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 Гакиев Артур Лечиевич

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Дата сдачи: 28.03.2022

Оглавление

[Описание задачи. 2](#_Toc63548272)

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

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

**Описание задачи.**

В рамках лабораторной работы необходимо изучить и реализовать одну из трёх структур очередь, в соответствии со своим вариантом, при этом, все структуры должны:

* Использовать шаблонный подход, обеспечивая работу контейнера с произвольными данными.
* Реализовывать свой итератор с реализацией операторов ++ и !=
* Обеспечивать работу стандартных библиотек и конструкции for each если она есть в языке.
* Проверку на пустоту и подсчет количества элементов.
* Операцию сортировки с использованием стандартной библиотеки

Очередь, операции:

* добавление в конец
* взятие с начала

**Описание метода/модели**

Очередь – это набор объектов, который поддерживает быструю семантику first-in, first-out (FIFO) для вставки и удаления. Операции вставки и удаления иногда называют enqueue и dequeuer. В отличие от списков и массивов, очереди, как правило, не пропускают случайный доступ к содержащимся объектам.

Удаление объектов очереди происходит по принципу первый вошел – первым вышел, то есть удаляется объект, который был добавлен недавно.

**Выполнение задачи.**

Реализация очереди выполнена на языке Python 3.10. Были реализованы классы Node – класс нодов для реализации самой очереди. Класс queue – сама реализованная очередь.

**Заключение.**

Очередь используется, когда нужно совершить какие-то действия в порядке их поступления. Данная реализация очереди является одной из двух возможных – также есть опция реализации через один массив, что также возможно, но сложнее в связи с необходимостью использования шаблонного подхода. В отличии от уже реализованного решения, такой метод бы позволил уменьшить затраты памяти, но потребовал бы многократно переопределять массив с последующим перемещением всех элементов в новый массив, что бы вызвало огромные потери во времени.