Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

Современные языки программирования

Отчет по лабораторной работе №3

«Использование языка программирования Swift: структуры и классы, методы»

Выполнил: Канунникова В. В.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

**Цель работы**: выполнить разработку приложения с использованием языка программирования Swift: функции, замыкания, перечисления.

**Вариант 9**: Экземляр класса задается тройкой координат в трехмерном пространстве (x,y,z). Обязательно должны быть реализованы методы: приведение вектора к строке с выводом координат, сложение векторов, вычитание векторов, скалярное произведение, умножение и деление на скаляр, векторное произведение, вычисление длинны вектора.

**Код программы:**

// Validator.swift

import Foundation

public struct Validator {

public static func validate(x: Double, y: Double, z: Double) -> Bool {

// метод возвращает логическое значение, которое является результатом логического И операций

// проверяет, является ли значение конечным числом

return x.isFinite && y.isFinite && z.isFinite

}

}

// Vector3D.swift

import Foundation

public class Vector3D {

private var x: Double

private var y: Double

private var z: Double

public init?(x: Double, y: Double, z: Double) { // ? означает, что конструктор может вернуть nil, если что-то пойдет не так

guard Validator.validate(x: x, y: y, z: z) else { return nil } // проверка, что координаты x, y и z правильные.

self.x = x // значение, переданное в конструктор как параметр x, будет сохранено в свойстве объекта x

self.y = y

self.z = z

}

public func toString() -> String { // преобразует объект класса в строку, которая содержит координаты вектора

return "(\(x), \(y), \(z))"

}

public func add(\_ vector: Vector3D) -> Vector3D { // сложение двух векторов

return Vector3D(x: self.x + vector.x, y: self.y + vector.y, z: self.z + vector.z)!

}

public func subtract(\_ vector: Vector3D) -> Vector3D { // вычитание двух векторов

return Vector3D(x: self.x - vector.x, y: self.y - vector.y, z: self.z - vector.z)!

}

public func dotProduct(\_ vector: Vector3D) -> Double { // вычисление скалярного произведения

return self.x \* vector.x + self.y \* vector.y + self.z \* vector.z

}

public func multiplyByScalar(\_ scalar: Double) -> Vector3D { // умножения на скаляр

return Vector3D(x: self.x \* scalar, y: self.y \* scalar, z: self.z \* scalar)!

}

public func divideByScalar(\_ scalar: Double) -> Vector3D { // деление на скаляр

return Vector3D(x: self.x / scalar, y: self.y / scalar, z: self.z / scalar)!

}

public func crossProduct(\_ vector: Vector3D) -> Vector3D { // векторное произведение

return Vector3D(

x: self.y \* vector.z - self.z \* vector.y,

y: self.z \* vector.x - self.x \* vector.z,

z: self.x \* vector.y - self.y \* vector.x

)!

}

public func length() -> Double { // длина вектора

return sqrt(self.x \* self.x + self.y \* self.y + self.z \* self.z)

}

}

// main.swift

import Foundation

// функция для чтения ввода и создания вектора

func readVectorInput() -> Vector3D? {

print("Введите координаты вектора (x, y, z) через пробел:")

guard let input = readLine(), // чтение строки ввода с консоли и присваивание её переменной

//если ввод неудачный, выполнение функции прерывается.

let x = Double(input.components(separatedBy: " ")[0]),

let y = Double(input.components(separatedBy: " ")[1]),

let z = Double(input.components(separatedBy: " ")[2]),

Validator.validate(x: x, y: y, z: z) else { // проверка корректности значений x, y и z

print("Неверный ввод. Попробуйте еще раз.")

return nil

}

return Vector3D(x: x, y: y, z: z) // создание и возвращение экземпляра класса Vector3D с указанными координатами x, y и z

}

// считываем первый вектор

print("Введите первый вектор:")

guard let vector1 = readVectorInput() else { // проверка, удалось ли функции создать объект

fatalError("Ошибка ввода первого вектора.")

}

// считываем второй вектор

print("Введите второй вектор:")

guard let vector2 = readVectorInput() else { // проверка, удалось ли функции создать объект

fatalError("Ошибка ввода второго вектора.")

}

// считываем скаляр

print("Введите скаляр:")

guard let scalarInput = readLine(), // считывает строку, введенную пользователем с клавиатуры

let scalar = Double(scalarInput), // преобразует введенную строку в число типа Double

scalar.isFinite else { // проверяет, является ли значение конечным числом

fatalError("Ошибка ввода скаляра.")

}

print("Вектор 1: \(vector1.toString())")

print("Вектор 2: \(vector2.toString())")

print("Добавление: \(vector1.add(vector2).toString())")

print("Вычитание: \(vector1.subtract(vector2).toString())")

print("Скалярное произведение: \(vector1.dotProduct(vector2))")

print("Векторное произведение: \(vector1.crossProduct(vector2).toString())")

print("Длина вектора 1: \(vector1.length())")

print("Длина вектора 2: \(vector2.length())")

print("Вектор 1, умноженный на скаляр: \(vector1.multiplyByScalar(scalar).toString())")

print("Вектор 1, разделенный на скалярный: \(vector1.divideByScalar(scalar).toString())")

print("Вектор 2, умноженный на скаляр: \(vector2.multiplyByScalar(scalar).toString())")

print("Вектор 2, разделенный на скалярный: \(vector2.divideByScalar(scalar).toString())")

Результат работы программы представлен на рисунке 1.

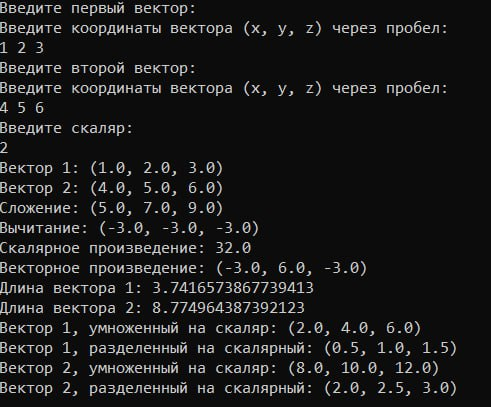


Рисунок 1 – Результат работы программы

**Ответы на вопросы к лабораторной работе:**

1. Что такое класс?

Класс — это шаблон или схема для создания объектов. Он определяет атрибуты (данные) и методы (функции), которые объект, созданный на основе этого класса, будет иметь. Класс представляет собой абстракцию для группировки данных и логики, которые оперируют этими данными.

2. Что такое структура?

Структура (struct) — это тип данных, который используется для группировки различных типов данных под одним именем. В отличие от классов, структуры чаще используются для хранения простых наборов данных и менее функционально насыщены. Структуры также обладают возможностью определять методы и конструкторы, как и классы.

3. Чем класс отличается от структуры?

Основные различия между классами и структурами:

* Тип данных: Классы являются ссылочными типами, а структуры — типами значений.
* Передача: Объекты классов передаются по ссылке, тогда как экземпляры структур передаются по значению.
* Наследование: Классы поддерживают наследование, а структуры — нет.
* Память: Классы создаются в управляемой куче (heap), а структуры — на стеке (stack).

4. Что такое объект?

Объект — это экземпляр класса. Он представляет собой конкретный пример абстрактного класса, в котором определены атрибуты (состояние) и методы (поведение), характерные для этого класса. Например, если класс — это чертеж автомобиля, то объект — это конкретный автомобиль, созданный по этому чертежу

5. Типы значений. Ссылочные типы. Чем отличаются друг от друга?

Типы значений (Value Types): хранят данные непосредственно в памяти, в переменной. Примеры: int, float, struct.

Ссылочные типы (Reference Types): хранят ссылку на адрес в памяти, где находятся данные. Примеры: string, array, class.

Различие заключается в способе хранения и передаче данных. Типы значений хранятся в стеке и передаются по значению, тогда как ссылочные типы хранятся в управляемой куче и передаются по ссылке.

6. Для чего используются указатели?

Указатели используются для хранения адресов памяти других переменных. Они позволяют работать с памятью на низком уровне, получая доступ и манипулируя данными по адресам. Указатели полезны в таких задачах, как управление динамической памятью, реализация структур данных и взаимодействие с оборудованием.

7. Ключевое слово self.

В языке программирования Swift ключевое слово self используется для обозначения текущего экземпляра класса, структуры или перечисления. Оно позволяет методам инициализаторам обращаться к свойствам и другим методам текущего объекта.

**Вывод**: был изучен язык программирования Swift, а именно информацию о структурах и классах, методах, и использован для разработки программы.