**Министерство науки и высшего образования Российской Федерации** ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий Дисциплина:**

«Алгоритмы и структуры данных»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

«Сортировка массива с использованием очереди»

**Выполнил:**

Чу Ван Доан N3247



(подпись)

**Проверил:**

Ерофеев С. А.



(подпись)

Санкт-Петербург 2024 г.

# СОДЕРЖАНИЕ

[Введение 4](#_heading=h.1fob9te)

1. [Описание функционала программы 5](#_heading=h.2et92p0)
2. [Блок-схема 6](#_heading=h.3dy6vkm)
3. [Код программы 7](#_heading=h.4d34og8)
4. [Результаты тестирования 8](#_heading=h.17dp8vu)

[Заключение 10](#_heading=h.26in1rg)

# ВВЕДЕНИЕ

Задача работы – Разработать программу шейкерной сортировки, использует кольцевую очередь на базе массива.

"Queue" (очередь) - это структура данных, которая представляет собой упорядоченную коллекцию элементов, где элементы добавляются в конец очереди и удаляются из начала. Это типичная структура данных, используемая для реализации FIFO (First In, First Out) - первым пришел, первым вышел. Это означает, что элементы обрабатываются в порядке их добавления: первый добавленный элемент будет первым, который будет удален.

Для реализации был выбран язык программирования C++.

# ОПИСАНИЕ ФУНКЦИОНАЛА ПРОГРАММЫ

Эта программа считывает массив чисел из файла "data.txt", сортирует его с использованием очереди (queue), а затем выводит отсортированный массив.

1. Функция sortArrayWithQueue: Эта функция принимает на вход ссылку на вектор целых чисел и сортирует его, используя очередь. Она работает следующим образом:

- Создает пустую очередь q для хранения чисел из массива.

- Добавляет все числа из входного массива в очередь.

- Использует алгоритм сортировки выбором, который заключается в том, чтобы на каждом шаге извлекать наименьший элемент из очереди и вставлять его в конец массива.

- После завершения сортировки, исходный массив будет отсортирован в порядке возрастания.

2. Функция main: В главной функции происходит следующее:

- Создается пустой вектор arr для хранения чисел из файла.

- Файл "data.txt" открывается для чтения, и числа из него добавляются в вектор arr.

- После того как все числа считаны, файл закрывается.

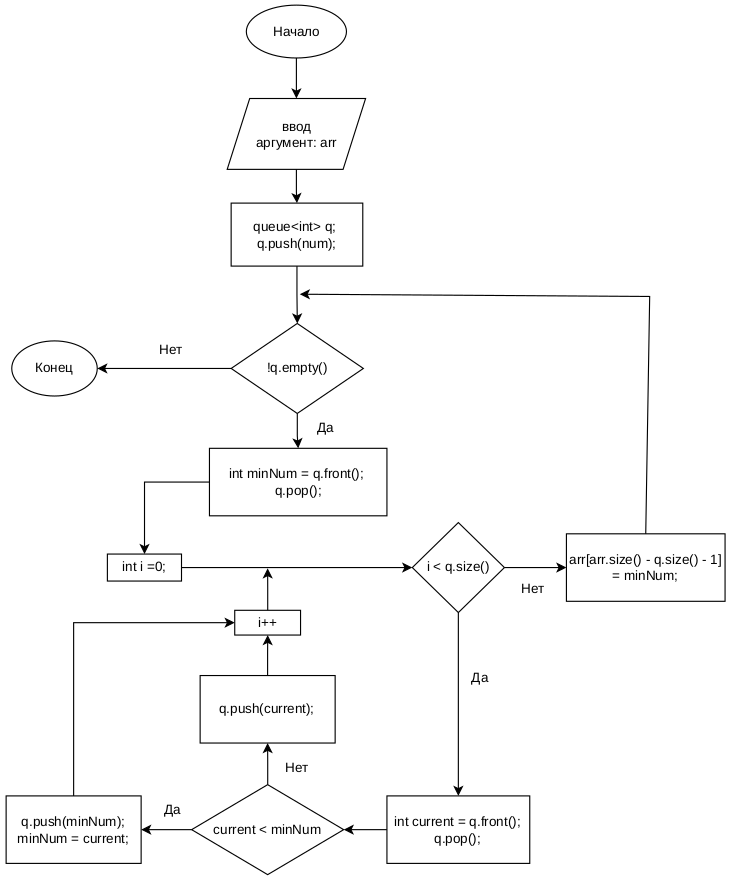
- Вызывается функция sortArrayWithQueue, чтобы отсортировать массив.

- Отсортированные числа выводятся на экран.

3. Файл данных "data.txt": Программа предполагает, что в файле "data.txt" числа разделены пробелами или символами новой строки.

Таким образом, программа считывает, сортирует и выводит массив чисел из файла "data.txt", используя алгоритм сортировки выбором и структуру данных очередь.

# БЛОК-СХЕМА

****

**Рисунок 1 – Блок-схема сортировки очереди**

# КОД ПРОГРАММЫ

#include <iostream>

#include <fstream>

#include <queue>

#include <vector>

using namespace std;

// Функция сортирует числовой массив, переключаясь в очередь

void sortArrayWithQueue(vector<int>& arr) {

// Создайте очередь для хранения чисел из массива

queue<int> q;

// Добавить числа из массива в очередь

for (int num : arr) {

q.push(num);

}

// Сортировка чисел в очереди

while (!q.empty()) {

// Получить наименьший элемент из очереди

int minNum = q.front();

q.pop();

// Просмотрите остальную часть очереди, чтобы найти наименьшее число.

for (int i = 0; i < q.size(); i++) {

int current = q.front();

q.pop();

// Если найдено меньшее число, поместите меньшее число в очередь и обновите наименьшее число.

if (current < minNum) {

q.push(minNum);

minNum = current;

} else {

q.push(current);

}

}

// Вставьте наименьшее число в массив

arr[arr.size() - q.size() - 1] = minNum;

}

}

int main() {

vector<int> arr;

// Откройте файл для чтения

ifstream inFile("data.txt");

if (!inFile) {

cerr << "Не может открыть файл!" << endl;

return 1;

}

// Прочитать числа из файла и добавить в массив

int num;

while (inFile >> num) {

arr.push\_back(num);

}

// Закрыть файл

inFile.close();

// Сортировка массива с помощью очереди

sortArrayWithQueue(arr);

// Распечатать массив после сортировки

cout << "Массив после сортировки: ";

for (int num : arr) {

cout << num << " ";

}

cout << endl;

return 0;

}

# РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

****

**Рисунок 2 – В файл «data.txt» вводится массив чисел.**

****

**Рисунок 3 – Массив чисел из файла после сортировки.**

# ЗАКЛЮЧЕНИЕ

В этой лабораторной работе я разработал и реализовал алгоритм сортировки числового массива с использованием алгоритма очереди.

В представленной программе был рассмотрен пример использования очереди (queue) для сортировки массива чисел. Этот подход демонстрирует применение структуры данных для упорядочивания элементов, предоставляя эффективный метод сортировки.

Преимуществом использования очереди для сортировки является её простота в реализации и понимании. Алгоритм сортировки выбором, применённый здесь, хотя и не является самым эффективным в больших массивах данных, всё же подходит для небольших наборов чисел и иллюстрирует базовые принципы сортировки.

Однако следует отметить, что при работе с большими объемами данных лучше использовать более эффективные алгоритмы сортировки, такие как быстрая сортировка или сортировка слиянием, для обеспечения более быстрой работы программы.

Несмотря на это, представленный пример демонстрирует основные концепции сортировки с использованием очереди, что может быть полезным для новичков в программировании, а также для понимания основных принципов алгоритмов сортировки.