Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчёт

К лабораторной работе № 3

на тему:

**ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ И ЛЯМБДА-ВЫРАЖЕНИЯ**

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Усенко Ф. В.

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.А. Барбук

(подпись) 214302

Минск 2024

**Вариант 1. Функциональный парсер выражений.**

Условие: Реализуйте программу, которая принимает математическое выражение в виде строки, разбивает его на компоненты и использует лямбда-выражения для его вычисления. Программа должна поддерживать сложные выражения с вложенными скобками.

**Код программы:**

fun main() {

println(“Введите математическое выражение:”)

val expression = readLine()!!

Try {

val result = evaluateExpression(expression)

println(“Результат: $result”)

} catch (e: Exception) {

println(“Ошибка: ${e.message}”)

}

}

fun evaluateExpression(expression: String): Double {

// Удаляем все пробелы из выражения

val cleanedExpression = expression.replace(“\\s”.toRegex(), “”)

// Парсим выражение в список токенов

val tokens = tokenize(cleanedExpression)

// Вычисляем результат

return parse(tokens).first()

}

fun tokenize(expression: String): List<String> {

val regex = “([\\d](file://d)+(\\.\\d\*)?|[+\\-\*/()^])”.toRegex()

return regex.findAll(expression).map { it.value }.toList()

}

fun parse(tokens: List<String>): List<Double> {

val values = mutableListOf<Double>()

val ops = mutableListOf<String>()

var i = 0

while (i < tokens.size) {

val token = tokens[i]

when {

token.isNumber() -> {

values.add(token.toDouble())

}

token == “(“ -> {

ops.add(token)

}

token == “)” -> {

while (ops.isNotEmpty() && ops.last() != “(“) {

applyOperator(ops.removeAt(ops.size – 1), values)

}

ops.removeAt(ops.size – 1) // Убираем открывающую скобку

}

token in listOf(“+”, “-“, “\*”, “/”, “^”) -> {

while (ops.isNotEmpty() && precedence(ops.last()) >= precedence(token)) {

applyOperator(ops.removeAt(ops.size – 1), values)

}

ops.add(token)

}

}

i++

}

while (ops.isNotEmpty()) {

applyOperator(ops.removeAt(ops.size – 1), values)

}

return values

}

fun String.isNumber() = this.toDoubleOrNull() != null

fun precedence(op: String): Int {

return when (op) {

“+”, “-“ -> 1

“\*”, “/” -> 2

“^” -> 3

else -> 0

}

}

fun applyOperator(operator: String, values: MutableList<Double>) {

val right = values.removeAt(values.size – 1)

val left = values.removeAt(values.size – 1)

val result = when (operator) {

“+” -> left + right

“-“ -> left – right

“\*” -> left \* right

“/” -> left / right

“^” -> Math.pow(left, right)

else -> throw IllegalArgumentException(“Неподдерживаемая операция: $operator”)

}

values.add(result)

}

**Контрольные вопросы:**

**Контрольные вопросы**

1. Как объявить функцию в *Kotlin*? В чем разница между обычной функцией и однострочной функцией?

// Обычная функция

fun greet(name: String): String {

return "Hello, $name!"

}

// Однострочная функция

fun greet(name: String): String = "Hello, $name!"

1. Что такое функция высшего порядка, и как её использовать? Приведите пример.

– Функция высшего порядка принимает другие функции в качестве параметров или возвращает функции.

1. Как передать функцию в качестве параметра другой функции? Приведите пример.

fun operate(x: Int, y: Int, operation: (Int, Int) -> Int): Int {

return operation(x, y)

}

val sum = operate(4, 2) { a, b -> a + b }

1. Чем отличается лямбда-выражение от обычной функции?

// Обычная функция

fun multiply(x: Int, y: Int): Int = x \* y

// Лямбда-выражение

val multiply: (Int, Int) -> Int = { x, y -> x \* y }

1. Как создать лямбда-выражение с двумя параметрами? Приведите пример.

val add: (Int, Int) -> Int = { x, y -> x + y }

1. Что такое анонимная функция, и как она отличается от лямбда-выражения?

// Анонимная функция

val sum = fun(x: Int, y: Int): Int = x + y

// Лямбда-выражение

val sumLambda: (Int, Int) -> Int = { x, y -> x + y }

1. Какие возможности предоставляют анонимные функции в Kotlin?

– Анонимные функции позволяют писать функции на месте, могут быть полезны при необходимости в сложных вычислениях внутри другой функции и могут использовать return для выхода из анонимной функции.

1. Что такое замыкание, и как оно работает? Приведите пример использования.

– Замыкание — это функция, которая "захватывает" переменные из внешней области видимости.

fun counter(): () -> Int {

var count = 0

return { ++count }

}

val myCounter = counter()

println(myCounter()) // 1

println(myCounter()) // 2

1. Как лямбда-выражения могут захватывать переменные из внешней области видимости?

var factor = 2

val multiply: (Int) -> Int = { number -> number \* factor }

println(multiply(3)) // 6

1. В чем преимущества использования функций высшего порядка в программировании?

– Функции высшего порядка позволяют писать более гибкий и переиспользуемый код, упрощают работу с функциональными операциями, такими как *map*, *filter* и *reduce*.

1. Как реализовать функцию *map*, используя лямбда-выражения? Приведите пример.

val numbers = listOf(1, 2, 3, 4, 5)

val doubled = numbers.map { it \* 2 }

println(doubled) // [2, 4, 6, 8, 10]

1. Как функция *reduce* работает с массивами? Приведите пример использования.

val numbers = listOf(1, 2, 3, 4, 5)

val sum = numbers.reduce { acc, number -> acc + number }

println(sum) // 15

1. Как работает функция *filter*, и как её использовать для фильтрации списка строк?

val words = listOf("Kotlin", "Java", "JavaScript", "Python")

val filteredWords = words.filter { it.startsWith("J") }

println(filteredWords) // [Java, JavaScript]

1. Как реализовать функцию *forEach* для выполнения действий над каждым элементом массива?

val numbers = listOf(1, 2, 3, 4, 5)

numbers.forEach { println(it) }

1. Что такое функция partition, и как она может быть полезна?

val numbers = listOf(1, 2, 3, 4, 5)

val (even, odd) = numbers.partition { it % 2 == 0 }

println(even) // [2, 4]

println(odd) // [1, 3, 5]

1. Как использовать лямбда-выражения для сортировки списка по произвольному критерию?

val people = listOf("Alice", "Bob", "Charlie")

val sortedByLength = people.sortedBy { it.length }

println(sortedByLength) // [Bob, Alice, Charlie]

1. Как создать замыкание, которое увеличивает значение переменной-счетчика при каждом вызове?

fun makeCounter(): () -> Int {

var count = 0

return { count++ }

}

val counter = makeCounter()

println(counter()) // 0

println(counter()) // 1

1. Как в Kotlin реализовать обработку ошибок при помощи анонимных функций?

fun divide(a: Int, b: Int): Int? {

return try {

a / b

} catch (e: ArithmeticException) {

null

}

}

1. Чем полезны функции высшего порядка в создании гибких и расширяемых программ?

– Функции высшего порядка позволяют разделять логику на небольшие, переиспользуемые блоки и легко адаптировать и расширять программы.

1. Как использовать лямбда-выражения для форматирования строк? Приведите пример.

val names = listOf("John", "Jane", "Doe")

val formattedNames = names.map { name -> name.uppercase() }

println(formattedNames) // [JOHN, JANE, DOE]