**Лабораторный Практикум 2: Основы SWIFT. Управление потоком выполнения**

**Цель:** Изучить основные типы данных в Swift, освоить использование условных операторов и циклов для управления логикой программы. --- Часть 1: Основные типы данных в Swift Swift — это язык со строгой типизацией, но он использует вывод типов, что делает код чище.

Swift

**Целочисленные типы**

let age: Int = 25 // Константа

var year: Int = 2024 // Переменная

var height = 180 // Тип Int выведен автоматически

**Числа с плавающей точкой**

let pi: Double = 3.14159

var weight: Float = 68.5

let temperature = 23.7 // Тип Double выведен по умолчанию

**Логический тип**

var isStudent: Bool = true

let hasCompletedLab = false

**Строковый тип**

let name: String = "Анна"

var greeting = "Привет, " // Вывод типа String

**Символьный тип (редко используется отдельно)**

let firstLetter: Character = "A"

**Составные типы**

var numbers: [Int] = [1, 2, 3, 4, 5] // Массив

var person: (String, Int) = ("Иван", 30) // Кортеж

**Ключевые моменты:** · let объявляет константу (значение нельзя изменить). · var объявляет переменную (значение можно менять). · Тип можно указывать явно через двоеточие, а можно доверить вывод компилятору.

**Часть 2: Условные операторы Условные операторы позволяют выполнять разные блоки кода в зависимости от выполнения условий.**

**2.1. Оператор if-else**

let score = 85

if score >= 90 {

print("Отлично! Оценка A")

} else if score >= 75 {

print("Хорошо! Оценка B")

} else if score >= 60 {

print("Удовлетворительно. Оценка C")

} else {

print("Неудовлетворительно. Необходимо повторить материал.")

}

**2.2.** Оператор switch-case Более мощная и безопасная альтернатива длинным цепочкам if-else.

let weather = "солнечно"

switch weather {

case "дождь":

print("Возьмите зонт! ☔️")

case "снег":

print("Оденьтесь теплее! ⛄️")

case "солнечно":

print("Идеальный день для прогулки! 😎")

// fallthrough // Ключевое слово для "проваливания" в следующий case

default: // Обязательный блок, если не все случаи покрыты

print("Неизвестная погода. Одевайтесь по обстоятельствам.")

}

**Особенности Swift:**

Switch должен быть полным (исчерпывать все возможные значения) или иметь default. · Нет неявного "проваливания" между случаями (case). Используйте fallthrough явно.

**Часть 3: Циклы Циклы используются для многократного выполнения блока кода.**

**3.1**. Цикл for-in Используется для итерации по последовательностям (массивы, диапазоны, словари и т.д.).

Swift

**Итерация по диапазону**

print("Счет от 1 до 5:")

for number in 1...5 { // Закрытый диапазон (включая 5)

print(number)

}

print("Каждый второй элемент:")

for number in stride(from: 0, to: 10, by: 2) { // от 0 до 10 (не включая), шаг 2

print(number)

}

**Итерация по массиву**

let fruits = ["Яблоко", "Банан", "Апельсин"]

for fruit in fruits {

print("Я люблю \(fruit)") // Структура "\(variable)" для интерполяции строк

}

**Если нужен индекс**

for (index, fruit) in fruits.enumerated() {

print("Фрукт №\(index + 1): \(fruit)")

}

**3.2.** Цикл while Выполняет блок кода, пока условие истинно.

var counter = 5

print("Обратный отсчет:")

while counter > 0 {

print(counter)

counter -= 1 // Уменьшаем счетчик. Важно: без этого цикл стал бы бесконечным!

}

print("Старт!")

**3.3.** Цикл repeat-while (аналог do-while) Тело цикла выполняется как минимум один раз, а затем проверяется условие.

var userInput: String

repeat {

print("Введите 'выход' для завершения:")

**Эмуляция ввода. В реальном приложении использовали бы readLine()**

userInput = "выход"

} while userInput != "выход"

print("Цикл завершен.")

**Часть 4: Управление циклами (Остановка и смена логики)**

Эти операторы позволяют гибко управлять выполнением циклов. · break — немедленно завершает выполнение всего цикла. · continue — немедленно завершает текущую итерацию цикла и переходит к следующей. · fallthrough — используется в switch для проваливания в следующий case. Пример использования break и continue:

Swift

**Поиск первого четного числа в массиве и остановка**

let numbersArray = [1, 3, 8, 5, 10, 7]

for number in numbersArray {

if number % 2 == 0 {

print("Найдено первое четное число: \(number)")

break // Выходим из цикла, т.к. нашли то, что искали

}

}

**Печать только нечетных чисел**

print("Нечетные числа в массиве:")

for number in numbersArray {

if number % 2 == 0 {

continue // Пропускаем четные числа и переходим к следующей итерации

}

print(number)

}