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: Майданюк А.С.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

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

**Вариант 15**: 1. Дано натуральное число P. Найдите все «совершенные» числа, не превосходящие Р. Совершенным, называется число, сумма делителей которого, не считая самого числа, равна этому числу. Первое совершенное число 6 (6 = 1 +2 + 3).

**Текст программы:**

//1 часть

import Foundation

// Функция для нахождения совершенных чисел, не превосходящих заданное число

func findPerfectNumbers(upTo limit: Int) -> [Int] {

var perfectNumbers = [Int]()

// Перебираем все числа от 2 до предела (limit)

for number in 2...limit {

var sumOfDivisors = 0

// Находим все делители числа, не считая самого числа

for divisor in 1...(number / 2) {

if number % divisor == 0 {

sumOfDivisors += divisor

}

}

// Если сумма делителей равна числу, добавляем его в список совершенных чисел

if sumOfDivisors == number {

perfectNumbers.append(number)

}

}

return perfectNumbers

}

// Основная часть программы

print("Введите натуральное число P:")

if let input = readLine(), let P = Int(input), P > 0 {

let perfectNumbers = findPerfectNumbers(upTo: P)

if perfectNumbers.isEmpty {

print("Совершенных чисел, не превосходящих \(P), не найдено.")

} else {

print("Совершенные числа, не превосходящие \(P): \(perfectNumbers)")

}

} else {

print("Введите корректное натуральное число.")

}

Результат работы программы представлен на рисунке 1

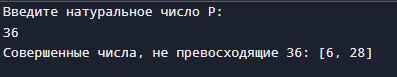


Рисунок 1 – Результат работы программы

2. а) для деления десятичных дробей;

б) для деления обыкновенных дробей.

**Текст программы:**

//2.1 часть

import Foundation

// Функция для деления десятичных дробей

let divideDecimals: (Double, Double) -> Double = { (a, b) in

return a / b

}

// Пример использования

let decimalA: Double = 10.5

let decimalB: Double = 2.5

let decimalResult = divideDecimals(decimalA, decimalB)

print("Результат деления \(decimalA) на \(decimalB) равен \(decimalResult)")

//2.2 часть

import Foundation

// Структура для представления обыкновенной дроби

struct Fraction {

var numerator: Int // числитель

var denominator: Int // знаменатель

// Инициализатор для создания дроби

init(\_ numerator: Int, \_ denominator: Int) {

self.numerator = numerator

self.denominator = denominator

}

// Метод для упрощения дроби

func simplified() -> Fraction {

let gcd = gcd(numerator, denominator)

return Fraction(numerator / gcd, denominator / gcd)

}

// Метод для нахождения наибольшего общего делителя (НОД)

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

if b == 0 {

return a

} else {

return gcd(b, a % b)

}

}

}

// Функция для деления обыкновенных дробей

let divideFractions: (Fraction, Fraction) -> Fraction = { (a, b) in

// Деление дробей: (a/b) / (c/d) = (a \* d) / (b \* c)

let resultNumerator = a.numerator \* b.denominator

let resultDenominator = a.denominator \* b.numerator

return Fraction(resultNumerator, resultDenominator).simplified()

}

// Пример использования

let fractionA = Fraction(3, 4)

let fractionB = Fraction(2, 5)

let fractionResult = divideFractions(fractionA, fractionB)

print("Результат деления \(fractionA.numerator)/\(fractionA.denominator) на \(fractionB.numerator)/\(fractionB.denominator) равен \(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, 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**: Выполнила разработку приложения с использованием языка программирования Swift: функции, замыкания, перечисления.