

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Дата сдачи на проверку:

«__» _____ 2025 г.

Проверено:

«__» _____ 2025 г.

СОРТИРОВКИ ВСТАВКАМИ И ПИРАМИДАЛЬНАЯ СОРТИРОВКА:
ФУНКЦИИ ПО ССЫЛКЕ И ОБРАБОТКА ФАЙЛОВ

Отчёт по лабораторной работе №5

по дисциплине

«Программирование»

Разработал студент гр. ИВТб-1301-05-00 _____ /Черкасов А. А./

(подпись)

Заместитель кафедры ЭВМ _____ /Долженкова М. Л./

(подпись)

Работа защищена «__» _____ 2025 г.

Киров

2025

Цель

Цель работы: Получить базовые знания о наиболее известных алгоритмах сортировки, изучить принципы работы с текстовыми файлами.

Задания

- Реализовать сортировку данных с помощью вставок.
- Реализовать сортировку данных с помощью пирамидальной сортировки.
- В обоих случаях необходимо предусмотреть возможность изменения компаратора (Реализация компаратора в виде передаваемой в программу функции).
- Считывание и вывод данных необходимо производить из текстового файла.
- Для демонстрации работы программных реализаций самостоятельно подготовить варианты входных данных (при этом объем тестовых файлов должен позволять оценить скорость работы программ).

Решение

Схема алгоритма решения задач представлена на рисунках 1.1 и 1.2. Исходный код представлен в Приложении А1.

Рисунок 1.1 - Схема алгоритма.

Рисунок 1.2 - Схема алгоритма.

Вывод

В ходе выполнения работы изучены основы работы с двусвязным списком, реализованы операции добавления и удаления элементов, а также организован интерактивный пользовательский интерфейс для управления списком.

Приложение А1. Исходный код

```
#include <stdio.h>
#include <stdlib.h>

const char *in = "./data/gen_input.txt";
const char *out = "./data/output.txt";

int cmp_ascending(int *array, int index, int extreme_index, int size) {
    if (index < size && array[index] > array[extreme_index]) {
        return index;
    }
    return extreme_index;
}

int cmp_descending(int *array, int index, int extreme_index, int size) {
    if (index < size && array[index] < array[extreme_index]) {
        return index;
    }
    return extreme_index;
}

void heapify(int *array, int size, int root_index, int (*cmp)(int *, int, int, int)) {
    int largest_index = root_index;
    int left_child_index = 2 * root_index + 1;
    int right_child_index = 2 * root_index + 2;

    largest_index = cmp(array, left_child_index, largest_index, size);
    largest_index = cmp(array, right_child_index, largest_index, size);

    if (largest_index != root_index) {
        int temp = array[root_index];
        array[root_index] = array[largest_index];
        array[largest_index] = temp;
    }
}
```

```

        heapify(array, size, largest_index, cmp);
    }
}

void heap_sort(int *array, int n, int (*cmp)(int *, int, int, int)) {
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(array, n, i, cmp);
    }

    for (int i = n - 1; i > 0; i--) {
        int temp = array[0];
        array[0] = array[i];
        array[i] = temp;

        heapify(array, i, 0, cmp);
    }
}

int main() {
    FILE *file = fopen(in, "r");
    if (file == NULL) {
        return 1;
    }

    int n;
    fscanf(file, "%d", &n);

    int *arr = malloc(sizeof(int) * n);
    if (arr == NULL) {
        fclose(file);
        return 1;
    }
}

```

```

for (int i = 0; i < n; i++) {
    fscanf(file, "%d", &arr[i]);
}

fclose(file);

// heap_sort(arr, n, cmp_ascending);
heap_sort(arr, n, cmp_descending);

FILE *output = fopen(out, "w");
if (output == NULL) {
    free(arr);
    return 1;
}

for (int i = 0; i < n; i++) {
    fprintf(output, "%d ", arr[i]);
}

fclose(output);
free(arr);

return 0;
}

```