Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

Современные языки программирования

Отчет по лабораторной работе №2

«Использование языка программирования Swift: функции, замыкания, перечисления»

Выполнил: Капуза А.И.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

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

**Вариант 10, Задание 1:** Дано натуральное число P, заданное в восьмеричной системе счисления. Напишите программу перевода этого числа в двоичную систему счисления. Встроенный метод перевода не использовать.

**Текст программы:**

import Foundation

// Перечисление для обработки ошибок, связанных с некорректными символами

enum NumberSystemError: Error

{ case invalidCharacter }

// Функция для перевода числа из восьмеричной системы счисления в двоичную

func octalToBinary(octal: String) throws -> String {

// Словарь для сопоставления восьмеричных цифр с двоичными строками

let octalToBinaryMap: [Character: String] = [

"0": "000", "1": "001", "2": "010", "3": "011",

"4": "100", "5": "101", "6": "110", "7": "111" ]

var binaryString = ""

// Проходим по каждой цифре восьмеричного числа

for char in octal {

// Проверяем, есть ли соответствие в словаре

guard let binaryValue = octalToBinaryMap[char] else {

throw NumberSystemError.invalidCharacter // Если нет, выбрасываем ошибку

}

binaryString += binaryValue // Добавляем двоичное значение к результату

}

// Удаляем ведущие нули для корректного отображения результата

while binaryString.first == "0" && binaryString.count > 1 {

binaryString.removeFirst()

}

return binaryString // Возвращаем двоичное представление числа

}

// Основная часть программы

print("Введите число: ", terminator: "")

if let input = readLine() {

do {

let binaryNumber = try octalToBinary(octal: input) // Переводим в двоичную систему

print("Octal: \(input) -> Binary: \(binaryNumber)") // Выводим результат

} catch {

print("Ошибка: \(error)") // Обрабатываем возможные ошибки

}

} else { print("Ошибка ввода") }

Результат работы программы представлен на рисунке 1

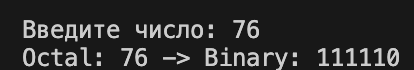


Рисунок 1 – Результат работы программы

**Вариант 10, Задание 2:** Используя функциональные типы, создайте программу согласно варианту. а) для вычитания десятичных дробей; б) для вычитания обыкновенных дробей.

**Текст программы:**

// Функция для вычитания десятичных дробей

func subtractDecimals(\_ a: Double, \_ b: Double) -> Double {

return a - b

}

let decimalResult = subtractDecimals(5.75, 2.30)

print("Результат вычитания десятичных дробей: \(decimalResult)")

// Структура для представления обыкновенной дроби

struct Fraction {

var numerator: Int // Числитель

var denominator: Int // Знаменатель

// Метод для вычитания другой дроби

func subtract(\_ other: Fraction) -> Fraction {

// Приведение дробей к общему знаменателю

let commonDenominator = self.denominator \* other.denominator

let newNumerator = self.numerator \* other.denominator - other.numerator \* self.denominator

// Возвращаем результат в упрощенном виде

return Fraction(numerator: newNumerator, denominator: commonDenominator).simplify()

}

// Метод для упрощения дроби

func simplify() -> Fraction {

let gcdValue = gcd(numerator, denominator)

return Fraction(numerator: numerator / gcdValue, denominator: denominator / gcdValue)

}

// Метод для нахождения наибольшего общего делителя (НОД)

private func gcd(\_ a: Int, \_ b: Int) -> Int {

return b == 0 ? a : gcd(b, a % b)

}

}

let fraction1 = Fraction(numerator: 3, denominator: 4)

let fraction2 = Fraction(numerator: 1, denominator: 2)

let fractionResult = fraction1.subtract(fraction2)

print("Результат вычитания обыкновенных дробей: \(fractionResult.numerator)/\(fractionResult.denominator)")

Результат работы программы представлен на рисунке 2

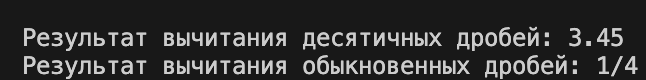


Рисунок 2 – Результат работы программы

**Ответы на вопросы к лабораторной работе:**

* 1. Что такое функция?

Функция – это блок кода, который выполняет определенную задачу, делая его более читаемым и сокращая количества повторов одинаковых операций.

* 1. Как объявить функцию?

Функция объявляется с помощью ключевого слова func, за которым идет имя функции, параметры и тело функции.

* 1. Как объявить функцию с несколькими параметрами?

Для объявления функции с несколькими параметрами, указать их в круглых скобках через запятую.

* 1. Что такое опциональный кортедж?

Опциональный кортеж – это кортеж, который может содержать значения или быть nil. Он используется для возвращения нескольких значений из функции, где одно или несколько значений могут отсутствовать.

* 1. Для чего служат ярлыки аргументов и имен параметров функции?

Ярлыки аргументов и имена параметров помогают сделать вызовы функций более читаемыми. Ярлыки используются при вызове функции, а имена параметров – внутри функции.

* 1. Что такое вариативные параметры?

Вариативные параметры позволяют функции принимать неопределенное количество аргументов одного типа. Они объявляются с помощью … после типа параметров.

* 1. Что такое сквозные параметры?

Сквозные параметры позволяют изменять значение параметра внутри функции и сохранять это изменение за пределами функции. Они объявляются с помощью ключевого слова inout.

* 1. Для чего служат функциональные типы? Как создать свой функциональный тип?

Функциональные типы позволяют использовать функции как значения. Можем передавать функции в качестве аргументов другим функциям или возвращать их. Создать функциональный тип можно, указав типы параметров и возвращаемое значение.

* 1. Что такое вложенные функции?

Вложенные функции – это функции, определенные внутри других функций. Они могут использоваться для организации кода и скрытия вспомогательных функций.

* 1. Что такое замыкание? Для чего оно служит?

Замыкание – это блок кода, который можно передавать и вызывать позже. Замыкания могут захватывать и сохранять ссылки на переменные и константы из окружающего контекста.

* 1. Что такое автозамыкание? Что такое сбегающее замыкание?

Автозамыкание – это замыкание, которое автоматически создается для оборачивания выражения, переданного в качестве аргумента функции; Сбегающее замыкание – это замыкание, которое может быть вызвано после завершения функции, в которой оно было передано. Для этого используются ключевое слово @escaping.

* 1. Как объявить перечисление? Для чего служат перечесления?

Перечисления используются для группировки связанных значений и объявления типов с ограниченным набором возможных значений.

**Вывод**: Изучена работа языка *Swift* и выполнена разработка приложения с использованием функций, замыканий и перечислений.