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: Еремейко А. А.

Студент группы 310901

Проверил: Усенко Ф.В.

Минск 2024

**Цель работы:** Выполнить разработку приложения с использованием языка программирования *Swift*: структуры и классы, методы.

**Вариант 2:** Николай – оригинальный человек. Он решил создать класс Nikola, принимающий при инициализации 2 параметра: имя и возраст. Но на этом он не успокоился. Не важно, какое имя передаст пользователь при создании экземпляра, оно всегда будет содержать «Николая». В частности – если пользователя на самом деле зовут Николаем, то с именем ничего не произойдет, а если его зовут, например, Максим, то оно преобразуется в “Я не Максим, а Николай”.

**Текст программы:**

class Nikola {

    private var internalName: String = "Николай"

    private var internalAge: Int = 0

    var name: String {

        get {

            return internalName

        }

        set(newName) {

            if newName.isEmpty {

                print("Ошибка: имя не может быть пустым. Установлено имя по умолчанию 'Николай'.\n")

                internalName = "Николай"

            } else if newName == "Николай" {

                internalName = newName

            } else {

                internalName = "Я не \(newName), а Николай"

            }

        }

    }

    var age: Int {

        get {

            return internalAge

        }

        set(newAge) {

            if newAge < 0 || newAge > 100 {

                print("Ошибка: возраст должен быть в диапазоне от 0 до 100. Установлен возраст по умолчанию 0.\n")

                internalAge = 0

            } else {

                internalAge = newAge

            }

        }

    }

    init(name: String, age: Int) {

        self.name = name

        self.age = age

    }

    func getInfo() -> String {

        return "Имя: \(name), Возраст: \(age)\n"

    }

}

print("\n")

// Примеры использования

let user1 = Nikola(name: "Максим", age: 25)

print(user1.getInfo()) // Вывод: "Имя: Я не Максим, а Николай, Возраст: 25"

let user2 = Nikola(name: "Николай", age: 30)

print(user2.getInfo()) // Вывод: "Имя: Николай, Возраст: 30"

// Изменение значений

user1.name = ""

user1.age = 200

print(user1.getInfo()) // Вывод: "Имя: Николай, Возраст: 0"

// Корректное изменение значений

user2.name = "Анна"

user2.age = 35

print(user2.getInfo()) // Вывод: "Имя: Я не Анна, а Николай, Возраст: 35"

Результат работы программы приведен на рисунке 1.

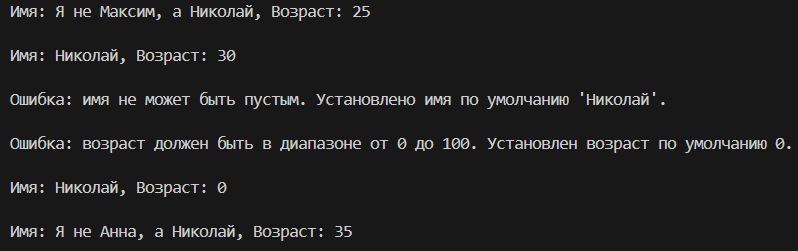


Рисунок 1 – Результат работы программы

**Ответы на вопросы к лабораторной работе:**

1. Что такое класс?

Класс в Swift — это пользовательский тип данных, который позволяет создавать объекты с определённым набором свойств (данных) и методов (функций). Классы поддерживают наследование, что означает, что один класс может наследовать свойства и методы другого.

2. Что такое структура?

Структура — это тип данных, который похож на класс, но имеет свои особенности. Она также может содержать свойства и методы, но является типом значения, а не ссылочным типом. Это означает, что при передаче структуры данные копируются, а не передаются по ссылке. Структуры в Swift чаще используются для более простых моделей данных, не требующих сложных механизмов наследования.

3. Чем класс отличается от структуры?

Тип данных: Классы — ссылочные типы, структуры — типы значений.

Наследование: Классы поддерживают наследование, структуры — нет.

Управление памятью: Классы используют автоматическое управление памятью через подсчёт ссылок (ARC), а структуры копируются при передаче.

Инициализация: Структуры предоставляют автоматическую реализацию инициализаторов для всех свойств, классы требуют их явного определения, если свойства не имеют начальных значений.

Приведение типов: Классы поддерживают приведение типов, структуры — нет.

4. Что такое объект?

Объект — это экземпляр класса. Он создаётся из класса и представляет собой конкретную реализацию его свойств и методов. Объекты позволяют взаимодействовать с данными и функциональностью, определённой в классе, и являются основным элементом объектно-ориентированного программирования.

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

Типы значений: Данные копируются при передаче. Это означает, что изменения, внесённые в копию, не влияют на оригинал. Примеры: структуры, перечисления, целые числа, строки.

Ссылочные типы: При передаче копируется ссылка на данные. Это означает, что изменения, внесённые в одну ссылку, отразятся на всех остальных. Примеры: классы, функции.

Главное отличие состоит в управлении памятью: типы значений создают независимые копии, а ссылочные типы работают с одними и теми же данными через разные ссылки.

6. Для чего используются указатели?

Указатели используются для работы с адресами памяти, чтобы обращаться напрямую к данным, расположенным в определённой области памяти. В Swift указатели встречаются редко, так как язык ориентирован на безопасное управление памятью. Однако они могут использоваться при работе с низкоуровневым кодом, взаимодействии с C/Objective-C библиотеками или при выполнении специфических задач, таких как оптимизация производительности.

7. Ключевое слово self.

Ключевое слово self используется внутри класса или структуры для ссылки на текущий экземпляр. Оно помогает обращаться к свойствам и методам объекта изнутри, разрешать конфликты именования между свойствами и параметрами метода, а также передавать текущий экземпляр как аргумент другим функциям.

**Вывод:** Таким образом, в ходе лабораторной работы я изучил структуры, классы и методы в языке программирования *Swift* и выполнил разработку приложения с использованием данного языка.