Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

Информатики и радиоэлектроники»

Учебная дисциплина: «Программирование мобильных информационных систем»

Отчет

по выполнению лабораторной работы «Основы языка Kotlin»

Выполнил студент: гр. 210101 Войтович Е.А.

Проверил: Усенко Ф.В.

Минск 2024

**1. Цель работы:** изучить основные конструкции языка Kotlin, научиться работать с типами данных, операциями ввода-вывода, а также условными операторами. Закрепить полученные знания через выполнение задач, требующих практического применения теории.

**2. Задание:** Напишите программу, которая позволяет пользователю вводить выражения с переменными (например, "x + y \* 2") и сохранять переменные для последующего использования. Программа должна поддерживать операции присваивания и выводить результат вычисления.

**3. Оснащение работы:** техническое задание, технический проект, ЭВМ, Kotlin, Itelij IDEA.

import java.util.\*

import kotlin.math.pow

fun main() {

val variables = mutableMapOf<String, Double>()

val scanner = Scanner(System.`in`)

while (true) {

print("Введите выражение, в случае если выражение сложно используйте скобки \"()\" (или 'exit' для выхода): ")

val input = scanner.nextLine().trim()

if (input.equals("exit", ignoreCase = true)) {

println("Программа завершена.")

break

}

try {

if (input.contains("=")) {

val parts = input.split("=").map { it.trim() }

if (parts.size == 2) {

val variableName = parts[0]

val expression = parts[1]

val result = calculateExpression(expression, variables, scanner)

variables[variableName] = result

println("$variableName = $result")

} else {

println("Неправильное выражение присваивания.")

}

} else {

val result = calculateExpression(input, variables, scanner)

println("Результат: $result")

}

} catch (e: Exception) {

println("Ошибка: ${e.message}")

}

}

}

fun calculateExpression(expression: String, variables: MutableMap<String, Double>, scanner: Scanner): Double {

val tokens = tokenize(expression)

val postfix = convertToPostfix(tokens, variables, scanner)

return evaluatePostfix(postfix)

}

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

val regex = Regex("""(\d+(\.\d+)?)|([a-zA-Z\_]\w\*)|[+\*/^()-]""")

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

}

fun convertToPostfix(tokens: List<String>, variables: MutableMap<String, Double>, scanner: Scanner): List<String> {

val precedence = mapOf("+" to 1, "-" to 1, "\*" to 2, "/" to 2, "^" to 3)

val output = mutableListOf<String>()

val operators = Stack<String>()

for (token in tokens) {

when {

token.matches(Regex("\\d+(\\.\\d+)?")) -> output.add(token) // число

token.matches(Regex("[a-zA-Z\_]\\w\*")) -> {

if (!variables.containsKey(token)) {

print("Введите значение для переменной $token: ")

val value = scanner.nextLine().toDoubleOrNull() ?: throw IllegalArgumentException("Некорректное значение для переменной $token")

variables[token] = value

}

output.add(variables[token].toString())

}

token == "(" -> operators.push(token) // Открывающая скобка

token == ")" -> { // Закрывающая скобка

while (operators.isNotEmpty() && operators.peek() != "(") {

output.add(operators.pop())

}

if (operators.isNotEmpty() && operators.peek() == "(") {

operators.pop()

}

}

token in precedence -> { // оператор

while (operators.isNotEmpty() && (precedence[operators.peek()] ?: 0) >= precedence[token]!!) {

output.add(operators.pop())

}

operators.push(token)

}

else -> throw IllegalArgumentException("Неизвестный токен: $token")

}

}

while (operators.isNotEmpty()) {

output.add(operators.pop())

}

return output

}

fun evaluatePostfix(postfix: List<String>): Double {

val stack = Stack<Double>()

for (token in postfix) {

when {

token.matches(Regex("\\d+(\\.\\d+)?")) -> stack.push(token.toDouble()) // число

token == "+" -> stack.push(stack.pop() + stack.pop())

token == "-" -> {

val b = stack.pop()

val a = stack.pop()

stack.push(a - b)

}

token == "\*" -> stack.push(stack.pop() \* stack.pop())

token == "/" -> {

val b = stack.pop()

val a = stack.pop()

stack.push(a / b)

}

token == "^" -> {

val b = stack.pop()

val a = stack.pop()

stack.push(a.pow(b))

}

}

}

return stack.pop()

}

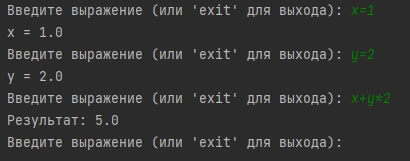


Рисунок 1 – Результат исполнения кода программы

**Вывод:** в ходе выполнения работы были изучены основные конструкции языка Kotlin, такие как типы данных, операции ввода-вывода, условные операторы, а также работа с переменными и операциями присваивания.