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: 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**:

**Задание 1**: Дано натуральное число Р. Напишите программу нахождения всех натуральных чисел, не превосходящих Р, которые можно представить в виде произведения двух простых чисел.

**Задание 2**: Используя функциональные типы, создайте программу согласно варианту.

а) для сложения десятичных дробей;

б) для сложения обыкновенных дробей.

**Код программы:**

//1

import Foundation

// Функция для проверки, является ли число простым

func isPrime(\_ number: Int) -> Bool

{

guard number > 1 // оператор guard для проверки, больше ли число number единицы

else

{

return false

}

// Проверяем делимость number на все числа от 2 до квадратного корня из number (включительно)

// Если number делится на любое из этих чисел, оно не является простым и возвращаем false

for i in 2..<Int(sqrt(Double(number))) + 1

{

if number % i == 0

{

return false

}

}

return true

}

// Функция для нахождения всех произведений двух простых чисел

func findProductsOfPrimes(upTo limit: Int) -> [Int]

{

var primes: [Int] = [] // Создаем пустой массив primes для хранения всех простых чисел до limit

var products: Set<Int> = [] // Создаем пустое множество products для хранения всех уникальных произведений простых чисел.

// Находим все простые числа до limit

for i in 2...limit

{

if isPrime(i)

{

primes.append(i) // добавляем число в массив

}

}

// Находим произведения простых чисел

for i in 0..<primes.count

{

for j in i..<primes.count

{

// j начинается с i, чтобы избежать дублирования

let product = primes[i] \* primes[j]

if product <= limit { products.insert(product)

}

else

{

break

// Если произведение превысило limit, выходим из цикла

}

}

}

return Array(products).sorted()

}

// Пример использования

print("Введите число P:")

if let input = readLine(),

let P = Int(input)

{

let result = findProductsOfPrimes(upTo: P)

print("Числа, не превосходящие \(P), которые можно представить в виде произведения двух простых чисел:")

print(result)

}

else {

print("Некорректный ввод. Пожалуйста, введите целое число.")

}

//2.1

import Foundation

// Определяем тип для операции сложения

typealias DecimalAddition = (Double, Double) -> Double

// Реализация функции сложения десятичных дробей

let addDecimals: DecimalAddition = { (a, b) in // замыкание

return a + b

}

// Пример использования

let decimal1 = 3.14

let decimal2 = 2.71

let sumDecimal = addDecimals(decimal1, decimal2)

print("Сумма десятичных дробей: \(sumDecimal)")

//2.2

import Foundation

// Функция для нахождения НОД (наибольший общий делитель)

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

return b == 0 ? abs(a) : greatestCommonDivisor(b, a % b)

}

// Функция для упрощения дроби

func simplifyFraction(numerator: Int, denominator: Int) -> (Int, Int) {

let gcd = greatestCommonDivisor(numerator, denominator)

return (numerator / gcd, denominator / gcd)

}

// Функция для сложения дробей

func addFractions(\_ frac1: (Int, Int), \_ frac2: (Int, Int)) -> (Int, Int) {

let commonDenominator = frac1.1 \* frac2.1 // Знаменатель

let newNumerator = (frac1.0 \* frac2.1) + (frac2.0 \* frac1.1) // Числитель

return simplifyFraction(numerator: newNumerator, denominator: commonDenominator)

}

// Пример использования

let fraction1 = (1, 2) // 1/2

let fraction2 = (1, 3) // 1/3

let sumFraction = addFractions(fraction1, fraction2)

print("Сумма обыкновенных дробей: \(sumFraction.0)/\(sumFraction.1)")

Результат работы программы представлен на рисунке 1.

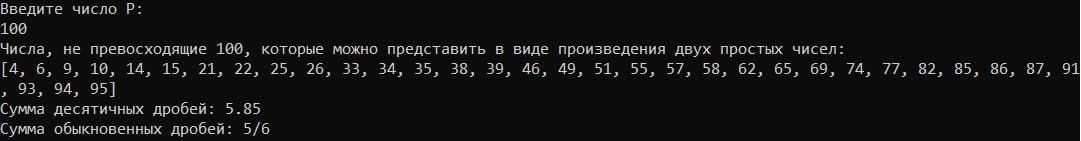


Рисунок 1 – Результат работы программы

**Ответы на вопросы к лабораторной работе:**

1. Что такое функция?

Функция – это блок кода, выполняющий определённую задачу, который можно вызывать многократно.

2. Как объявить функцию?

func имяФункции() {

// код

}

3. Как объявить функцию с несколькими параметрами?

func имяФункции(параметр1: Тип, параметр2: Тип) {

// код

}

4. Что такое опциональный кортеж?

Опциональный кортеж может содержать nil для одного или нескольких значений кортежа, что позволяет обозначить отсутствие значения.

5. Для чего служат ярлыки аргументов и имен параметров функции?

Ярлыки аргументов облегчают читаемость вызова функции, тогда как имена параметров используются внутри самой функции.

6. Что такое вариативные параметры?

Вариативные параметры позволяют функции принимать неопределённое количество значений одного типа:

func имяФункции(параметры: Тип...) {

// код

}

7. Что такое сквозные параметры?

Сквозные параметры передаются дальше в другую функцию или замыкание без изменения.

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

Функциональные типы используются для хранения функций в переменных и передачи их как аргументов. Можно объявить тип функции, указав его параметры и возвращаемый тип:

let функция: (Тип1, Тип2) -> ВозвращаемыйТип

9. Что такое вложенные функции?

Вложенные функции определяются внутри других функций и могут использоваться только внутри них.

10. Что такое замыкание? Для чего оно служит?

Замыкание – это блок кода, который можно сохранить и передать. Используется для выполнения задач по завершению, обработки данных и т.д.

let замыкание = { (параметры) -> Тип в

// код

}

11. Что такое автозамыкание? Что такое сбегающее замыкание?

Автозамыкание – это замыкание, которое автоматически создаётся для выражения, переданного как аргумент функции. Сбегающее замыкание позволяет замыканию существовать после выхода из области видимости функции.

12. Как объявить перечисление? Для чего служат перечисления?

swift

enum ИмяПеречисления {

case значение1

case значение2

}

Перечисления используются для группировки связанных значений и упрощения кода.

**Вывод**: был изучен язык программирования Swift, а именно функции, замыкания и перечисления, и использован для разработки программы.