
|  | Vector v3; |
|  | v3.x\_ = v1.x\_ + v2.x\_; |
|  | v3.y\_ = v1.y\_ + v2.y\_; |
|  | v3.z\_ = v1.z\_ + v2.z\_; |
|  |  |
|  | return v3; |
|  | } |
|  |  |
|  | Vector operator- (Vector& v1, Vector& v2) |
|  | { |
|  | Vector v3; |
|  | v3.x\_ = v1.x\_ - v2.x\_; |
|  | v3.y\_ = v1.y\_ - v2.y\_; |
|  | v3.z\_ = v1.z\_ - v2.z\_; |
|  |  |
|  | return v3; |
|  | } |
|  |  |
|  | Vector operator\* (Vector& v1, Vector& v2) |
|  | { |
|  | Vector v3; |
|  | v3.x\_ = v1.y\_ \* v2.z\_ - v1.z\_ \* v2.y\_; |
|  | v3.y\_ = v1.z\_ \* v2.x\_ - v1.x\_ \* v2.z\_; |
|  | v3.z\_ = v1.x\_ \* v2.y\_ - v1.y\_ \* v2.x\_; |
|  |  |
|  | return v3; |
|  | } |
|  |  |
|  | Vector operator\* (Vector& v1, double a) |
|  | { |
|  | Vector v3; |
|  | v3.x\_ = v1.x\_ \* a; |
|  | v3.y\_ = v1.y\_ \* a; |
|  | v3.z\_ = v1.z\_ \* a; |
|  |  |
|  | return v3; |
|  | } |
|  |  |
|  | double Vector::SkalarUm(Vector& v2) |
|  | { |
|  | double s = x\_ \* v2.x\_ + y\_ \* v2.y\_ + z\_ \* v2.z\_; |
|  | return s; |
|  | } |