Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Нижегородский Государственный Университет им.

Н.И.Лобачевского» (ННГУ)

Институт информационных технологий математики и механики

Отчёт по лабораторной работе

Сравнение сортировок

Выполнил:

студент группы 3821Б1ФИ3

Исаев Д.А.

Проверил:

заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г.

Нижний Новгород

2021 г.

Содержание

[1. Введение 3](#_Toc90821025)

[2. Постановка задачи 4](#_Toc90821026)

[3. Руководство пользователя 5](#_Toc90821027)

[a. Первая программа 5](#_Toc90821028)

[b. Вторая программа 6](#_Toc90821029)

[4. Руководство программиста 10](#_Toc90821030)

[a. Описание структуры программы 10](#_Toc90821031)

[b. Описание структур данных 10](#_Toc90821032)

[c. Описание алгоритмов 10](#_Toc90821033)

[5. Эксперименты 14](#_Toc90821034)

[6. Заключение 15](#_Toc90821035)

[7. Литература 16](#_Toc90821036)

[8. Приложения 17](#_Toc90821037)

[Приложение A 17](#_Toc90821038)

[Приложение B 18](#_Toc90821039)

# Введение

Программирование — процесс создания [компьютерных программ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). По выражению одного из основателей языков программирования [Никлауса Вирта](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82,_%D0%9D%D0%B8%D0%BA%D0%BB%D0%B0%D1%83%D1%81), «Программы = [алгоритмы](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) + [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)». Программирование основывается на использовании [языков программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), на которых записываются исходные тексты программ.

Массив — упорядоченный набор элементов, каждый из которых хранит одно значение, идентифицируемое с помощью одного или нескольких индексов. Алгоритм сортировки — это алгоритм для упорядочивания элементов в массиве. Так как данные могут хранится в разных структурах, то и алгоритмы для каждой структуры могут отличаться.

В ходе выполнения лабораторный работы на языке программирования «С» будут реализованы сортировки массива данных, такие как сортировка пузырьком (bubble sort), сортировка вставкой (insertion sort) и быстрая сортировка (quick sort). Так же будет проведено сравнение этих сортировок по времени их работы с одним и тем же массивом данных.

# Постановка задачи

Реализовать сортировки массивов данных (тип данных определяется преподавателем (long int)) задаваемых: обязательно случайно, дополнительно с клавиатуры или из файла. Реализовать сортировки: пузырьком, вставкой, быстрая. Сравнить время работы, сделать выводы.

Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n, максимальное и минимальное значение.

Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия.

Дополнительные задания:

* Первая программа может создавать массив не только из случайных чисел, но и получать их с клавиатуры и из файла.
* Добавить вычисление первой номы вектора (массива);
* Добавить вычисление второй нормы вектора (массива);
* Добавить вычисление Гёльдеровой нормы вектора (массива);
* Добавить вычисление бесконечной нормы вектора (массива);
* Добавить нормировку вектора (массива);

# Руководство пользователя

* 1. Первая программа

Запустить программу. После запуска программы нужно выбрать способ задания массива: из случайных чисел (1), ввести с клавиатуры (2), считать из файла (3). Также можно выбрать, завершение работы (4), ведя нужное число, и нажав ENTER (рис. 1).

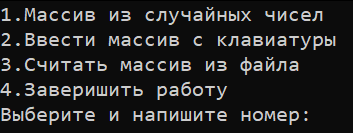


Рисунок 1. Меню выбора

После ввода в консоль числа 1 и нажатия ENTER, нужно ввести минимальное значение, максимальное и размер массива (рис. 2). После чего массив будет сгенерирован и записан в TXT файл, который будет создан или перезаписан (рис. 3).

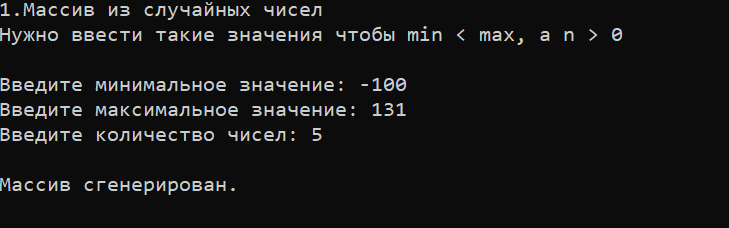


Рисунок 2. Генерация массива из случайных чисел

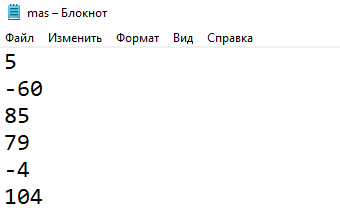


Рисунок 3. TXT файл с массивом

После ввода в консоль числа 2 и нажатия ENTER, нужно ввести размер массива. После чего необходимо вводить числа массива. Когда будет введен последний элемент и будет нажат ENTER, массив будет и записан в TXT файл, который будет создан или перезаписан. Программа выведет, что массив записан (рис. 4).

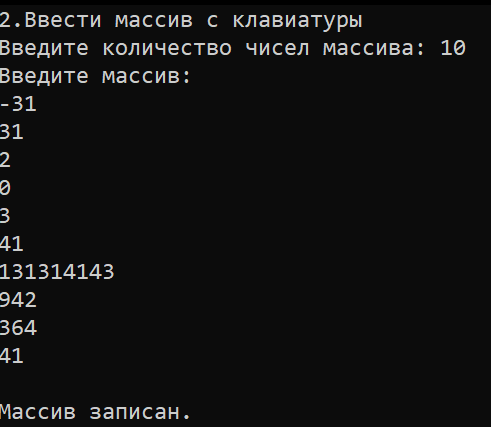


Рисунок 4. Ввод массива с клавиатуры

После ввода в консоль числа 3 и нажатия ENTER, будет выведено сообщение, какие нужны требования. После чего необходимо ввести путь к файлу (например: C:\\Users\\dima4\\source\\example.txt) и нажать ENTER. Массив из этого файла будет скопирован и записан в TXT файл, который будет создан или перезаписан. Программа выведет, что массив считан (рис. 5).

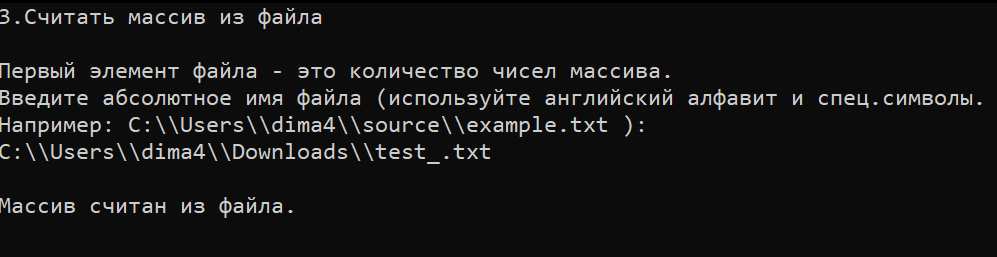


Рисунок 5. Считывание массива из файла

* 1. Вторая программа

После запуска программы будет предоставлен выбор: напечатать массив, сортировать массив, найти норму массива, сбросить, выйти (рис. 6).

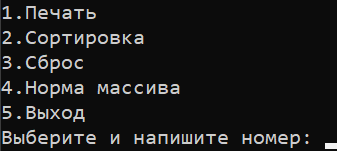


Рисунок 6. Главное меню выбора

После ввода в консоль числа 1, программа напечатает массив (рис. 7) и вернет главное меню выбора.

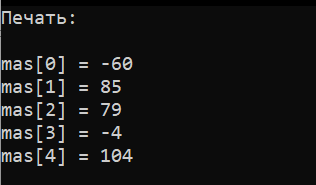


Рисунок 7. Печать массива

После ввода в консоль числа 2, программа предложит сортировки: пузырьком, вставкой, быструю и пункт назад (рис. 8).

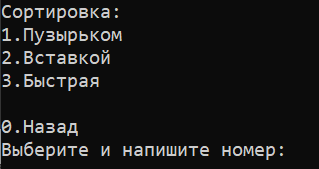


Рисунок 8. Меню выбора сортировки

Например, после ввода в консоль числа 2, программа отсортирует массив вставкой, и выведет время выполнения сортировки, и вернет в главное меню выбора (рис. 9).

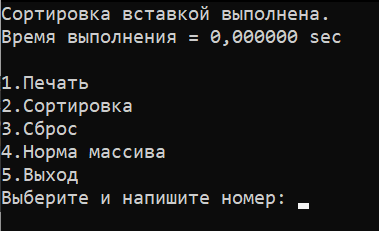


Рисунок 9. Время выполнения и главное меню

При желании можно убедиться, что массив отсортирован, напечатав массив (выбрать число 1)(рис. 10).

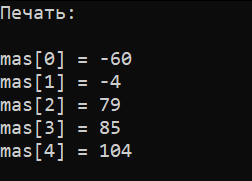


Рисунок 10. Массив после сортировки

При вводе числа 3 происходит сброс массива до изначального (рис. 11).

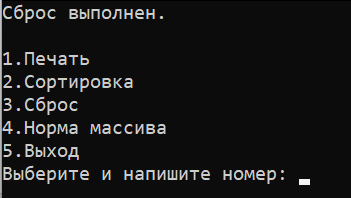


Рисунок 11. Сброс

После ввода в консоль числа 4, программа открывает меню выбора для вычисления норм массива и пункт назад (рис. 12).

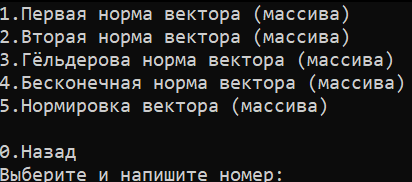


Рисунок 12. Меню норм массива

Например, после ввода в консоль числа 1, программа найдет первую норму массива, и выведет результат, и вернет в меню норм массива (рис. 13).

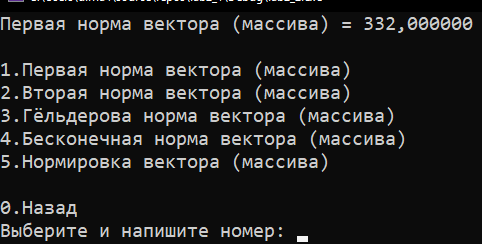


Рисунок 13. Первая норма массива

Для завершения работы программы, в главном меню выбора достаточно ввести число 5 и нажать ENTER.

# Руководство программиста

* 1. Описание структуры программы

Программа состоит из двух модулей: one (см. Приложение A) и two (см. Приложение B). В обоих присутствует главная функция main с возвращаемым типом long int.

Во вторую программу также входят:

Функция void bubbleSort(long \*mas, long n) сортировки пузырьком.

Функция void insertionSort(long\* mas, long n) сортировки вставкой.

Функция void quickSort(long\* mas, long left, long right) быстрой сортировки.

Функция long double norm(long\* mas, long n, long p) нахождения нормы массива.

Функция void writeFile(long\* mas, long n) записи массива в файл.

* 1. Описание структур данных

В программах используются динамические массивы с типом long int. Во второй программе используется два массива. Один - для работы с ним: сортировка, нахождение нормы. Второй – для хранения исходного массива и операции сброса.

* 1. Описание алгоритмов

Сортировка пузырьком. Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется перестановка элементов. Проходы по массиву повторяются n - 1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма)(рис. 14).

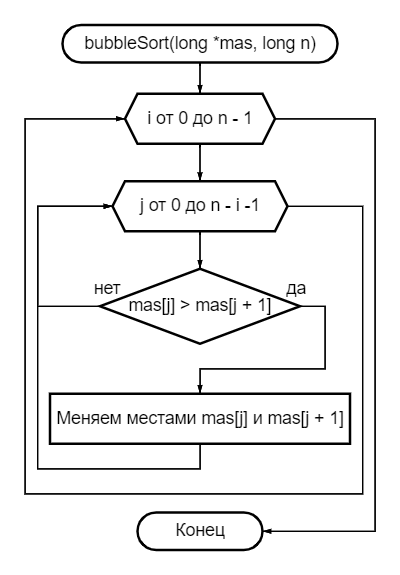


Рисунок 14. Блок-схема сортировки пузырьком

Сортировка вставкой. Есть часть массива, которая уже отсортирована, и требуется вставить остальные элементы массива в отсортированную часть, сохранив при этом упорядоченность. Для этого на каждом шаге алгоритма мы выбираем один из элементов входных данных и вставляем его на нужную позицию в уже отсортированной части массива, до тех пор пока весь набор входных данных не будет отсортирован. Метод выбора очередного элемента из исходного массива произволен, однако обычно (и с целью получения устойчивого алгоритма сортировки), элементы вставляются по порядку их появления во входном массиве (рис. 15).

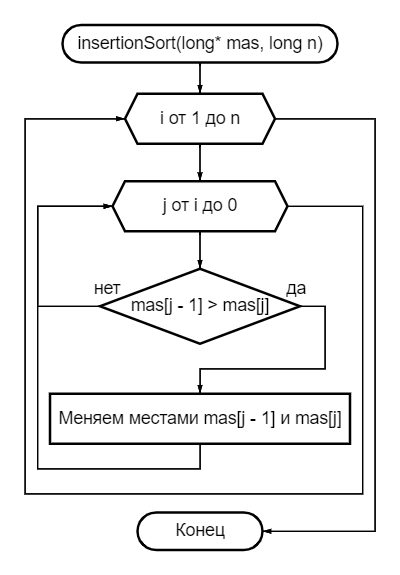


Рисунок 15. Блок-схема сортировки вставкой

Быстрая сортировка (рис.16). Быстрая сортировка относится к алгоритмам «разделяй и властвуй».

Алгоритм состоит из трёх шагов:

1. Выбрать элемент из массива. Назовём его опорным.
2. Разбиение: перераспределение элементов в массиве таким образом, что элементы, меньшие опорного, помещаются перед ним, а большие или равные - после.
3. Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором только один элемент или отсутствуют элементы.

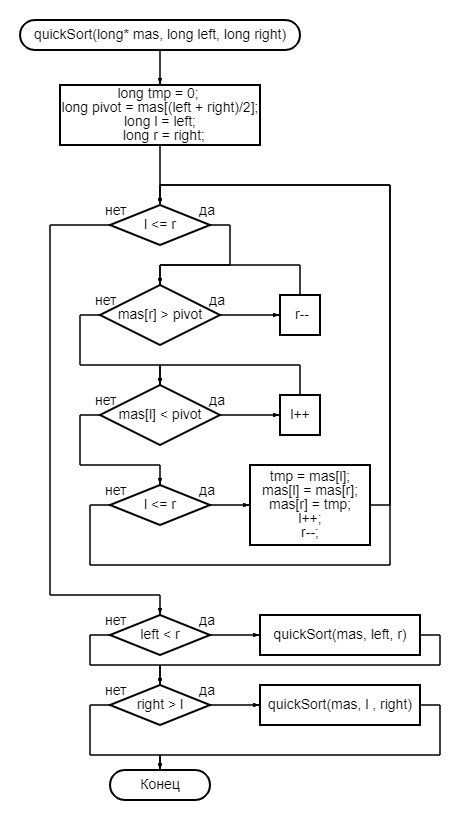


Рисунок 16. Блок-схема быстрой сортировки

# Эксперименты

Проведем несколько тестов с различным количеством элементов для сортировок пузырьком, вставкой, быстрой. Запишем время работы и сравним.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество элементов | 100 000 | 250 000 | 500 000 | 1 000 000 | 2 000 000 |
| Пузырьком, сек | 12,582 | 78,977 | 316,547 | 1271,304 | ~ |
| Вставкой, сек | 3,02 | 18,922 | 75,762 | 304,624 | ~ |
| Быстрая, сек | 0,006 | 0,015 | 0,03 | 0,063 | 0,121 |

Таблица 1. Сравнение времени работы сортировок

Время(пузырьком) > время(вставкой) > время(быстрая).

Как видим время сортировки у быстрой самое малое, следовательно данный алгоритм сортировки более эффективен, чем сортировки вставкой и пузырьком.

# Заключение

В ходе лабораторной работы были написаны две программы на языке Си, в которых удалось реализовать сортировки (пузырьком, вставкой, быстрая) массивов данных (тип данных определялся преподавателем (long int)) задаваемых: обязательно случайно, дополнительно с клавиатуры или из файла. Сравнить время работы и сделать выводы.

Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n, максимальное и минимальное значение.

Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия.

Также были выполнены дополнительные задания:

* Первая программа может создавать массив не только из случайных чисел, но и получать их с клавиатуры и из файла.
* Добавить вычисление первой номы вектора (массива);
* Добавить вычисление второй нормы вектора (массива);
* Добавить вычисление Гёльдеровой нормы вектора (массива);
* Добавить вычисление бесконечной нормы вектора (массива);
* Добавить нормировку вектора (массива);

# Литература

1. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб.: Питер, 2003. —461 с: ил.
2. Программирование. – Режим доступа: <https://ru.wikipedia.org/wiki/Программирование>. – Загл. с экрана.
3. Генератор случайных чисел. – Режим доступа: <https://ru.wikipedia.org/wiki/Генератор_псевдослучайных_чисел>. – Загл. с экрана.
4. Алгоритм сортировки. – Режим доступа: <https://ru.wikipedia.org/wiki/Алгоритм_сортировки>. – Загл. с экрана.
5. Сортировки. – Режим доступа: <https://neerc.ifmo.ru/wiki/index.php?title=Сортировки>. – Загл. с экрана.
6. Быстрая сортировка и с чем её едят. – Режим доступа: <https://habr.com/ru/sandbox/29775/> – Загл. с экрана.
7. Массив. – Режим доступа: <https://ru.wikipedia.org/wiki/Массив_(тип_данных)>. – Загл. с экрана.

# Приложения

## Приложение A

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <locale.h>

#include <time.h>

long main()

{

long max = 0, min = 0, n = 0;

long\* mas;

int a;

FILE\* fo;

FILE\* fi;

setlocale(LC\_ALL, "Rus");

printf("1.Массив из случайных чисел\n2.Ввести массив с клавиатуры\n3.Считать массив из файла\n4.Заверишить работу\nВыберите и напишите номер: ");

scanf("%d", &a);

if (a == 1)

{

while ((min >= max) || (n <= 0))

{

system("cls");

printf("1.Массив из случайных чисел\nНужно ввести такие значения чтобы min < max, a n > 0\n\nВведите минимальное значение: ");

scanf("%ld", &min);

printf("Введите максимальное значение: ");

scanf("%ld", &max);

printf("Введите количество чисел: ");

scanf("%ld", &n);

}

mas = (long\*)malloc(n \* sizeof(long));

srand(time(NULL));

fo = fopen("mas.txt", "w+");

fprintf(fo, "%ld\n", n);

for (long i = 0; i < n; i++)

{

mas[i] = ((double)rand() / RAND\_MAX) \* (max - min) + min;

fprintf(fo, "%ld\n", mas[i]);

}

printf("\nМассив сгенерирован.\n\n");

fclose(fo);

free(mas);

} else if (a == 2)

{

system("cls");

long b = 0;

printf("2.Ввести массив с клавиатуры\nВведите количество чисел массива: ");

scanf("%ld", &n);

fo = fopen("mas.txt", "w");

fprintf(fo, "%d\n", n);

printf("Введите массив:\n");

for (long i = 0; i < n; i++)

{

scanf("%ld", &b);

fprintf(fo, "%ld\n", b);

}

printf("\nМассив записан.\n\n");

fclose(fo);

} else if (a == 3)

{

system("cls");

char\* address;

long c = 0;

printf("3.Считать массив из файла\n\nПервый элемент файла - это количество чисел массива.\nВведите абсолютное имя файла (используйте английский алфавит и спец.символы. Например: C:\\\\Users\\\\dima4\\\\source\\\\example.txt ):\n");

address = (char\*)malloc(RAND\_MAX \* sizeof(char));

scanf("%s", address);

fo = fopen("mas.txt", "w");

fi = fopen(address, "r");

free(address);

fscanf(fi, "%ld\n", &n);

fprintf(fo, "%ld\n", n);

for (long i = 0; i < n; i++)

{

fscanf(fi, "%ld\n", &c);

fprintf(fo, "%ld\n", c);

}

printf("\nМассив считан из файла.\n\n");

fclose(fi);

fclose(fo);

} else return 0;

return 0;

}

## Приложение B

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <locale.h>

#include <time.h>

#include <math.h>

void bubbleSort(long \*mas, long n)

{

long i = 0, j = 0, tmp = 0;

for (i = 0; i < n - 1; i++) {

for (j = 0; j < n - i - 1; j++) {

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

tmp = mas[j];

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

mas[j + 1] = tmp;

}

}

}

}

void insertionSort(long\* mas, long n)

{

long i = 0, j = 0, tmp = 0;

for (int i = 1; i < n; i++) {

for (int j = i; j > 0 && mas[j - 1] > mas[j]; j--) {

tmp = mas[j - 1];

mas[j - 1] = mas[j];

mas[j] = tmp;

}

}

}

void quickSort(long\* mas, long left, long right)

{

long tmp = 0;

long pivot = mas[(left + right)/2];

long l = left;

long r = right;

while (l <= r)

{

while (mas[r] > pivot)

r--;

while (mas[l] < pivot)

l++;

if (l <= r)

{

tmp = mas[l];

mas[l] = mas[r];

mas[r] = tmp;

l++;

r--;

}

}

if (left < r)

quickSort(mas, left, r);

if (right > l)

quickSort(mas, l , right);

}

long double norm(long\* mas, long n, long p)

{

long double result = 0;

for (long i = 0; i < n; i++)

result += pow(abs(mas[i]),p);

return pow(result, 1.0/p);

}

long normInf(long\* mas, long n)

{

long tmp = 0;

for (long i = 0; i < n; i++)

{

if (abs(mas[i]) > tmp)

tmp = abs(mas[i]);

}

return tmp;

}

void writeFile(long\* mas, long n)

{

FILE\* fo;

fo = fopen("../lab2\_1/mas.txt", "w");

fprintf(fo, "%ld\n", n);

for (long i = 0; i < n; i++) fprintf(fo, "%ld\n", mas[i]);

fclose(fo);

}

long main()

{

long n = 0;

int a = 0;

long\* mas;

long\* backup;

FILE\* fi;

setlocale(LC\_ALL, "Rus");

fi = fopen("../lab2\_1/mas.txt", "r");

fscanf(fi, "%ld\n", &n);

mas = (long\*)malloc(n \* sizeof(long));

backup = (long\*)malloc(n \* sizeof(long));

for (long i = 0; i < n; i++)

{

fscanf(fi, "%ld\n", &mas[i]);

backup[i] = mas[i];

}

fclose(fi);

do

{

printf("1.Печать\n2.Сортировка\n3.Сброс\n4.Норма массива\n5.Выход\nВыберите и напишите номер: ");

scanf("%d", &a);

if (a == 1)

{

system("cls");

printf("Печать:\n\n");

for (long i = 0; i < n; i++) printf("mas[%ld] = %ld\n", i, mas[i]);

printf("\n");

}

else if (a == 2)

{

int b = 0;

system("cls");

printf("Сортировка:\n1.Пузырьком\n2.Вставкой\n3.Быстрая\n\n0.Назад\nВыберите и напишите номер: ");

scanf("%d", &b);

if (b == 1)

{

system("cls");

clock\_t t;

t = clock();

bubbleSort(mas, n);

t = clock() - t;

writeFile(mas, n);

printf("Сортировка пузырьком выполнена.\nВремя выполнения = %f sec\n\n", ((double)t) / CLOCKS\_PER\_SEC);

}

else if (b == 2)

{

system("cls");

clock\_t t;

t = clock();

insertionSort(mas, n);

t = clock() - t;

writeFile(mas, n);

printf("Сортировка вставкой выполнена.\nВремя выполнения = %f sec\n\n", ((double)t) / CLOCKS\_PER\_SEC);

}

else if (b == 3)

{

system("cls");

clock\_t t;

t = clock();

quickSort(mas, 0, n - 1);

t = clock() - t;

writeFile(mas, n);

printf("Быстрая сортировка выполнена.\nВремя выполнения = %f sec\n\n", ((double)t) / CLOCKS\_PER\_SEC);

}

else

{

system("cls");

continue;

}

}

else if (a == 3)

{

for (long i = 0; i < n; i++)

mas[i] = backup[i];

system("cls");

writeFile(mas, n);

printf("Сброс выполнен.\n\n");

}

else if (a == 4)

{

system("cls");

int c = 0;

do

{

printf("1.Первая норма вектора (массива)\n2.Вторая норма вектора (массива)\n3.Гёльдерова норма вектора (массива)\n");

printf("4.Бесконечная норма вектора (массива)\n5.Нормировка вектора (массива)\n\n0.Назад\nВыберите и напишите номер: ");

scanf("%d", &c);

if (c == 1)

{

system("cls");

printf("Первая норма вектора (массива) = %Lf\n\n", norm(mas, n, 1));

}

else if (c == 2)

{

system("cls");

printf("Вторая норма вектора (массива) = %Lf\n\n", norm(mas, n, 2));

}

else if (c == 3)

{

system("cls");

long p = 0;

printf("Введите p - норму (p>=1): ");

scanf("%ld", &p);

printf("Гёльдерова норма вектора (массива) = %Lf\n\n", norm(mas, n, p));

}

else if (c == 4)

{

system("cls");

printf("Бесконечная норма вектора (массива) = %ld\n\n", normInf(mas, n));

}

else if (c == 5)

{

system("cls");

for (long i = 0; i < n; i++)

{

printf("unit[%ld] = %Lf\n", i, (double)mas[i] / norm(mas, n, 2));

}

printf("Вектор (массив) нормирован.\n\n");

}

else

{

system("cls");

break;

}

} while (c >= 0);

}

else if (a == 5)

{

free(backup);

free(mas);

return 0;

}

else

{

system("cls");

continue;

}

} while (a >= 0);

free(backup);

free(mas);

return 0;

}