Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

Выполнил студент группы КС-36 Меркушов Даниил Игоревич

Ссылка на репозиторий: https://github.com/VooDooCry/Algorithms\_KS36.git

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: 17.02.2025

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

Необходимо реализовать алгоритм очереди (Очередь через односвязный список и очередь через два стека (стек из библиотеки)

# Описание метода/модели.

# Очереди очень похожи на стеки. Они также не дают доступа к произвольному элементу, но, в отличие от стека, элементы кладутся (enqueue) и забираются (dequeue) с разных концов. Такой метод называется «первый вошел, первый вышел» (First-In-First-Out или FIFO). То есть забирать элементы из очереди мы будем в том же порядке, что и клали. Как реальная очередь или конвейер.

# Выполнение задачи.

Язык программирования: **Golang**

Входные данные / тесты: test1, test2, test3

# Структуры данных:

type Node[T any] struct { Узел для связного списка

data T Значение элемента

next \*Node[T] Ссылка на следующий узел

}

type Queue[T any] struct { Очередь на связном списке

head \*Node[T] Первый элемент

tail \*Node[T] Последний элемент

size int Текущий размер

}

type Queue2[T any] struct { Очередь на двух стеках

stackIn []T Стек для добавления элементов

stackOut []T Стек для извлечения элементов

}

Реализация методов для Queue (связный список)

Push(value T) Добавляет элемент в конец через tail

Pop() Удаляет элемент из начала через head

Front() Возвращает head.data

IsEmpty() Проверяет size == 0

GetSize() Возвращает size

Elements() Собирает все элементы в слайс

**Реализация методов для Queue2 (два стека)**

shiftStacks() Переносит элементы из stackIn в stackOut

Push() Добавляет в stackIn

Pop() Удаляет из stackOut после shiftStacks()

Front() Возвращает из stackOut после shiftStacks()

Elements() Объединяет stackOut (в обратном порядке) + stackIn

# Заключение.

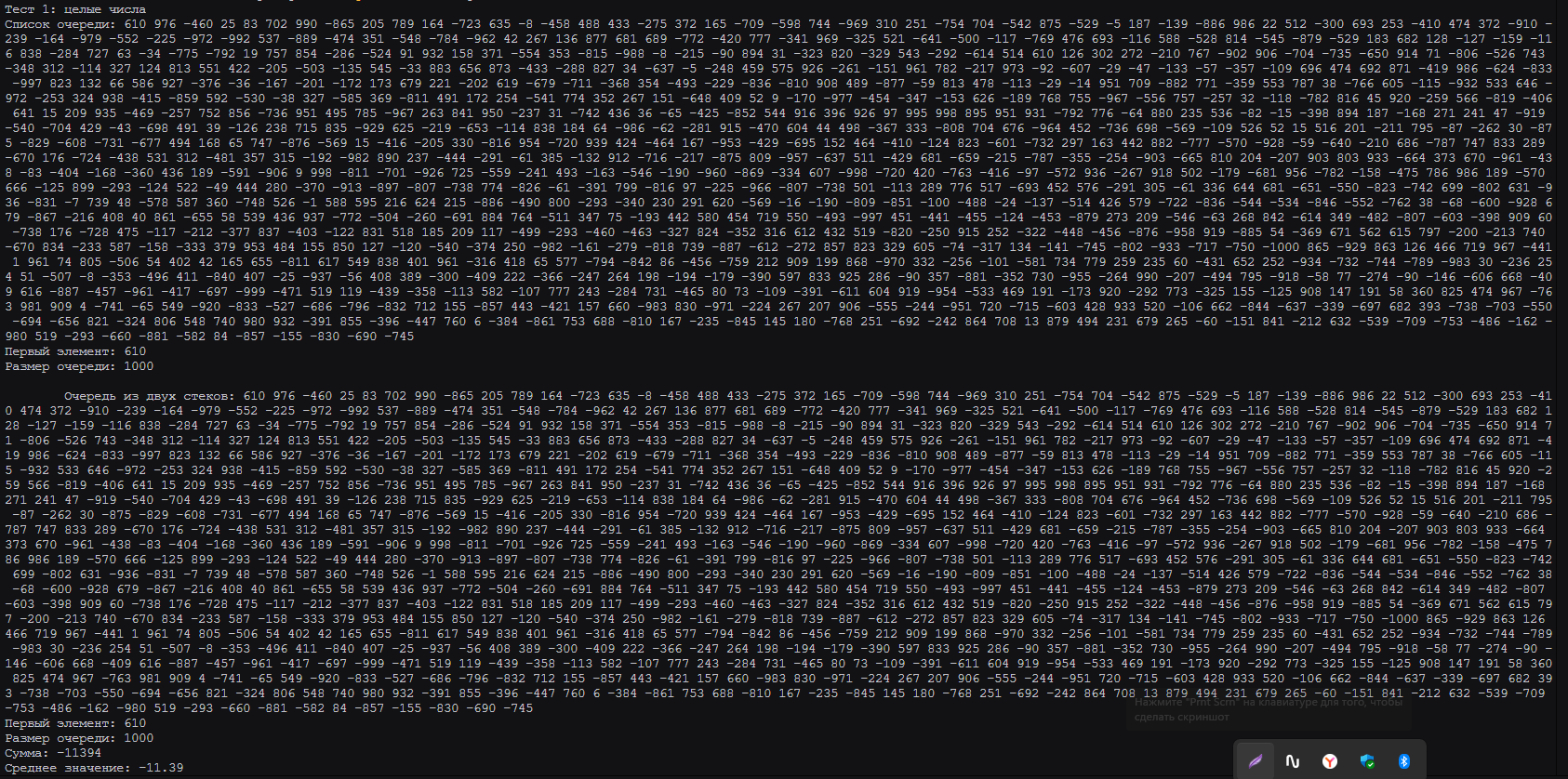
**Основные выводы:**

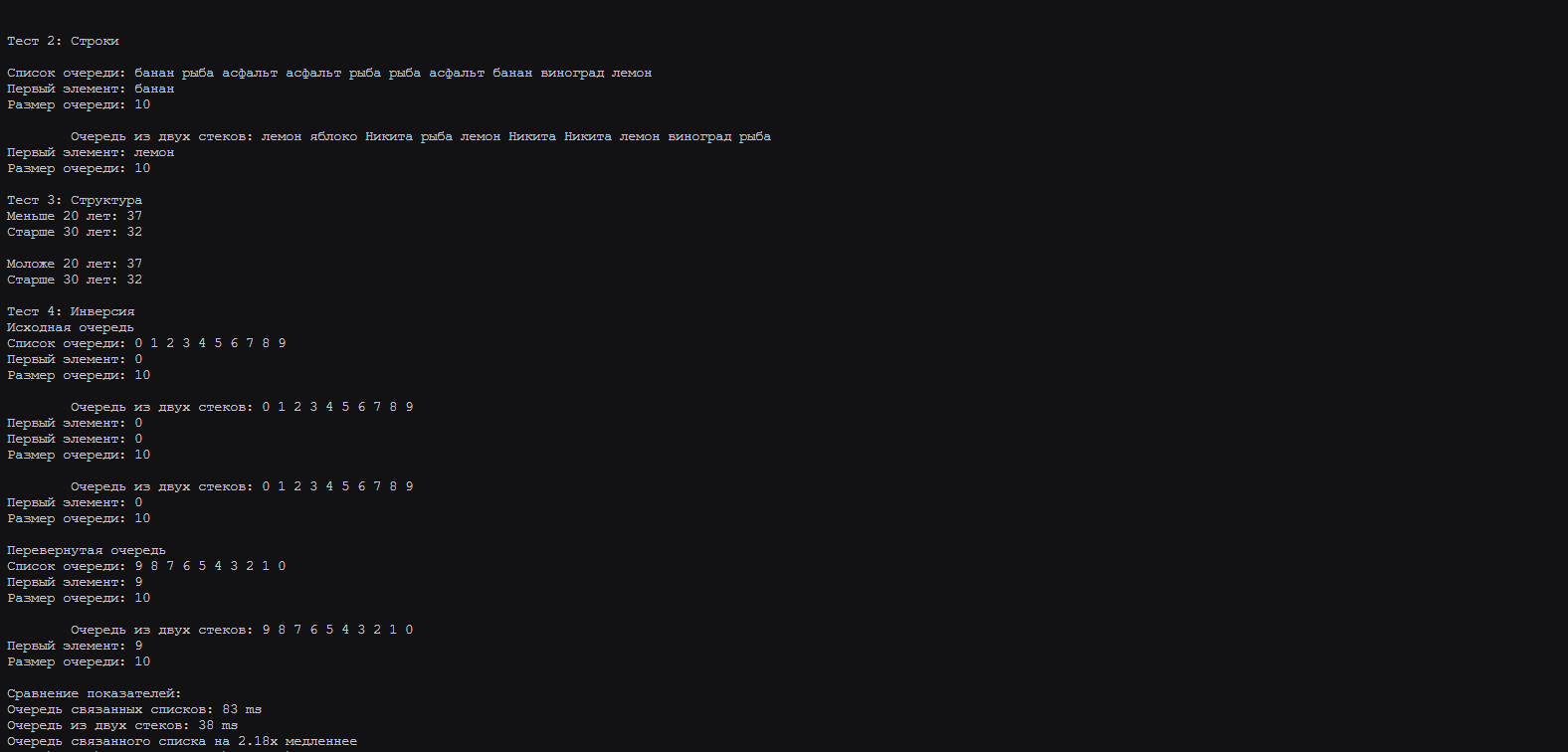
1. Очередь поддерживает две основные операции: добавление элемента (enqueue) и удаление элемента (dequeue). Эти операции имеют амортизированное время выполнения O(1).
2. Очереди широко используются в различных областях, включая обработку задач в многозадачных системах, управление печатью, сетевую передачу данных и алгоритмы поиска.
3. Очереди могут быть реализованы с использованием массивов или связанных списков, причем каждая из реализаций имеет свои преимущества и недостатки в зависимости от сценария использования.

**Различия:**

Стек и очередь используют однонаправленные ссылки для отслеживания элементов (хотя база может быть реализована с помощью массива или связного списка), а двусвязный список имеет связи в обе стороны.  
Производительность:  
  
Очереди и стеки имеют фиксированный доступ к элементам в зависимости от их структуры, в то время как в двусвязных списках можно быстро добавлять и удалять элементы из любого места.  
  
  
Стек и очередь имеют строгую дисциплину по добавлению и удалению, в то время как двусвязный список более свободен

**Тесты**

****

****