Федеральное государственное образовательное учреждение

высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

КАФЕДРА СИСТЕМНОГО АНАЛИЗА И ИНТЕГРАЦИИ IT-РЕШЕНИЙ

**ПРОВЕРИЛ ВЫПОЛНИЛ**

Старший преподаватель кафедры САИТ Студент группы КТбо1-2

Лапшин Вячеслав Сергеев Зворыгин Никита Петрович

«25» февраля 2023 г. «22» февраля 2023 г.

**ЛАБОРАТОРНАЯ РАБОТА №1**

По дисциплине

«Основы алгоритмизации и

программирования»

Вариант 2

На тему

«Алгоритмы сортировки»

**Цель**: Цель данной лабораторной работы - ознакомиться с сортировкой данных и ее видами, а также реализовать некоторые из алгоритмов в соответствии с вариантом лаб.работы .

**Задачи**: Написать программу, которая должна сгенерировать массив из 1000000 случайных целых чисел и записать их в файл input.txt. После этого произвести анализ по времени и выделенной памяти для сортировки сгенерированного массива. Произвести сортировку алгоритмами: сортировка пузырьком, сортировка Шелла, сортировка слиянием.

**Составление математической модели:**

Создаю несколько процедур в программе. Первая процедура отвечает за запись в файл input.txt некоторого кол-ва рандомных значений. Вторая процедура является алгоритмом сортировка пузырьком. Третья сортировка

алгоритмом сортировка Шелла. Четвертая сортировка алгоритмом сортировка слиянием. В главном блоке программы происходит извлечение данных файла в массив. Дальше высчитываем время каждой сортировки. В завершение записываем отсортированный массив в файл out.txt.

Buble Sort: худшем и среднем случае – O(n2), в лучшем случае – O(n).

Shell Sort:  Последовательность Шелла – первый элемент равен длине массива, каждый следующий вдвое меньше предыдущего. Асимптотика в худшем случае – O(n2).

Merge Sort:  Слияние работает за O(n), уровней всего logn, поэтому асимптотика O(nlogn). Эффективно заранее создать временный массив и передать его в качестве аргумента функции. Эта сортировка рекурсивна,

**Ход работы**:

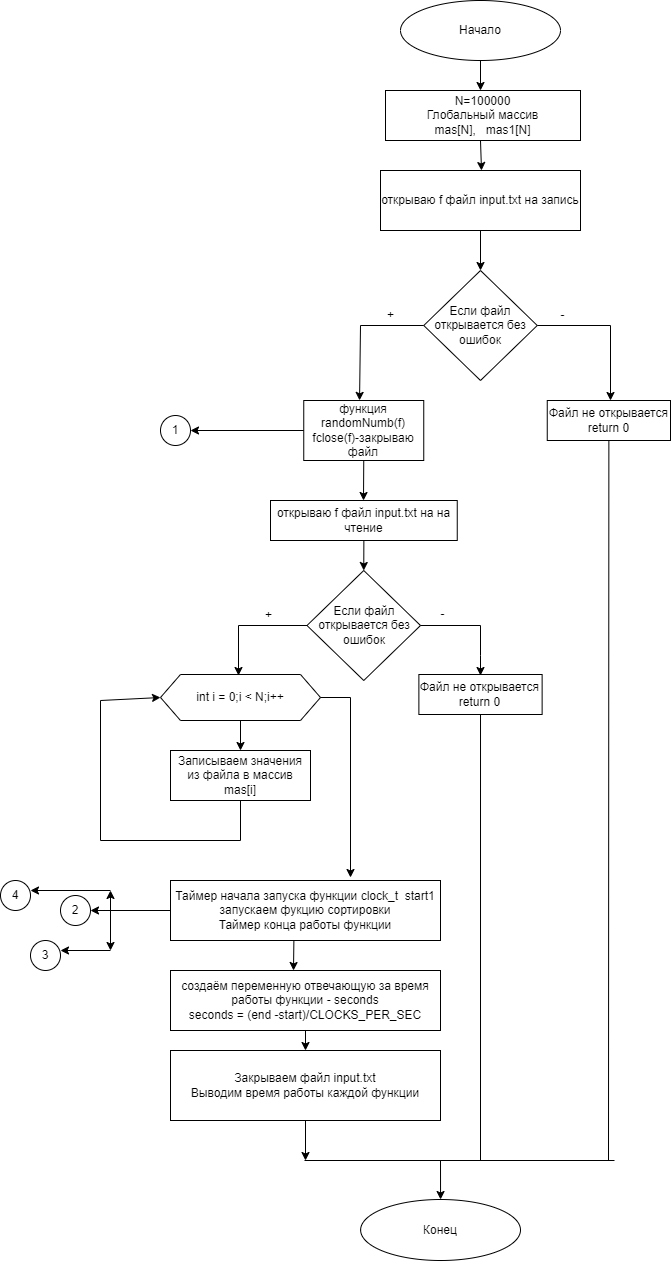


Рисунок 1. Алгоритм работы главной функции

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2. Алгоритм работы процедуры по заполнению массива рандомными элементами

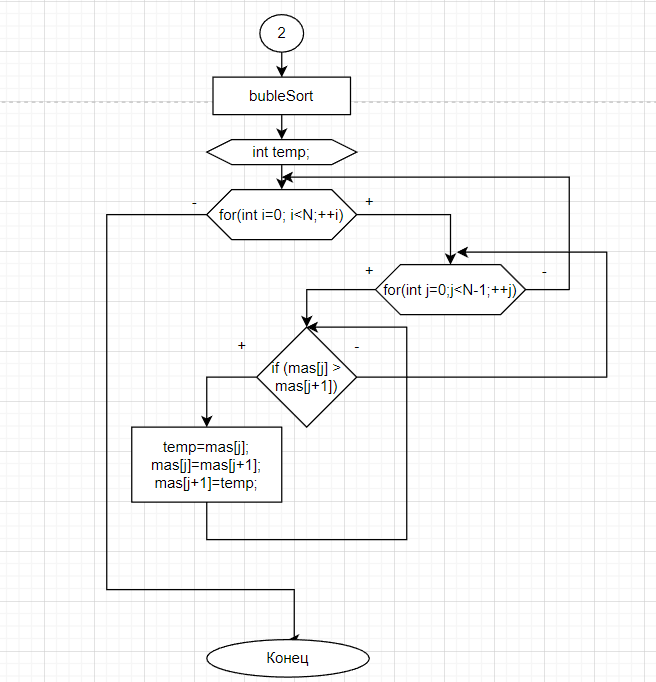


Рисунок 3. Алгоритм работы процедуры алгоритма сортировка пузырьком

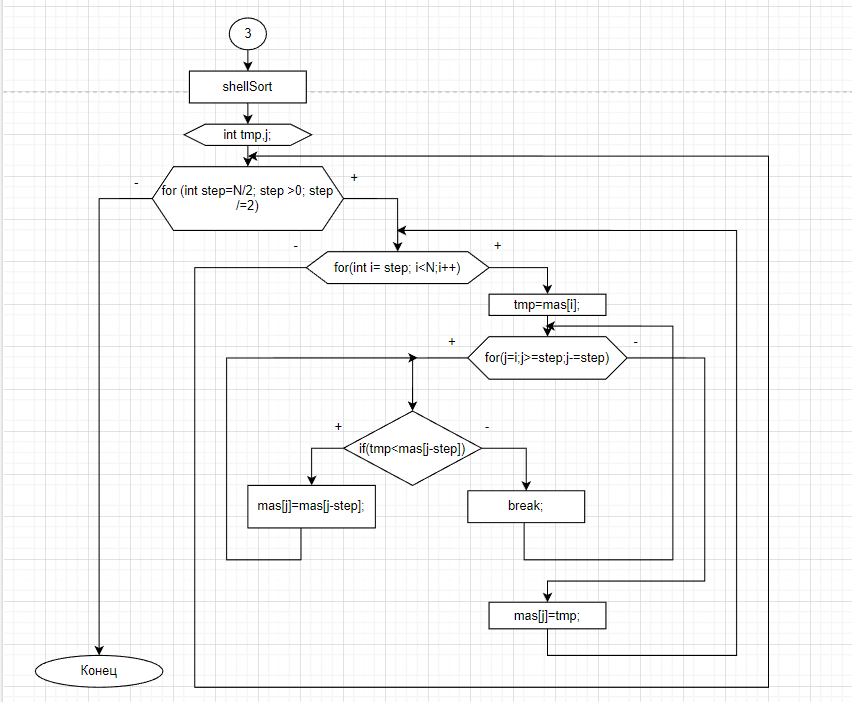
****

Рисунок 4. Алгоритм работы процедуры алгоритма сортировка Шелла

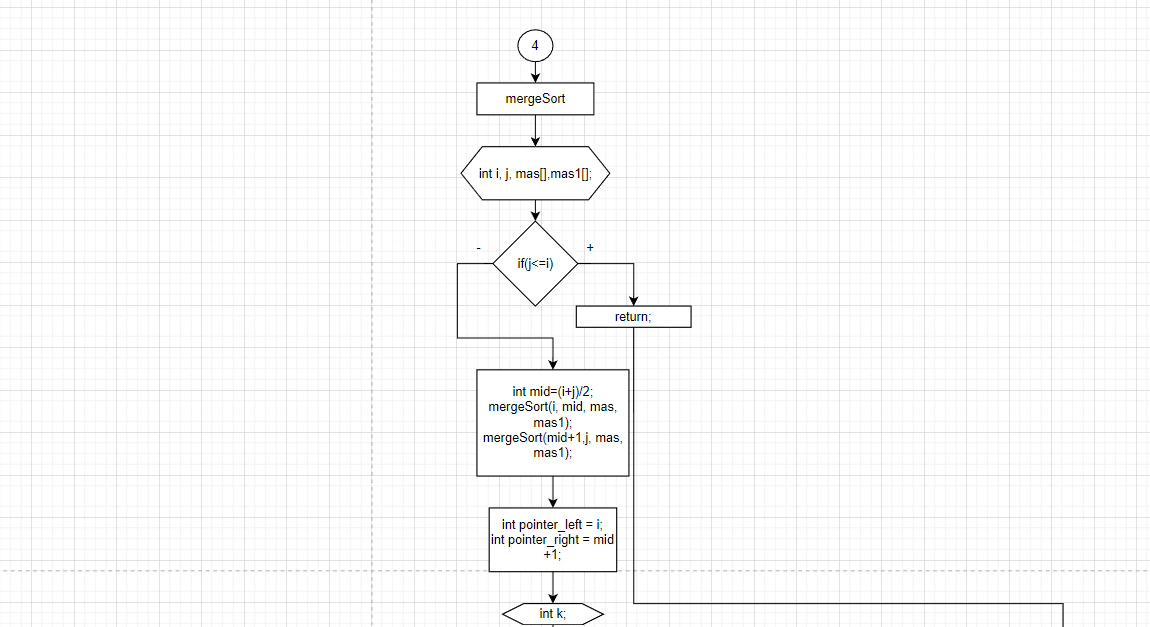
****

Рисунок 5.1 Алгоритм работы процедуры алгоритма сортировка слиянием

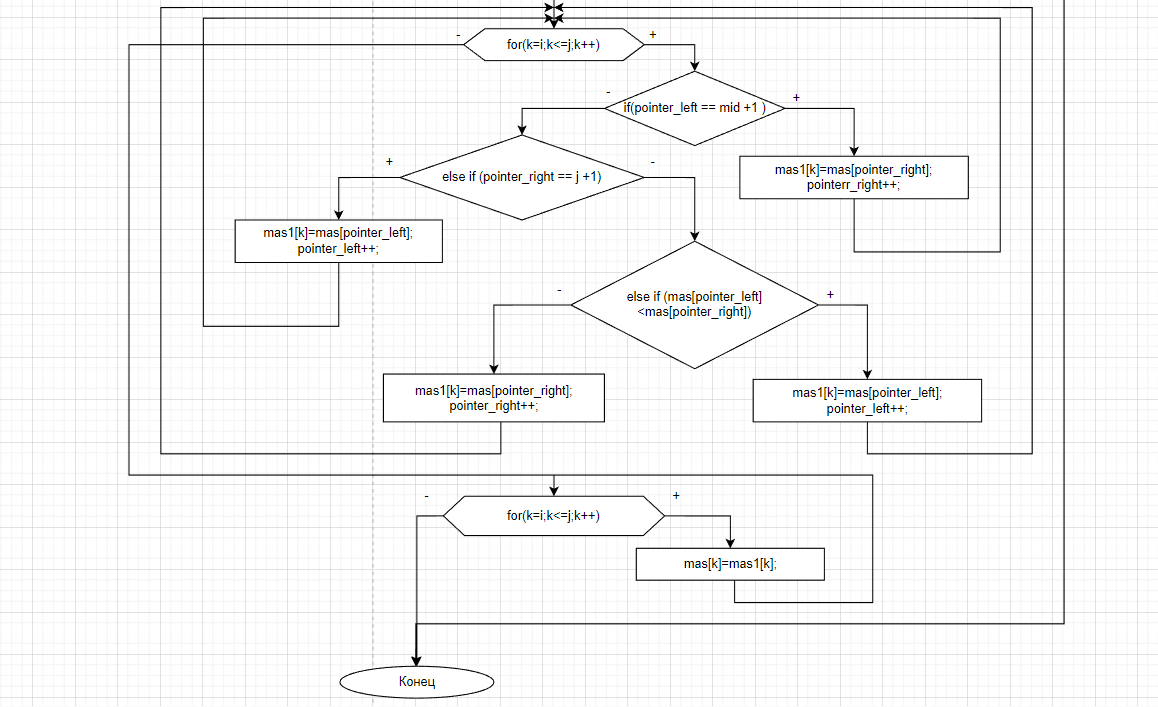
****

Рисунок 5.2 Продолжение процедуры алгоритма сортировка слиянием

**Код программы:**

#include <stdio.h>

#include <time.h>

#include <Windows.h>

const int N = 1000;//значение показывающие кол-во сгенерированных элементов и диапозон чисел

int mas[N];

int mas1[N];

//заполнение массива рандомными значениями

void randomNumb(FILE\* f)

{

for (int i = 0; i < N; ++i)

{

fprintf\_s(f, "%d ", 0 + rand() % N);

}

}

//сортировка пузырьком

void bubleSort()

{

int temp;

for (int i = 0; i < N; ++i)

{

for (int j = 0; j < N - 1; ++j)

{

if (mas[j] > mas[j + 1])

{

temp = mas[j];

mas[j] = mas[j + 1];

mas[j + 1] = temp;

}

}

}

}

//сортировка Шелла

void shellSort()

{

int tmp,j;

for (int step = N / 2; step > 0; step /= 2)

for (int i = step; i < N; i++)

{

tmp = mas[i];

for (j = i; j >= step; j -= step)

{

if (tmp < mas[j - step])

mas[j] = mas[j - step];

else

break;

}

mas[j] = tmp;

}

}

//сортировка слиянием

void mergeSort(int i, int j, int mas[], int mas1[]) {

if (j <= i) {

return;

}

int mid = (i + j) / 2;

mergeSort(i, mid, mas, mas1);

mergeSort(mid + 1, j, mas, mas1);

int pointer\_left = i;

int pointer\_right = mid + 1;

int k;

for (k = i; k <= j; k++) {

if (pointer\_left == mid + 1) {

mas1[k] = mas[pointer\_right];

pointer\_right++;

}

else if (pointer\_right == j + 1) {

mas1[k] = mas[pointer\_left];

pointer\_left++;

}

else if (mas[pointer\_left] < mas[pointer\_right]) {

mas1[k] = mas[pointer\_left];

pointer\_left++;

}

else {

mas1[k] = mas[pointer\_right];

pointer\_right++;

}

}

for (k = i; k <= j; k++) {

mas[k] = mas1[k];

}

}

int main()

{

FILE\* f = fopen("input.txt", "w+");//создаю файл для записи рандомных значений

if (f == NULL)

{

printf\_s("Файл не открылся!");

return 0;

}

randomNumb(f);

fclose(f);

FILE\* f1 = fopen("input.txt", "r");//открываю файл чтобы скопировать значения в массив

if (f1 == NULL)

{

printf\_s("Файл не открылся!");

return 0;

}

for (int i = 0; i < N; i++)

{

fscanf\_s(f1, "%d", &mas[i]);

}

//каждый блок отвечает за вычисления времени каждой сортировки

clock\_t start1 = clock();

bubleSort();

clock\_t end1 = clock();

double seconds1 = (double)(end1 - start1) / CLOCKS\_PER\_SEC;

clock\_t start2 = clock();

shellSort();

clock\_t end2 = clock();

double seconds2 = (double)(end2 - start2) / CLOCKS\_PER\_SEC;

clock\_t start3 = clock();

mergeSort(0,N-1,mas,mas1);

clock\_t end3 = clock();

double seconds3 = (double)(end3 - start3) / CLOCKS\_PER\_SEC;

fclose(f1);

FILE\* f3 = fopen("out.txt", "w");

if (f3 == NULL)

{

printf("Файл не открылся!");

return 0;

}

for (int i = 0; i < N; ++i)

{

fprintf(f3, "%d\n", mas[i]);

}

fclose(f3);

printf("Time sort %d values - %f\n",N, seconds1);

printf("Time sort %d values - %f\n", N, seconds2);

printf("Time sort %d values - %f\n", N, seconds3);

}

**Результат работы программы.**

Таблица 1 — Время на различных сортировках по данным.

|  |  |  |  |
| --- | --- | --- | --- |
| Data/sort | Bubble | Shell | Merge |
| 1000 | 0.003 | 0.0000 | 0.0000 |
| 10000 | 0.449 | 0.001 | 0.003 |
| 100000 | 85.4580 | 0.0100 | 0.0150 |
| 1000000 | - | 0.1280 | 0.1610 |

**Вывод:** понял, как работать с сортировками. Самую эффективную сортировку выявить не удалось, так как иногда выигрывает Шелла, иногда слиянием. Сколько проходит сортировка пузырьком на размере массива 1000000 значений я так и не узнал. Программа работала слишком долго. Сортировка слиянием оказалась самая сложная как в понимании, так и в реализации.