МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



**ОТЧЁТ**

**по лабораторной работе № 2-3**

**«Алгоритмы*»***

по дисциплине:

**«Программирование»**

|  |  |
| --- | --- |
| Выполнил(а):  Дашкин С.М. Группа АТ-34 | Проверил:  Кухто А.В. |

Новосибирск

2024

**ЦЕЛЬ РАБОТЫ:**

**Вариант 6.**

1. Написать функцию со следующим прототипом: void GenerateDataset (char \*filename, int num). Функция создает текстовый файл (имя файла передается в переменной filename)из num строк. Каждая строка содержит слово из трех заглавных латинских букв, сгенерированное случайным образом.

2. Написать функцию со следующим прототипом: int SortDataset (char \*filename). Функция читает содержимое файла в массив, сортирует массив по убыванию его элементов, записывает упорядоченный массив в новый файл (один элемент - одна строка). Алгоритм сортировки: пирамидальная сортировка. Тип данных в файле: см. условие задачи 1. Имя нового файла формируется путем добавления к имени предыдущего файла метки .sort (например, filename.txt -> filename.sort.txt).  Функция возвращает целое число, равное общему количеству сравнений элементов массива в процессе сортировки.

3. Написать программу, которая оценивает трудоемкость алгоритма сортировки из задачи 2 для файла данных из задачи 1. Экспериментальная трудоемкость (ТЭ) оценивается как количество сравнений элементов массива в процессе сортировки и далее сравнивается с теоретическими оценками Т1 = num2 и Т2 = num · log2num. Повторите оценки для следующих значений num: 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096. После каждого подобного оценивания записывайте результат в текстовый файл до получения в итоге таблицы вида:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| num | ТЭ | Т1 | Т2 | ТЭ / Т1 | ТЭ / Т2 |
| 8 | … | … | … | … | … |
| 16 | … | … | … | … | … |
| … | … | … | … | … | … |

По результатам экспериментов сделайте вывод о том, к какой модели расчета трудоемкости оказалась ближе рассмотренная задача сортировки.

**ХОД РАБОТЫ:**

Функция GenerateDataset:

Эта функция генерирует случайный файл с заданным количеством строк, каждая из которых содержит три случайные буквы. Она выполняет следующие действия:

1. Инициализирует генератор случайных чисел с использованием текущего времени.

2. Открывает файл для записи.

3. Для указанного количества строк генерирует три случайные буквы и записывает их в файл.

4. Закрывает файл.

Функция swap:

Эта функция обменивает значения двух строк. Она выполняет следующие действия:

1. Сохраняет значение первой строки во временной переменной.

2. Присваивает первой строке значение второй строки.

3. Присваивает второй строке значение временной переменной.

Функция heapify:

Эта функция преобразует поддерево в пирамиду, размещая наибольший элемент в корне. Она выполняет следующие действия:

1. Инициализирует самый большой элемент как корень поддерева.

2. Находит левый и правый дочерние элементы корня.

3. Если левый дочерний элемент больше, чем корень, обновляет самый большой элемент как левый дочерний элемент.

4. Если правый дочерний элемент больше, чем самый большой элемент, обновляет самый большой элемент как правый дочерний элемент.

5. Если самый большой элемент не является корнем, обменивает корень с самым большим элементом и рекурсивно вызывает heapify для поддерева с новым корнем.

Функция heapSort:

Эта функция сортирует массив строк с использованием пирамидальной сортировки. Она выполняет следующие действия:

1. Строит максимальную пирамиду из массива.

2. Повторяет следующие шаги, пока пирамида не станет пустой:

* Обменивает корень пирамиды с последним элементом пирамиды.
* Уменьшает размер пирамиды на 1.
* Вызывает heapify для новой пирамиды, чтобы восстановить ее свойство.

3. Возвращает количество сравнений, выполненных во время сортировки.

Функция SortDataset:

Эта функция считывает строки из файла, сортирует их с использованием пирамидальной сортировки и записывает отсортированные строки в новый файл. Она выполняет следующие действия:

1. Открывает файл для чтения.

2. Считывает строки из файла и сохраняет их в массиве строк.

3. Закрывает файл.

4. Сортирует массив строк с использованием пирамидальной сортировки.

5. Открывает новый файл для записи.

6. Записывает отсортированные строки в новый файл.

7. Закрывает новый файл.

Основная функция main:

Эта функция выполняет следующие действия:

1. Создает массив с различным количеством строк.

2. Открывает файл для записи результатов.

3. Для каждого количества строк в массиве:

* Генерирует файл данных с указанным количеством строк.
* Сортирует файл данных и получает количество сравнений.
* Вычисляет теоретические оценки сложности.
* Записывает результаты в файл.

4. Закрывает файл результатов.

**ЛИСТИНГ:**

#define \_CRT\_SECURE\_NO\_DEPRECATE

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <math.h>

// Генерация случайной строки

void GenerateDataset(char\* filename, int num) {

srand(time(NULL)); // Инициализация генератора случайных чисел

FILE\* file = fopen(filename, "w"); // Открытие файла для записи

if (file == NULL) {

printf("Open error\n");

return;

}

// Генерация и запись случайных строк в файл

for (int i = 0; i < num; ++i) {

fprintf(file, "%c%c%c\n", 'A' + rand() % 26, 'A' + rand() % 26, 'A' + rand() % 26);

}

fclose(file); // Закрытие файла

}

// Функция для обмена значений двух строк

void swap(char\*\* x, char\*\* y) {

char\* temp = \*x;

\*x = \*y;

\*y = temp;

}

// Просеивание элемента вниз пирамиды

void heapify(char\*\* arr, int n, int i) {

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && strcmp(arr[l], arr[largest]) < 0) // Изменение условия для сортировки в порядке убывания

largest = l;

if (r < n && strcmp(arr[r], arr[largest]) < 0) // Изменение условия для сортировки в порядке убывания

largest = r;

if (largest != i) {

swap(&arr[i], &arr[largest]);

heapify(arr, n, largest);

}

}

// Пирамидальная сортировка

int heapSort(char\*\* arr, int n) {

int comparisons = 0;

// Построение max-heap

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

// Извлечение элементов из кучи один за другим

for (int i = n - 1; i > 0; i--) {

swap(&arr[0], &arr[i]);

heapify(arr, i, 0);

comparisons += i; // Количество сравнений

}

return comparisons;

}

// Функция для сортировки и записи в файл

int SortDataset(char\* filename) {

FILE\* file = fopen(filename, "r");

if (file == NULL) {

printf("Open error\n");

return -1;

}

// Считывание строк из файла

char\*\* lines = NULL;

char line[4]; // Так как каждая строка содержит три буквы и символ конца строки '\0'

int numLines = 0;

while (fgets(line, sizeof(line), file) != NULL) {

char\* temp = \_strdup(line); // Копируем строку, чтобы избежать ее перезаписи

lines = (char\*\*)realloc(lines, (numLines + 1) \* sizeof(char\*));

lines[numLines++] = temp;

}

fclose(file);

// Сортировка строк

int comparisons = heapSort(lines, numLines);

// Запись отсортированных строк в новый файл

char\* sortedFilename = (char\*)malloc(strlen(filename) + 6);

strcpy(sortedFilename, filename);

strcat(sortedFilename,".txt");

file = fopen(sortedFilename, "w");

if (file == NULL) {

printf("Create file for record - Error\n");

return -1;

}

for (int i = 0; i < numLines; ++i) {

fprintf(file, "%s", lines[i]);

free(lines[i]); // Освобождение памяти для каждой строки

}

fclose(file);

free(lines);

free(sortedFilename);

return comparisons;

}

// Основная функция

int main() {

int nums[] = { 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 };

int numTests = sizeof(nums) / sizeof(nums[0]);

FILE\* resultsFile = fopen("sorting\_results.txt", "w");

if (resultsFile == NULL) {

printf("Create file for results - Error\n");

return -1;

}

fprintf(resultsFile, "num\tT1\t\tT2\t\tcomparisons\t\tTexp\t\tTexp2\n");

for (int i = 0; i < numTests; ++i) {

int num = nums[i];

char filename[20];

sprintf(filename, "data\_%d.txt", num);

// Генерация файла данных

GenerateDataset(filename, num);

// Сортировка файла и получение количества сравнений

int comparisons = SortDataset(filename);

// Теоретические оценки

double T1 = pow(num, 2);

double T2 = num \* log2(num);

double Texp = comparisons / T1;

double Texp2 = comparisons / T2;

// Запись результатов в файл

fprintf(resultsFile, "%d\t%.2f\t\t%.2f\t\t%d\t\t%.2f\t\t%.2f\n", num, T1, T2, comparisons, Texp, Texp2);

}

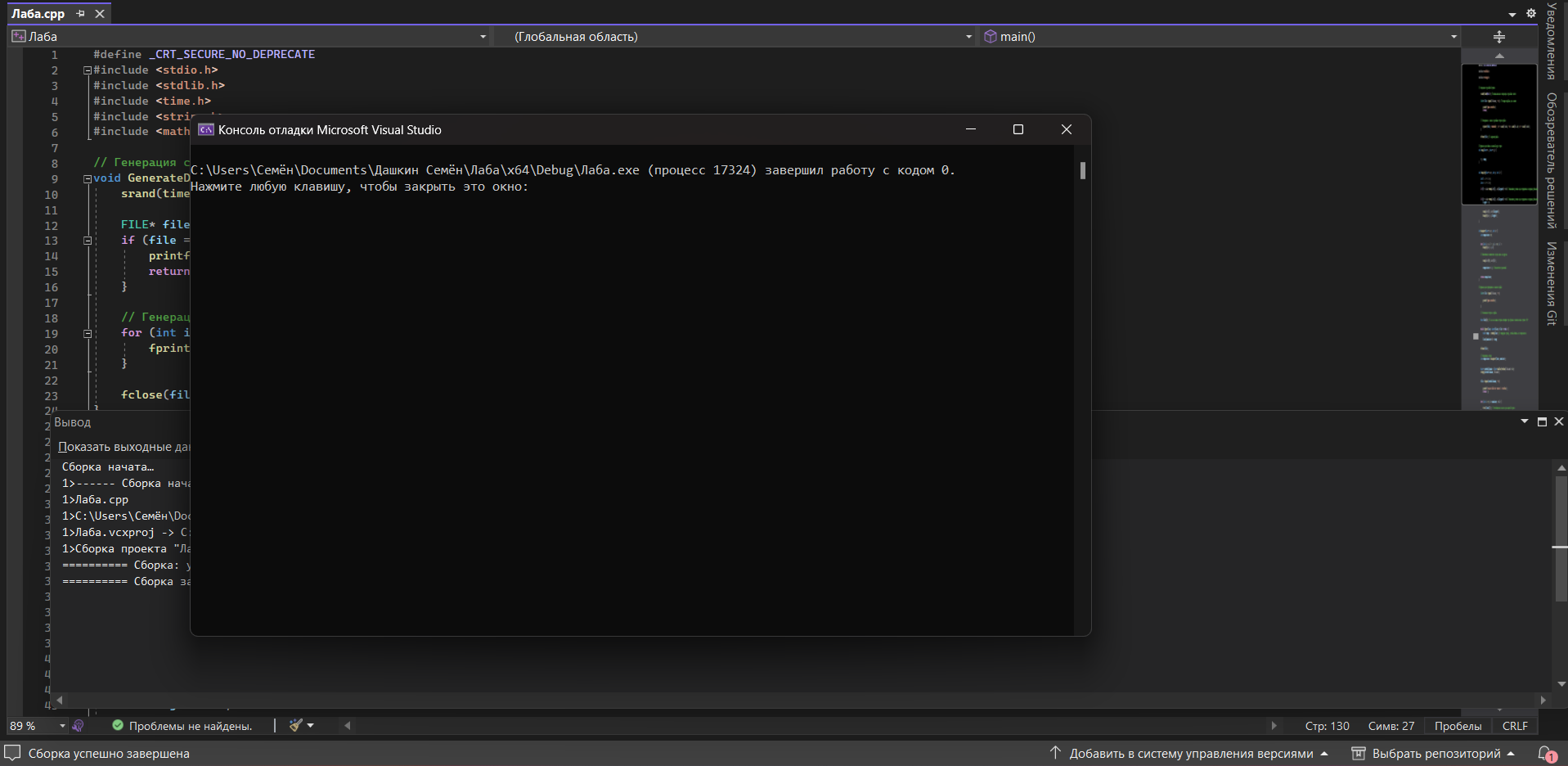
fclose(resultsFile);

return 0;

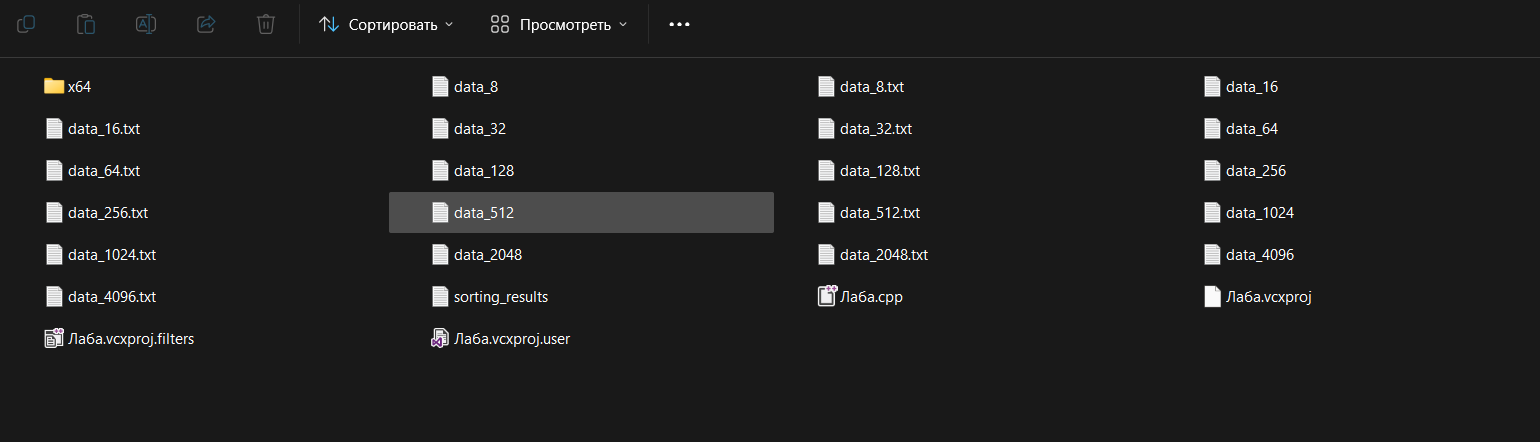
}

**ТЕСТИРОВАНИЕ:**

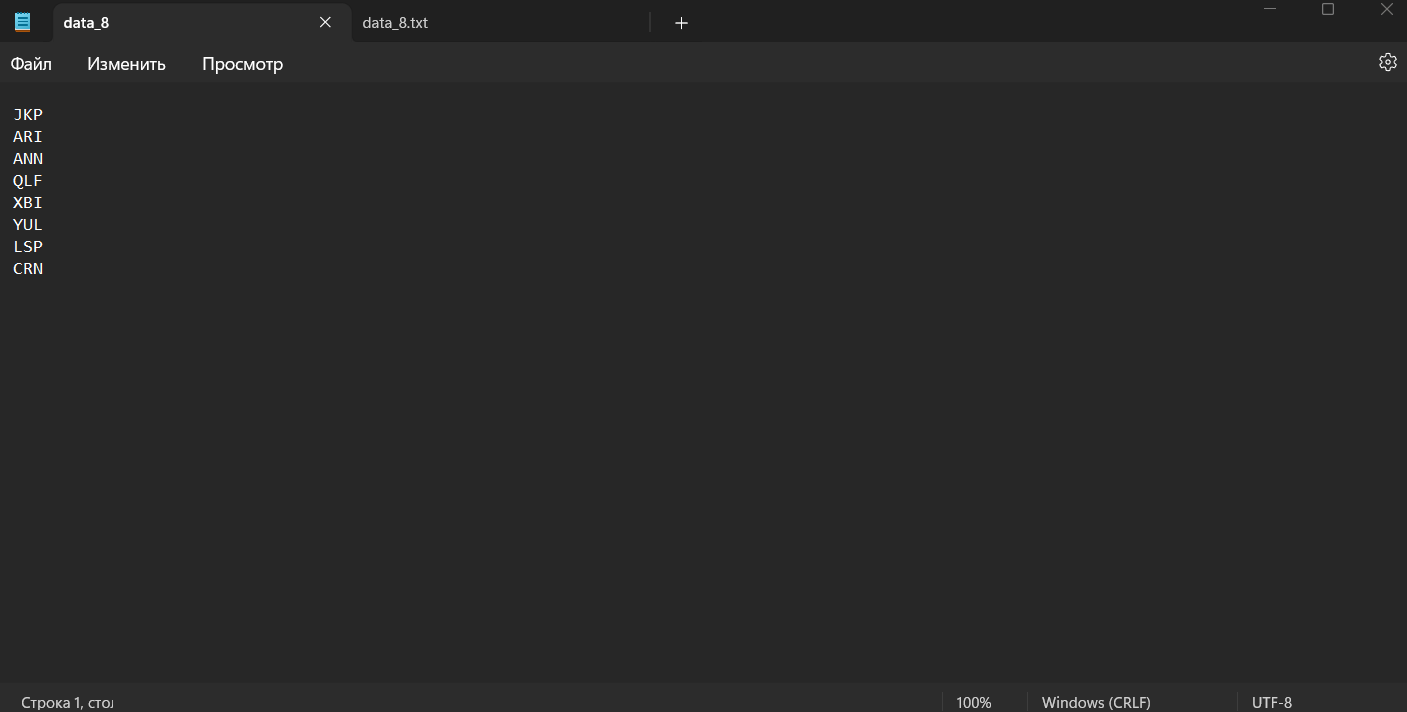
Запуск программы:

****

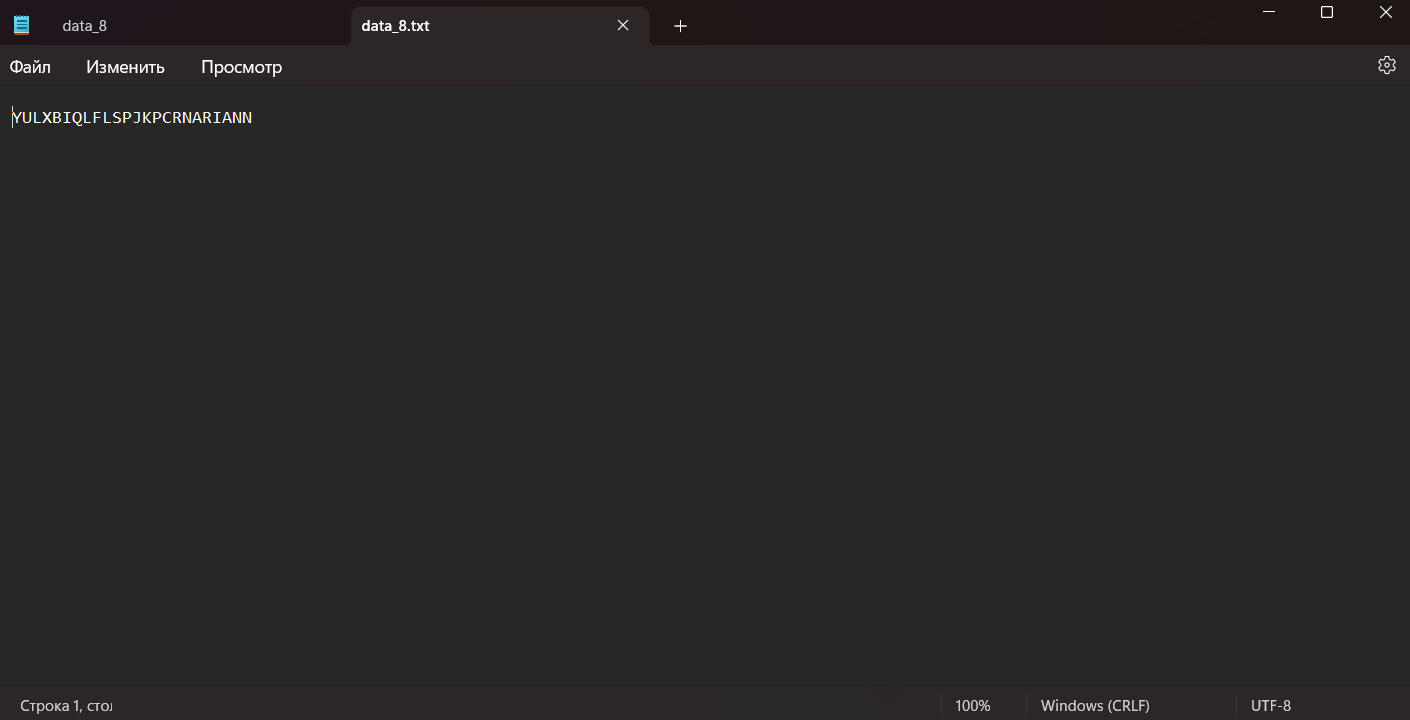
data\_8 – файл без сортировки, data\_8.txt – отсортированный по убыванию, sorting\_results – результаты сортировки для всех массивов(num)



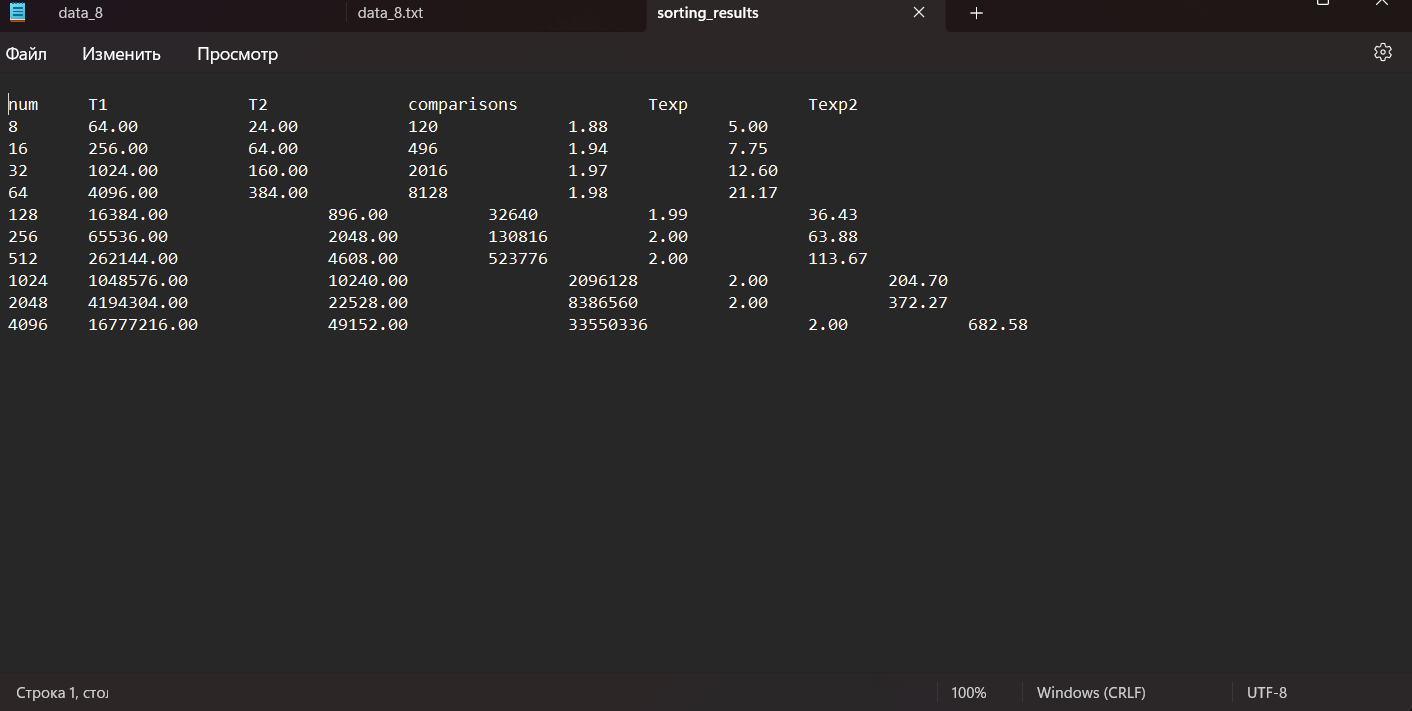
data\_8:



data\_8.txt:

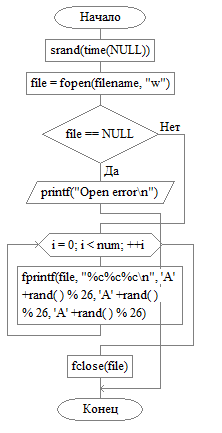


sorting\_results:

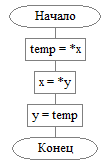


**БЛОК СХЕМЫ:**

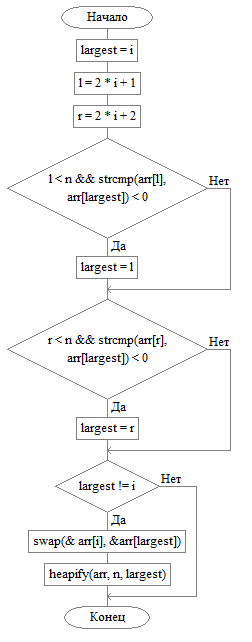
Блок-схема функции generateDataset:



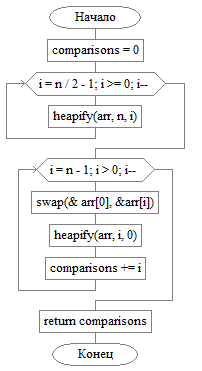
Блок-схема функции swap:



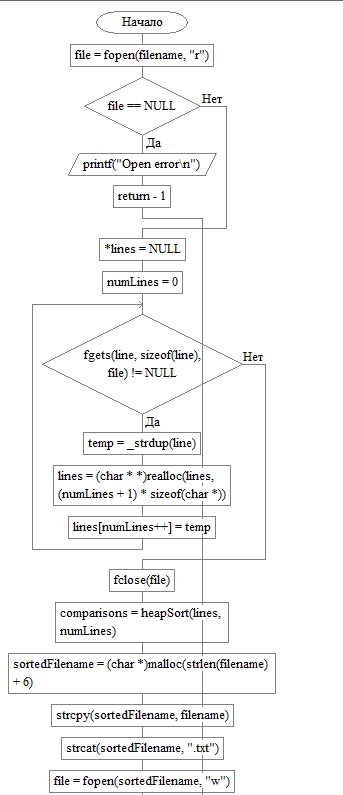
Блок-схема функции heapify:

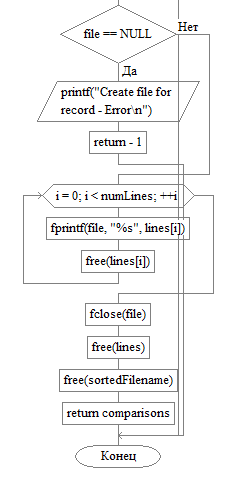


Блок-схема функции heapSort:



Блок-схема функции SortDataset:





Блок-схема функции main:

