**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 №3**

«Шейкерная сортировка с использованием обычной очереди на базе связного списка»

**Выполнила:**

Чу Ван Доан, студент группы N3247

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Ерофеев С. А., преподаватель ФБИТ

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(отметка о выполнении)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1 ОПИСАНИЕ ФУНКЦИОНАЛА ПРОГРАММЫ 4

2 БЛОК-СХЕМА 6

3 КОД ПРОГРАММЫ 7

4 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 11

ЗАКЛЮЧЕНИЕ 12

**ВВЕДЕНИЕ**

Задача работы – Разработать программу шейкерной сортировки, использует обычную очередь на базе связного списка.

Шейкерная сортировка (Shaker Sort) является алгоритмом сортировки, который использует подход, аналогичный пузырьковой сортировке.

Обычная очередь на базе связного списка - это структура данных, которая использует связанный список для реализации операций добавления элемента в конец очереди (Enqueue) и удаления элемента из начала очереди (Dequeue).

Для реализации был выбран язык программирования C++.

# ОПИСАНИЕ ФУНКЦИОНАЛА ПРОГРАММЫ

Вот описание алгоритма Shakersort, использует обычную очередь на базе связного списка:

1. Создается класс Queue для реализации очереди на базе связного списка.
2. В функции shakerSort создаются две очереди: queue и sorted\_queue, а также объявляются переменные swapped, start, end, first\_element и second\_element.
3. Выполняется первый проход слева направо:

* Каждый элемент массива arr добавляется в очередь queue.
* Извлекается первый элемент из очереди queue и сохраняется в first\_element.
* Пока очередь queue не пуста:
* Извлекается следующий элемент из очереди queue и сохраняется в second\_element.
* Если first\_element больше second\_element, значит элементы нужно поменять местами:
* Устанавливается флаг swapped в значение true.
* Второй элемент добавляется в очередь sorted\_queue.
* Из очереди queue извлекается следующий элемент и сохраняется в second\_element.
* Иначе, первый элемент добавляется в очередь sorted\_queue, а second\_element становится новым first\_element.
* Последний first\_element добавляется в очередь sorted\_queue.
* Элементы из sorted\_queue записываются обратно в массив arr для текущего диапазона.

1. Выполняется второй проход справа налево:

* Каждый элемент массива arr добавляется в очередь queue.
* Извлекается первый элемент из очереди queue и сохраняется в first\_element.
* Пока очередь queue не пуста:
* Извлекается следующий элемент из очереди queue и сохраняется в second\_element.
* Если first\_element меньше second\_element, значит элементы нужно поменять местами:
* Устанавливается флаг swapped в значение true.
* Второй элемент добавляется в очередь sorted\_queue.
* Из очереди queue извлекается следующий элемент и сохраняется в second\_element.
* Иначе, первый элемент добавляется в очередь sorted\_queue, а second\_element становится новым first\_element.
* Последний first\_element добавляется в очередь sorted\_queue.
* Элементы из sorted\_queue записываются обратно в массив arr для текущего диапазона.

1. Границы start и end обновляются для следующего прохода.
2. Шаги 3-5 повторяются, пока в процессе сортировки происходят обмены элементов (swapped равно true).

* учитывается диапазон массива от -1,7 \* 10^308 до 1,7 \* 10^308.

В функции main считывается входной массив из файла input.txt, выполняется шейкерная сортировка с использованием функции shakerSort, а затем отсортированный массив выводится на экран.

Программа использует очередь на базе связного списка для временного хранения элементов во время сортировки. Это позволяет избежать лишних переносов элементов при выполнении перестановок в шейкерной сортировке.

# БЛОК-СХЕМА

Изображение выглядит как текст, диаграмма, Шрифт, снимок экрана

Автоматически созданное описание

1. Блок-схема алгоритма Shakersort

# КОД ПРОГРАММЫ

#include <iostream>

#include <fstream>

using namespace std;

struct Node {

    int data;

    Node \*next;

};

class Queue {

private:

    Node \*head, \*tail;

    int \_size;

public:

    Queue(){

        head = NULL;

        tail = NULL;

        \_size = 0;

    }

    bool empty() {

        return head == NULL;

    }

    void push(int data){

        Node\* p = new Node;

        p->data = data;

        p->next = NULL;

        if (head == NULL){

            head = p;

            tail = p;

        }

        else {

            tail->next = p;

            tail = p;

        }

        \_size++;

    }

    void pop(){

        if(empty()) return;

        int value = head->data;

        Node\* temp = head;

        head = head->next;

        delete temp;

        \_size--;

    }

    int front(){

        return head->data;

    }

    int size(){

        return \_size;

    }

    void clear(){

        while (!empty()){

            pop();

        }

    }

};

void shakerSort(int arr[], int n) {

    Queue queue, sorted\_queue;

    bool swapped = true;

    int start = 0;

    int end = n;

    int first\_element;

    int second\_element;

    while (swapped) {

        swapped = false;

        // Проход слева направо

        for (int i = start; i < end; i++) {

            queue.push(arr[i]);

        }

        first\_element = queue.front();

        queue.pop();

        while (!queue.empty()){

            second\_element = queue.front();

            if (first\_element > second\_element){

                swapped = true;

                sorted\_queue.push(second\_element);

                second\_element = queue.front();

                queue.pop();

            }

            else {

                sorted\_queue.push(first\_element);

                first\_element = second\_element;

                second\_element = queue.front();

                queue.pop();

            }

        }

        sorted\_queue.push(first\_element);

        for (int i = start; i < end; i++) {

            arr[i] = sorted\_queue.front();

            sorted\_queue.pop();

        }

        // Проход справа налево

        for (int i = end - 1; i >= start; i--) {

            queue.push(arr[i]);

        }

        first\_element = queue.front();

        queue.pop();

        while (!queue.empty()){

            second\_element = queue.front();

            if (first\_element < second\_element){

                swapped = true;

                sorted\_queue.push(second\_element);

                second\_element = queue.front();

                queue.pop();

            }

            else {

                sorted\_queue.push(first\_element);

                first\_element = second\_element;

                second\_element = queue.front();

                queue.pop();

            }

        }

        sorted\_queue.push(first\_element);

        for (int i = end - 1; i >= start; i--) {

            arr[i] = sorted\_queue.front();

            sorted\_queue.pop();

        }

        // Обновление границ

        start++;

        end--;

    }

}

int main() {

    ifstream file("input.txt");

    if (!file.is\_open()) {

        cout << "The file cannot be opened for reading." << endl;

        return -1;

    }

    int n; file >> n;

    int \*arr = new int[n];

    for (int i = 0; i < n; i++){

        file >> arr[i];

    }

    file.close();

    shakerSort(arr, n);

    cout << "Sorted array: ";

    for (int i = 0; i < n; i++){

        cout << arr[i] << " ";

    }

    cout <<  endl;

    return 0;

}

# РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

1. Массив чисел вводится в программу.



1. После сортировки массив чисел с использованием обычной очереди на базе связного списка с помощью алгоритма Shakersort

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

1. Массив чисел вводится в программу.



1. После сортировки массив чисел с использованием обычной очереди на базе связного списка с помощью алгоритма Shakersort

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной лабораторной работы, я разработал и реализовал алгоритм шейкерной сортировка с использованием обычной очереди на базе связного списка.

Шейкерная сортировка — это алгоритм сортировки, который является вариацией сортировки пузырьком. Алгоритм шейкерной сортировки выполняет проходы по списку в обоих направлениях, сравнивая и переставляя соседние элементы, если это необходимо.

В данной реализации алгоритма шейкерной сортировки используется очередь на базе связного списка для выполнения перестановок элементов. Очередь представляет собой структуру данных, где новые элементы добавляются в конец очереди, а извлекаются из начала. Это позволяет эффективно управлять элементами, которые нужно переставить во время сортировки.

Алгоритм шейкерной сортировки с использованием очереди на базе связного списка выполняет проходы слева направо и справа налево по массиву, сравнивая и перемещая соседние элементы в нужном порядке. После каждого прохода обновляются границы, чтобы исключить уже отсортированные элементы. Процесс повторяется до тех пор, пока не будет выполнена полная сортировка массива.

Таким образом, реализация шейкерной сортировки с использованием обычной очереди на базе связного списка предоставляет эффективный и устойчивый алгоритм сортировки, который может быть использован для упорядочивания элементов любого типа данных. Лабораторная работа успешно реализует шейкерную сортировку с использованием обычной очереди на базе связного списка на языке программирования C++.