Федеральное государственное образовательное бюджетное учреждение  
высшего профессионального образования  
«Нижегородский Государственный Университет им.  
Н.И.Лобачевского» (ННГУ)

Национальный исследовательский Университет  
Институт Информационных Технологий Математики и Механики

Отчёт по лабораторной работе  
Генерация случайных чисел. Выполнение операций с  
ними.

Выполнил:  
студент группы 3821Б1ФИ3  
Казанцев Е. А.  
Проверил:  
заведующий лабораторией  
суперкомпьютерных технологий и  
высокопроизводительных вычислений  
Лебедев И.Г

Нижний Новгород  
2021 г.

**Содержание.**

[Введение. 3](#_Введение.)

[Постановка задачи. 4](#_Toc86237500)

[Руководство пользователя. 5](#_Toc86237501)

[Руководство программиста. 8](#_Toc86237502)

[Эксперименты. 14](#_Toc86237503)

[Заключение. 19](#_Toc86237504)

[Литература. 20](#_Toc86237505)

[Приложение. 21](#_Toc86237506)

# Введение.

**Программирование** — это процесс превращения алгоритма в нотацию, написанную на языке программирования, которая может быть выполнена компьютером. Одной из ключевых задач компьютера является работа с данными. В данной работе речь пойдет о сортировке данных.

Сортировка является одной из наиболее приятных для умственного анализа категорией алгоритмов, поскольку процесс сортировки очень хорошо определен. [Алгоритмы](https://intellect.icu/metodicheskoe-rukovodstvo-k-kursovoj-rabote-struktury-dannykh-i-algoritmy-4434#term-algoritmy)сортировки были подвергнуты обширному анализу, и способ их работы хорошо понятен. К сожалению, вследствие этой изученности сортировка часто воспринимается как нечто само собой разумеющееся. При необходимости отсортировать данные многие программисты просто вызывают стандартную функцию qsort(), входящую в стандартную библиотеку С. Однако различные подходы к сортировке обладают разными характеристиками. Несмотря на то, что некоторые способы сортировки могут быть в среднем лучше, чем другие, ни один [алгоритм](https://intellect.icu/algoritm-ponyatie-terminologiya-svojstva-zapis-38#term-algoritm)не является идеальным для всех случаев. Поэтому широкий набор алгоритмов сортировки — полезное добавление в инструментарий любого программиста.

**1. Постановка задачи.**

1. Реализовать сортировки массивов данных (тип данных “ long double”) задаваемых: обязательно случайно, дополнительно с клавиатуры или из файла.
2. Реализовать сортировки: пузырьком, вставкой, быстрая.
3. Сравнить время работы, сделать выводы.
4. Должно быть два решения в одном проекте, где 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.

# 2. Руководство пользователя.

Используется две программы. Запустив первую, видим предложение ввести данные, на основе которых будут сгенерированы псевдослучайные числа. Первым предлагается ввести размер массива, затем вводятся числа верхней и нижней границ, в которых будут находиться наши числа. После каждого ввода значений необходимо нажать кнопку «Enter»(см. рис. 1)

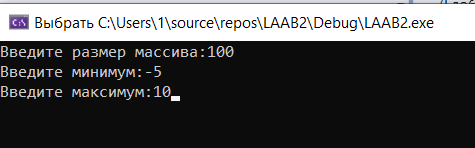


Рисунок 1.Выбор способа ввода.

Если пользователь введет отрицательное значение для кол-ва случайных чисел, то программа запросит ввести все данные снова (см. рис. 2)

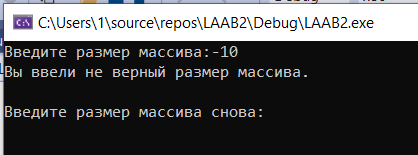


Рисунок 2. Ввод значений.

После ввода всех значений генерируются числа, которые записываются в файл. Затем программа завершает работу. (см. рис. 3)

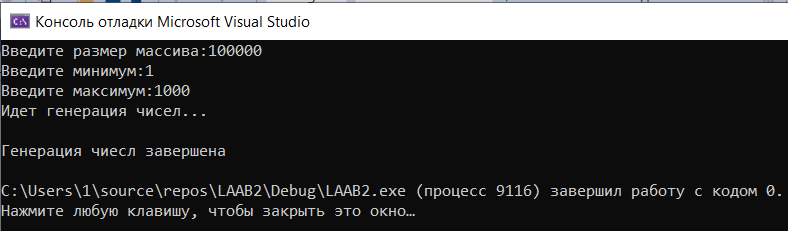


Рисунок 3. Завершение работы первой программы.

После завершения первой программы переходим во вторую. После ее запуска на экран будет выведен консольный интерфейс. (см. рис. 4)

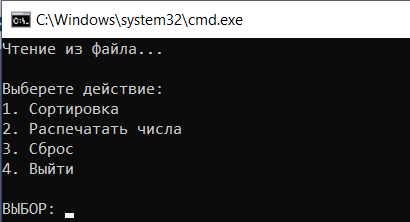


Рисунок 4. Консольный интерфейс.

Консольный интерфейс представляет собой набор из четырех команд:

1. Сортировка

2. Распечатать числа

3. Сброс

4. Выход

* Команда «Сортировка» сортирует числа.
* Команда «Распечатать числа» осуществляет вывод на экран чисел, записанных в файле.
* Команда «Сброс» сбрасывает сортировку.
* Команда «Выход» завершает программу.

Чтобы вывести на экран числа, отсортированные по возрастанию, следует сначала выполнить «Сортировку» и после «Печать».

При выборе команды «Сортировка» пользователю предоставляется выбор типа сортировки. (см. рис. 5)

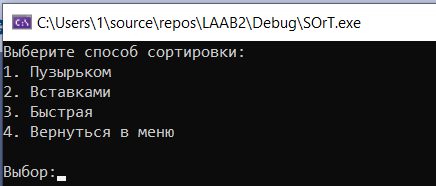


Рисунок 5. Типы сортировки.

После выполнения какой-либо сортировки на экран будет выведено сообщение о времени, занимаемом этим типом сортировки. (см. рис. 6)

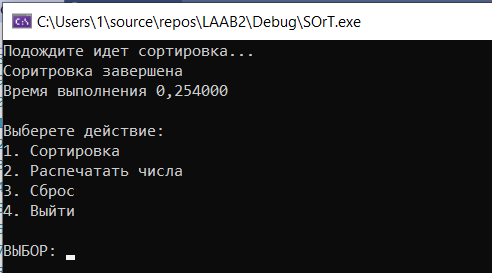


Рисунок 6. Время сортировки.

Чтобы выполнить другую сортировку нужно сбросить предыдущую. Для этого следует воспользоваться командой «Сброс».

Таким образом, мы можем разными способами сортировать числа в файле, сравнивать скорость сортировок и выводить отсортированные по возрастанию числа на экран.

(см. рис. 7)

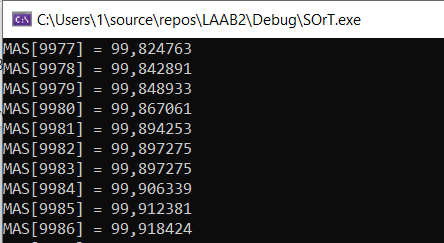


Рисунок 7. Отсортированные числа.

# 3. Руководство программиста.

***Описание структуры программы.***

Первая программа состоит из одного модуля int main() {…}, в котором находится код создания чисел и записи их в файл. Вторая программа также состоит из одного модуля int main() {…}, в котором находится код сортировки чисел.

***Описание алгоритмов.***

**1. Алгоритм записи случайных чисел в файл.**

FILE\* file = fopen("..\\FPrior.txt", "w");

long double\* mas;

long double min, max, tmp = 0.0;

mas = (long double\*)malloc(N \* sizeof(long double));

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

{

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

}

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

{

fprintf(file, "%lf\n", mas[i]);

}

**2. Алгоритм считывания чисел из файла.**

FILE\* file = fopen("..\\FPrior.txt", "r");

double\* MAS, \* MAS1;

MAS = (double\*)malloc(size \* sizeof(double));

MAS1 = (double\*)malloc(size \* sizeof(double));

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

fscanf\_s(file, "%lf", &a[i]);

**3. Алгоритм консольного интерфейса.**

memcpy(MAS1, MAS, sizeof(long double) \* size);

do

{

printf("Выберете действие:\n1. Сортировка\n2. Распечатать числа\n3. Сброс\n4. Выйти\n\nВЫБОР: ");

do

{

scanf\_s("%d", &flag);

if (flag < 1 or flag > 4)

{

printf("Ошибка. Введите значение снова\n\nВЫБОР:");

}

} while (flag < 1 or flag > 4);

if (flag == 1)

{

clrscr();

printf("Выберите способ сортировки:\n1. Пузырьком\n2. Вставками\n3. Быстрая\n4. Вернуться в меню\n\nВыбор:");

do

{

scanf\_s("%d", &flag2);

if (flag2 < 1 or flag2 > 4)

{

printf("Неверный ввод. Введите снова\n Выбор:");

}

} while (flag2 < 1 or flag2 > 4);

if (flag2 == 1)

{

clrscr();

startTime = clock();

bSort(MAS1, size);

endTime = clock();

time\_on\_prog = (double)(endTime - startTime) / 1000;

printf("Соритровка завершена\n");

printf\_s("Время выполнения %f\n\n", time\_on\_prog);

}

else if (flag2 == 2)

{

clrscr();

startTime = clock();

sortInsertion(MAS1, size);

endTime = clock();

time\_on\_prog = (double)(endTime - startTime) / 1000;

printf("Сортировка завершена\n");

printf\_s("Время выполнения %f\n\n", time\_on\_prog);

}

else if (flag2 == 3)

{

clrscr();

startTime = clock();

fastSort(MAS1, 0, size - 1);

endTime = clock();

time\_on\_prog = (double)(endTime - startTime) / 1000;

printf("Сортировка завершена\n");

printf\_s("Время выполнения %f\n\n", time\_on\_prog);

}

else

{

clrscr();

}

}

if (flag == 2)

{

for (i = 0; i < size; i++)

printf\_s("MAS[%d] = %lf\n", i, MAS1[i]);

}

if (flag == 3)

{

clrscr();

memcpy(MAS1, MAS, sizeof(long double) \* size);

printf\_s("Сортировка успешно сброшена.\n\n");

}

if (flag == 4)

{

printf("Программа завершена\n");

return 0;

}

} while (flag == 1 or flag == 2 or flag == 3 or flag == 4);

**4. Сортировки.**

**4.1. Сортировка пузырьком.**

void bSort(long double\* arr1, int size)

{

printf("Подождите идет сортировка...\n");

int i, j;

long double b;

for (i = 0; i < size; i++)

{

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

{

if (arr1[j] > arr1[j + 1])

{

b = arr1[j];

arr1[j] = arr1[j + 1];

arr1[j + 1] = b;

}

}

}

}

* 1. **Сортировка вставкой.**

void sortInsertion(long double\* arr2, int size)

{

printf("Подождите идет сортировка...\n");

long double b;

int i, k;

for (i = 1; i < size; i++)

{

for (k = i; k > 0 and arr2[k - 1] > arr2[k]; k--)

{

b = arr2[k];

arr2[k] = arr2[k - 1];

arr2[k - 1] = b;

}

}

} **4.3. Быстрая сортировка.**

void fastSort(long double\* arr3, int start, int end)

{

printf("Подождите идет сортировка...\n");

int left = start;

int right = end;

long double middle = arr3[(left + right) / 2];

while (left <= right)

{

while (arr3[left] < middle)

{

left++;

}

while (arr3[right] > middle)

{

right--;

}

if (left <= right)

{

double buff = arr3[left];

arr3[left] = arr3[right];

arr3[right] = buff;

left++;

right--;

}

}

if (start < right)

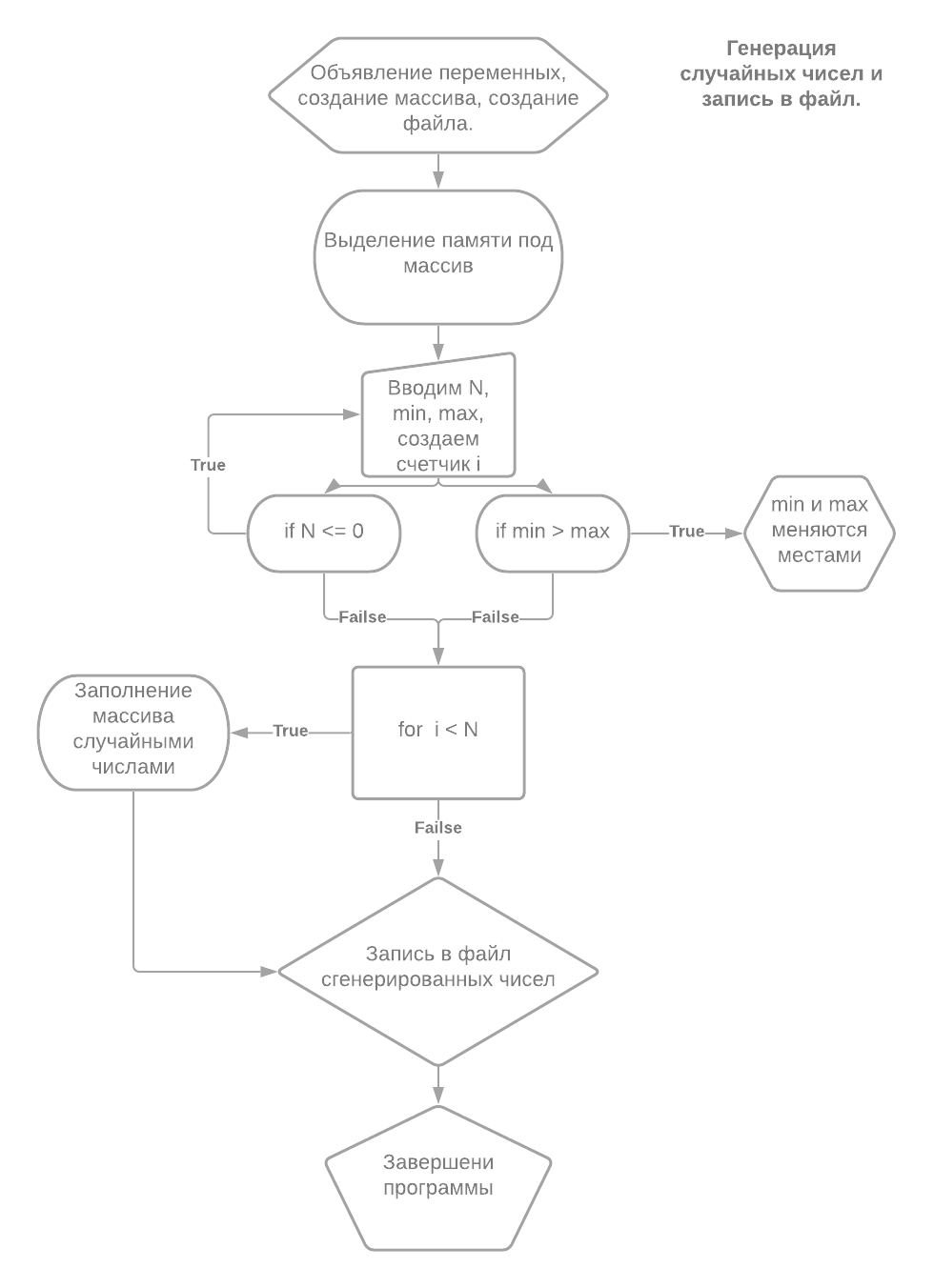
fastSort(arr3, start, right);

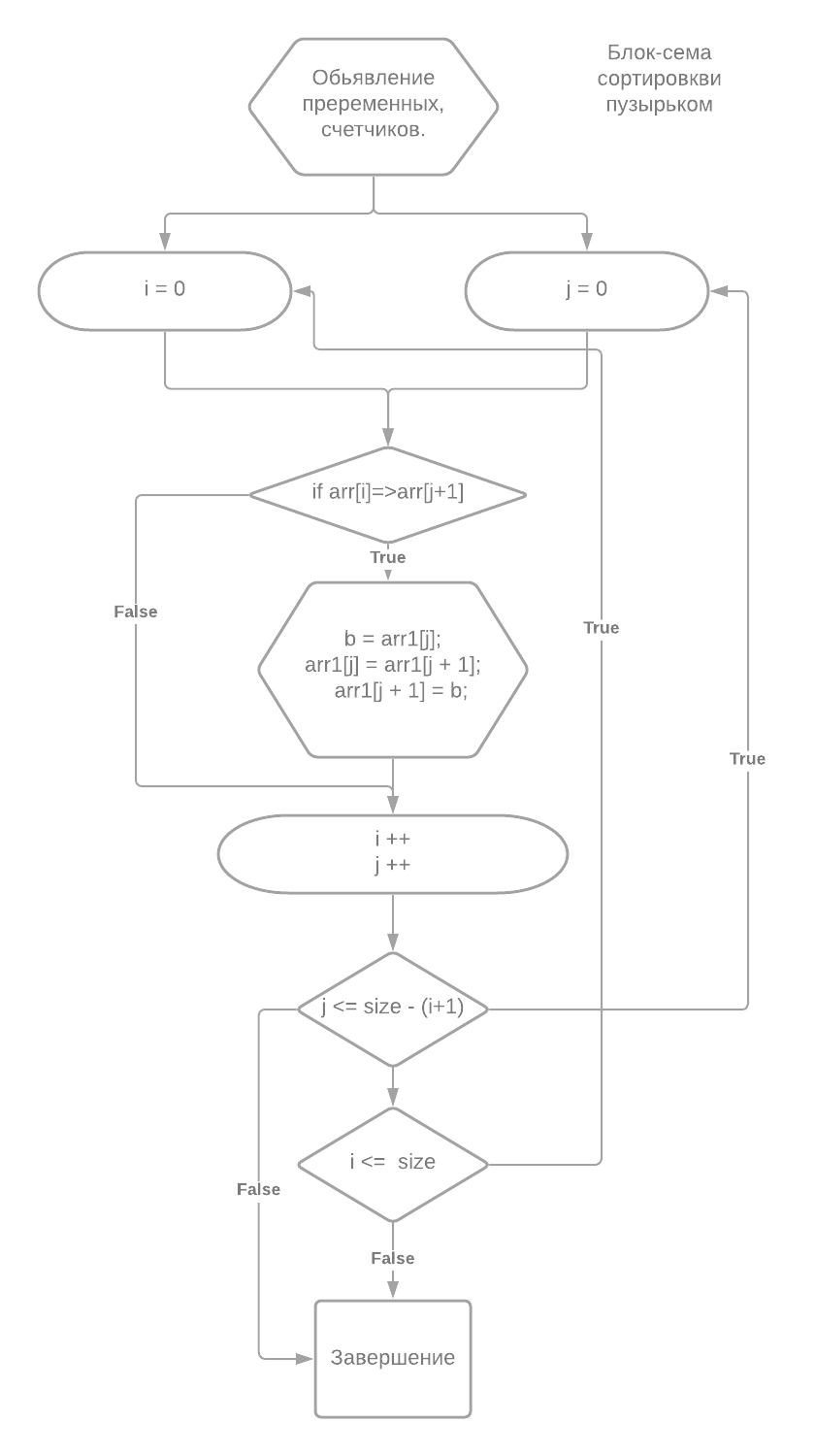
if (end > left)

fastSort(arr3, left, end);

}

# Блок-схемы.

****



# Эксперименты.

Для начала убедимся, что первая программа записывает числа в файл. Изначально «Блокнот» чист:

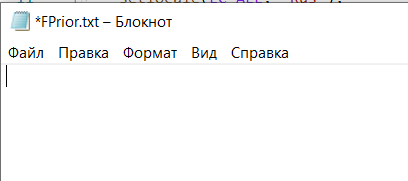


Рисунок 8. Чистый файл.

Далее запустим программу для записи чисел в файл и проверим его на наличие записанных чисел:

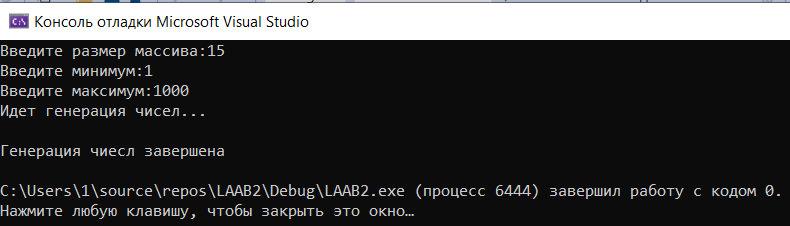


Рисунок 9. Выполнение программы.

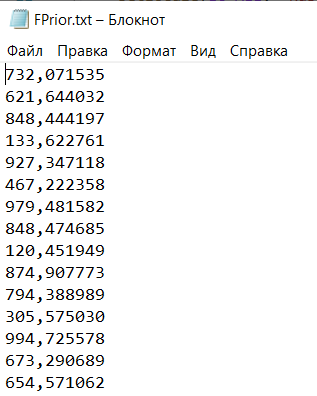


Рисунок 10. Записанные числа в файл.

Действительно, программа записала 15 случайных чисел в правильном диапазоне.

Заметим, если будут введены некорректные данные, то программа выведет на экран сообщение с указанием на ошибку при вводе:

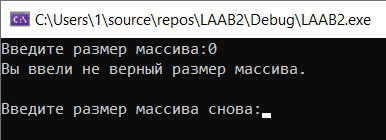


Рисунок 11. Некорректное кол-во чисел.

Далее проверим правильность выполнения второй программы, которая представляет собой консольный интерфейс.

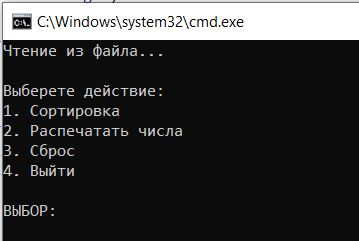


Рисунок 12. Консольный интерфейс.

Выберем команду «Распечатать числа». Для этого введем «2» и нажмем «Enter». На экран будут выведены числа, записанные в файле:

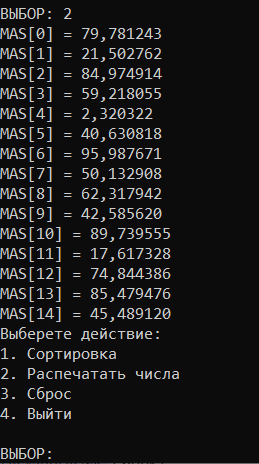
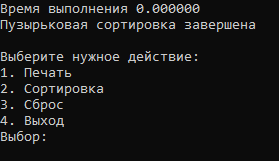


Рисунок 13. Распечатать числа.

Теперь отсортируем их методом пузырька и выведем результат:



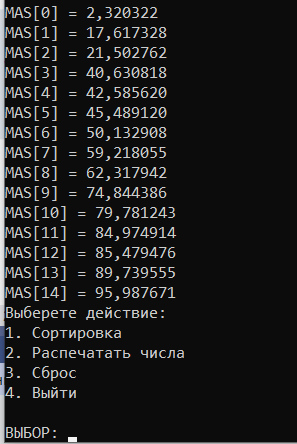


Рисунок 14. Сортировка пузырьком и вывод результата.

Проверим сброс сортировки. Выберем команду «Сброс», после «Распечатать числа»:

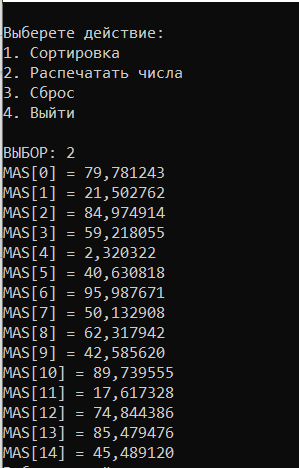


Рисунок 15. Сброс сортировки.

Проверим оставшиеся две сортировки. Для этого после каждой сортировки будем сбрасывать ее:

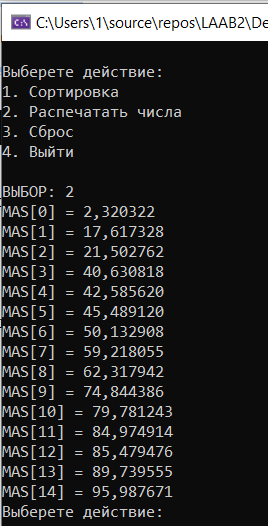


Рисунок 16. Сортировка вставками и вывод результата.

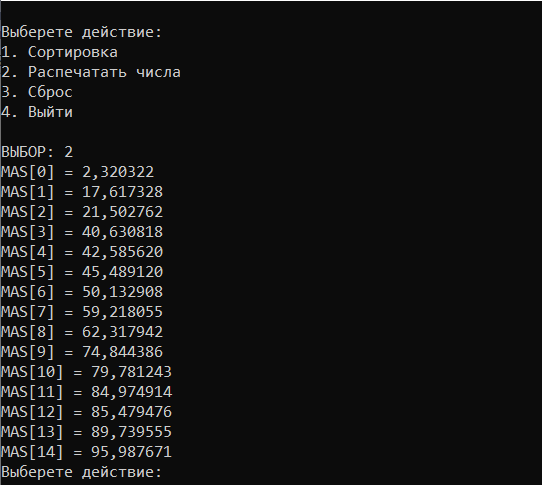


Рисунок 17. Быстрая сортировка и вывод результата.

Наконец, выберем команду «Выход» и завершим программу. На экран выведется сообщение о завершении.

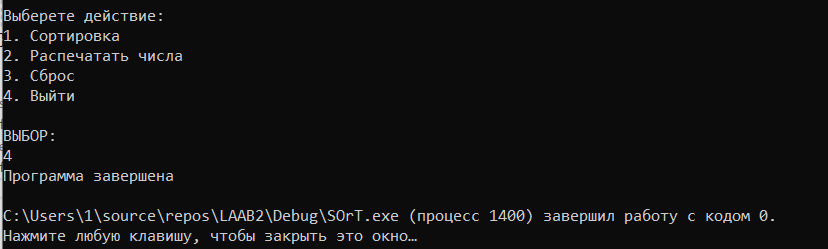


Рисунок 18. Завершение работы программы.

*Таблица скорости работы сортировок*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Вид сортировки** | **1К** | **50К** | **100К** | **500К** | **1М** | **10М** |
| Пузырьком | 0,007с | 7,666с | 31,760с | 993,663с | undefined | undefined |
| Вставками | 0,002с | 3,781с | 15,168с | 439,780с | undefined | undefined |
| Быстрая | 0,000с | 0,018с | 0,039с | 0,138с | 0.216c | 1.658c |

# Заключение.

В ходе лабораторной работы была написана программа на языке программирования «С», которая полностью выполняет поставленную задачу, а именно:

«Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных “long double”) 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.

В ходе сравнения сортировок можно сделать вывод: «быстрая сортировка (fastSort)» выполняет сортировку чисел намного быстрее остальных типов сортировки и очень быстро работает с любыми типами данных; сортировка «пузырьком» занимает наибольшее время; «сортировка вставками» работает быстрее чем «пузырьком».

8.Литература.  
1. Т.А. Павловская Учебник по программированию на языках высокого  
уровня(С/С++) – Режим доступа: http://cph.phys.spbu.ru/documents/First/books/7.pdf  
2. Бьерн Страуструп. Язык программирования С++ - Режим доступа:  
<http://8361.ru/6sem/books/Straustrup-Yazyk_programmirovaniya_c.pdf>

[Википедия язык СИ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2_%D0%A1%D0%B8)

[http://codeforces.com/](http://codeforces.com/gym/101502/problem/E)

# Приложение.

Приложение 1.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include<stdlib.h>

#include <locale.h>

#include<time.h>

#include<iso646.h>

int main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

int i = 0;

int N = 0;

FILE\* file = fopen("..\\FPrior.txt", "w");

long double\* mas;

long double min, max, tmp = 0.0;

printf("Введите размер массива:");

scanf("%d", &N);

if (N <= 0)

{

printf("Вы ввели не верный размер массива.\n\nВведите размер массива снова:");

scanf("%d", &N);

}

printf("Введите минимум:");

scanf("%lf", &min);

printf("Введите максимум:");

scanf("%lf", &max);

if (min > max)

{

min = tmp;

tmp = max;

max = min;

}

printf("Идет генерация чисел...\n\n");

mas = (long double\*)malloc(N \* sizeof(long double));

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

{

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

}

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

{

fprintf(file, "%lf\n", mas[i]);

}

printf("Генерация чиесл завершена\n");

fclose(file);

free(mas);

return 0;

}

Приложение 2.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <conio.h>

#include<locale.h>

#include<iso646.h>

clrscr()

{

system("@cls||clear");

}

void bSort(long double\* arr1, int size)

{

printf("Подождите идет сортировка...\n");

int i, j;

long double b;

for (i = 0; i < size; i++)

{

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

{

if (arr1[j] > arr1[j + 1])

{

b = arr1[j];

arr1[j] = arr1[j + 1];

arr1[j + 1] = b;

}

}

}

}

void sortInsertion(long double\* arr2, int size)

{

printf("Подождите идет сортировка...\n");

long double b;

int i, k;

for (i = 1; i < size; i++)

{

for (k = i; k > 0 and arr2[k - 1] > arr2[k]; k--)

{

b = arr2[k];

arr2[k] = arr2[k - 1];

arr2[k - 1] = b;

}

}

}

void fastSort(long double\* arr3, int start, int end)

{

int left = start;

int right = end;

long double middle = arr3[(left + right) / 2];

while (left <= right)

{

while (arr3[left] < middle)

{

left++;

}

while (arr3[right] > middle)

{

right--;

}

if (left <= right)

{

double buff = arr3[left];

arr3[left] = arr3[right];

arr3[right] = buff;

left++;

right--;

}

}

if (start < right)

fastSort(arr3, start, right);

if (end > left)

fastSort(arr3, left, end);

}

const long double RAND\_MAX\_F = RAND\_MAX;

int stringCount(FILE\* strC) {

int result = 0;

while (!ferror(strC) and !feof(strC)) {

if (fgetc(strC) == '\n')

result++;

}

rewind(strC);

return result;

}

int main()

{

int size, flag = 0, flag2 = 0, i;

double startTime, endTime, time\_on\_prog;

long double\* MAS, \* MAS1;

FILE\* file = fopen("..\\FPrior.txt", "r");

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

printf\_s("Чтение из файла...\n\n");

size = stringCount(file);

MAS = (long double\*)malloc(size \* sizeof(long double));

MAS1 = (long double\*)malloc(size \* sizeof(long double));

if (MAS == 0 or MAS1 == 0)

return 1;

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

fscanf\_s(file, "%lf", &MAS[i]);

memcpy(MAS1, MAS, sizeof(long double) \* size);

do

{

printf("Выберете действие:\n1. Сортировка\n2. Распечатать числа\n3. Сброс\n4. Выйти\n\nВЫБОР: ");

do

{

scanf\_s("%d", &flag);

if (flag < 1 or flag > 4)

{

printf("Ошибка. Введите значение снова\n\nВЫБОР:");

}

} while (flag < 1 or flag > 4);

if (flag == 1)

{

clrscr();

printf("Выберите способ сортировки:\n1. Пузырьком\n2. Вставками\n3. Быстрая\n4. Вернуться в меню\n\nВыбор:");

do

{

scanf\_s("%d", &flag2);

if (flag2 < 1 or flag2 > 4)

{

printf("Неверный ввод. Введите снова\n Выбор:");

}

} while (flag2 < 1 or flag2 > 4);

if (flag2 == 1)

{

clrscr();

startTime = clock();

bSort(MAS1, size);

endTime = clock();

time\_on\_prog = (double)(endTime - startTime) / 1000;

printf("Соритровка завершена\n");

printf\_s("Время выполнения %f\n\n", time\_on\_prog);

}

else if (flag2 == 2)

{

clrscr();

startTime = clock();

sortInsertion(MAS1, size);

endTime = clock();

time\_on\_prog = (double)(endTime - startTime) / 1000;

printf("Сортировка завершена\n");

printf\_s("Время выполнения %f\n\n", time\_on\_prog);

}

else if (flag2 == 3)

{

clrscr();

startTime = clock();

fastSort(MAS1, 0, size - 1);

endTime = clock();

time\_on\_prog = (double)(endTime - startTime) / 1000;

printf("Сортировка завершена\n");

printf\_s("Время выполнения %f\n\n", time\_on\_prog);

}

else

{

clrscr();

}

}

if (flag == 2)

{

for (i = 0; i < size; i++)

printf\_s("MAS[%d] = %lf\n", i, MAS1[i]);

}

if (flag == 3)

{

clrscr();

memcpy(MAS1, MAS, sizeof(long double) \* size);

printf\_s("Сортировка успешно сброшена.\n\n");

}

if (flag == 4)

{

printf("Программа завершена\n");

return 0;

}

} while (flag == 1 or flag == 2 or flag == 3 or flag == 4);

fclose(file);

free(MAS);

free(MAS1);

}