Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчёт

по лабораторной работе №3

на тему:

**ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ И ЛЯМБДА-ВЫРАЖЕНИЯ**

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

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Зинковская

(подпись) гр. 214301

Минск, 2024

**Цель:** Реализовать программу, которая использует функции высшего порядка для работы с динамическими структурами данных, такими как очереди и стеки. Программа должна поддерживать разные режимы обработки данных.

**Код**

fun main() {

val stack = Stack<Int>()

val queue = Queue<Int>()

while (true) {

println(

"""

Меню:

1. Стек

2. Очередь

3. Выход

""".trimIndent()

)

var operation = readlnOrNull()?.toIntOrNull()

when (operation){

1-> steckOperations(stack)

2-> quequeOperations(queue)

else -> return

}

}

}

fun steckOperations(stack: Stack<Int>){

while (true){

println(

"""

Меню:

1. Добавить элемент в стек

2. Удалить элемент из стека

3. Просмотреть верхний элемент стека

4. Просмотреть стек

5. Выход

""".trimIndent()

)

var operation = readlnOrNull()?.toIntOrNull()

when (operation){

1-> {

println("Введите элемент для добавления:")

var element = readlnOrNull()?.toIntOrNull()

stack.push(element!!)

}

2-> {

println("Удаление элемента из стека")

stack.pop()

}

3-> {

var element = stack.peek()

println("Просмотр верхнего элемента стека $element")

}

4-> stack.printStack()

else -> return

}

}

}

fun quequeOperations(queue: Queue<Int>) {

while (true){

println(

"""

Меню:

1. Добавить элемент в очередь

2. Удалить элемент из очереди

3. Просмотреть первый элемент очереди

4. Просмотреть очередь

5. Выход

""".trimIndent()

)

var operation = readlnOrNull()?.toIntOrNull()

when (operation){

1-> {

println("Введите элемент для добавления:")

var element = readlnOrNull()?.toIntOrNull()

queue.enqueue(element!!)

}

2-> {

println("Удаление элемента из очереди")

queue.dequeue()

}

3-> {

var element = queue.peekQueue()

println("Просмотр верхнего элемента очереди $element")

}

4-> queue.printQueue()

else -> return

}

}

}

class Queue<T> {

private val queue = mutableListOf<T>()

private fun processQueue(operation: (MutableList<T>) -> T?):T? {

return operation(queue)

}

fun enqueue(element: T) {

processQueue {

it.add(element)

null

}

}

fun dequeue(): T? {

return if (queue.isNotEmpty()) {

val firstElement = queue.first()

processQueue { it.removeAt(0) }

firstElement

} else null

}

fun peekQueue(): T? {

return processQueue { it.firstOrNull() }

}

fun printQueue() {

println("Очередь: $queue")

}

}

class Stack<T> {

private val stack = mutableListOf<T>()

private fun processStack(operation: (MutableList<T>) -> T?): T? {

return operation(stack)

}

fun push(element: T) {

processStack {

it.add(element)

null

}

}

fun pop(): T? {

return if (stack.isNotEmpty()) {

processStack {

it.removeAt(it.size - 1)

stack.lastOrNull()

}

} else null

}

fun peek(): T? {

return processStack { it.lastOrNull() }

}

fun printStack() {

println("Стек: $stack")

}

}

**Контрольные вопросы**

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

Функции в Kotlin объявляются с использованием ключевого слова fun, после которого следует имя функции, параметры и возвращаемый тип. Если функция не возвращает значение, можно не указывать тип возвращаемого значения (по умолчанию используется Unit).

Если функция состоит из одного выражения, её можно объявить как однострочную, без использования return. While выполняется пока истинно условие, а for заданное количество раз

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

Анонимные функции похожи на лямбда-выражения, но могут иметь более сложное тело, включая несколько операторов. Анонимные функции определяются с использованием ключевого слова fun без имени функции.

В отличие от лямбда-выражений, анонимные функции могут содержать в себе несколько инструкций и поддерживают доступ к меткам возврата (return).

**Вывод**: Были изучены анонимные функции и лямбда-выражения. Написана программа, работающая со стеком и очередью.