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  
Загорулько А.А.

Проверил:  
заведующий лабораторией  
суперкомпьютерных технологий и  
высокопроизводительных вычислений  
Лебедев И.Г

Нижний Новгород  
2021 г.

Содержание

[Введение 3](#_Toc90733486)

[1. Постановка задачи 3](#_Toc90733487)

[2. Руководство пользователя 4](#_Toc90733488)

[2.1 Программа 1 5](#_Toc90733489)

[2.1.1 «Генерация чисел» 5](#_Toc90733490)

[2.1.2 «Ввести с клавиатуры» 6](#_Toc90733491)

[2.1.3 «Считать данные с другого файла» 7](#_Toc90733492)

[2.2 Программа 2 7](#_Toc90733493)

[2.2.1 Команда «Печать» 7](#_Toc90733494)

[2.2.2 Команда «Сортировка» 8](#_Toc90733495)

[2.2.3 Команда «Сброс» 8](#_Toc90733496)

[2.2.4 Команда «Работа с массивом» 9](#_Toc90733497)

[3. Руководство программиста. 9](#_Toc90733498)

[3.1 Описание структуры программы. 10](#_Toc90733499)

[3.2 Описание алгоритмов. 10](#_Toc90733500)

[3.2.1. Алгоритм записи случайных чисел в файл. 10](#_Toc90733501)

[3.2.2. Алгоритм записи вводимых чисел в файл. 10](#_Toc90733502)

[3.2.3. Сортировка пузырьком. 11](#_Toc90733503)

[3.2.4. Сортировка вставкой. 11](#_Toc90733504)

[3.2.5. Быстрая сортировка. 12](#_Toc90733505)

[4. Эксперименты 13](#_Toc90733506)

[5. Заключение. 17](#_Toc90733507)

[6. Литература. 18](#_Toc90733508)

[7. Приложение 19](#_Toc90733509)

[7.1 Приложение 1 19](#_Toc90733510)

[7.2 Приложение 2 22](#_Toc90733511)

# Введение

**Программирование** — это процесс создания программ (программного обеспечения). Для этого программисты пишут исходный код на одном из языков программирования.

Сортировка является одной из наиболее приятных для умственного анализа категорией алгоритмов, поскольку процесс сортировки очень хорошо определен. Алгоритмы 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 не является идеальным для всех случаев. Поэтому широкий набор алгоритмов сортировки — полезное добавление в инструментарий любого программиста.

# 

# 1. Постановка задачи

Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных “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

# 2. Руководство пользователя

## 2.1 Программа 1

Используется две программы. При запуске первой программы на экран выводится консольный интерфейс, который позволяет выбрать: «Ввести вручную», «Сгенерировать числа» или «Копирование из файла» (см. рис. 1)

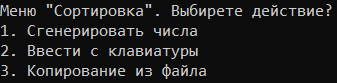


Рис. 1. Консольный интерфейс.

### 2.1.1 «Генерация чисел»

При выборе «Сгенерировать числа», Вам нужно сначала ввести кол-во чисел, затем нажать «Enter». Далее, ввести минимальный элемент диапазона случайных чисел, затем нажать «Enter». Далее, ввести максимальный элемент диапазона случайных чисел, затем нажать «Enter». Затем, будет произведена проверка на корректность данных. (см. рис. 2)

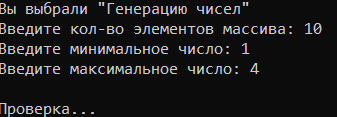


Рис. 2. Ввод значений и проверка на корректность.

Если пользователь введет отрицательное значение «кол-во элементов массива», программа выдаст ошибку и попросит пользователя ввести данные заново. В случае, если «минимальный элемент» будет больше, чем «максимальный элемент», программа также попросит пользователя заново ввести данные. (см. рис. 3, 4)

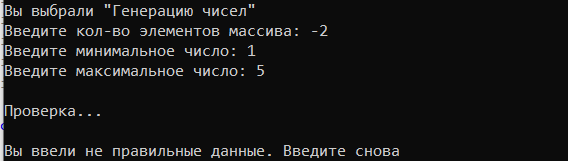


рис 3. Проверка на корректность «элементов массива»

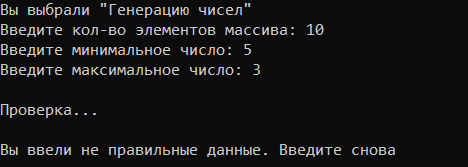


рис 4. Проверка на корректность «минимальный элемент» и «максимальный элемент».

В случае, если данные корректны, произойдет подтверждение и запись сгенерированных чисел в текстовой файл. (см. рис. 5, 6)

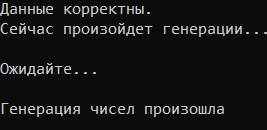


Рис. 5. Завершение генерации и запись в файл.

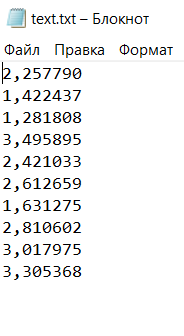


Рис. 6. Записанные числа в файле.

### 2.1.2 «Ввести с клавиатуры»

При выборе «Ввести с клавиатуры», Вам нужно сначала ввести кол-во чисел, затем нажать «Enter». Далее, Вам нужно вводить свои числа через «запятую». Каждое новое число записывать через «Enter». После чего будет произведена запись в текстовой файл. (см. рис 7)

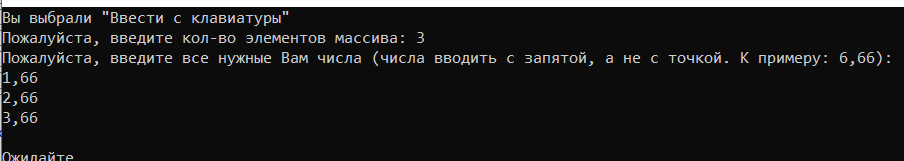


Рис. 7. Ввод и запись.

### 2.1.3 «Считать данные с другого файла»

При выборе «Считать данные с другого файла», Вам нужно ввести кол-во элементов в другом файле. (cм. Рис 8)

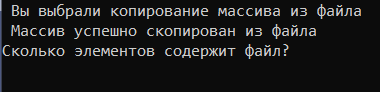


Рис. 8. Считывание данных с другого файла.

## 2.2 Программа 2

После завершения первой программы следует запустить вторую. После ее запуска на экран будет выведен консольный интерфейс. (см. рис. 9)

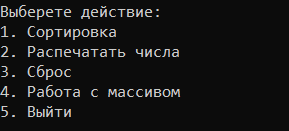


Рис. 9. Консольный интерфейс.

Консольный интерфейс представляет собой набор из пяти команд:

1. Сортировка

2. Печать

3. Сброс

4. Работа с массивом

5. Выход

* Команда «Печать» осуществляет вывод на экран чисел, записанных в файле.
* Команда «Сортировка» сортирует числа.
* Команда «Сброс» сбрасывает сортировку.
* Команда «Работа с массивом» предоставляет несколько норм вычисления массива.
* Команда «Выход» завершает программу.

### 2.2.1 Команда «Печать»

При выборе «Печать» Вы сможете вывести в консоль все числа, которые записаны в файле. После вывода на экран, Вы можете выбрать другое действие из интерфейса (см. рис 10)

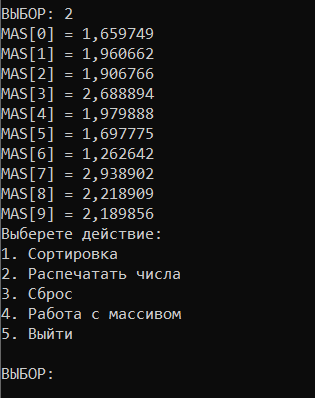


Рис. 10. Вывод чисел на экран.

### 2.2.2 Команда «Сортировка»

При выборе «Сортировка» откроется дополнительный консольный интерфейс с выбором варианта сортировки или возвращение в меню. (см. рис. 11)

1. Пузырьком

2. Вставками

3. Быстрая

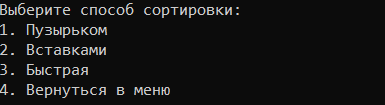


Рис. 11. Варианты сортировки.

После выполнения какой-либо сортировки на экран будет выведено сообщение о времени, занимаемом этим типом сортировки. (см. рис. 12)

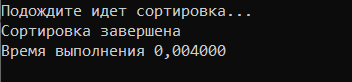


Рис. 12. Время сортировки.

### 2.2.3 Команда «Сброс»

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. 13)

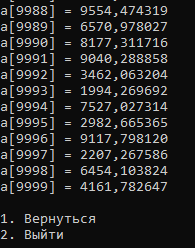


Рис. 13. Сброс сортировки

### 2.2.4 Команда «Работа с массивом»

Чтобы вычислить норму вектора (массива), можно воспользоваться данной вкладкой в меню. У вас будет доступно 5 возможных вычислений норм (см. рис 14) :

1. Вычисление первой номы вектора (массива);

2. Вычисление второй нормы вектора (массива);

3. Вычисление Гельдеровой нормы вектора (массива);

4. Вычисление бесконечной нормы вектора (массива);

5. Нормировку вектора (массива).

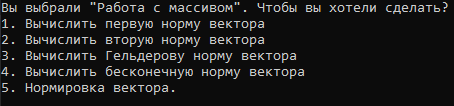


Рис. 14. Работа с массивом.

# 

# 3. Руководство программиста.

## 3.1 Описание структуры программы.

Первая программа состоит из одной функции **int main() {…},** в котором находится код создания или ввод чисел и записи их в файл.

Вторая программа состоит из функции **int main() {…},** в котором находится код консольного интерфейса, из трех функций сортировки и из функций счёта векторов.

## 3.2 Описание алгоритмов.

### 3.2.1. Алгоритм записи случайных чисел в файл.

|  |
| --- |
| for (j = 0; j < size\_gener; j++)  {  fprintf(file, "%lf\n", (rand() / RAND\_MAX\_F) \* ((double)max\_gener - (double)min\_gener) + min\_gener);  } |

### 3.2.2. Алгоритм записи вводимых чисел в файл.

|  |
| --- |
| for (j = 0; j < len\_inter\_keyboard; j++)  {  scanf\_s("%lf", &inter\_keyboard);  fprintf(file, "%f\n", inter\_keyboard);  } |

### 3.2.3. 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;  }  }  }  } |

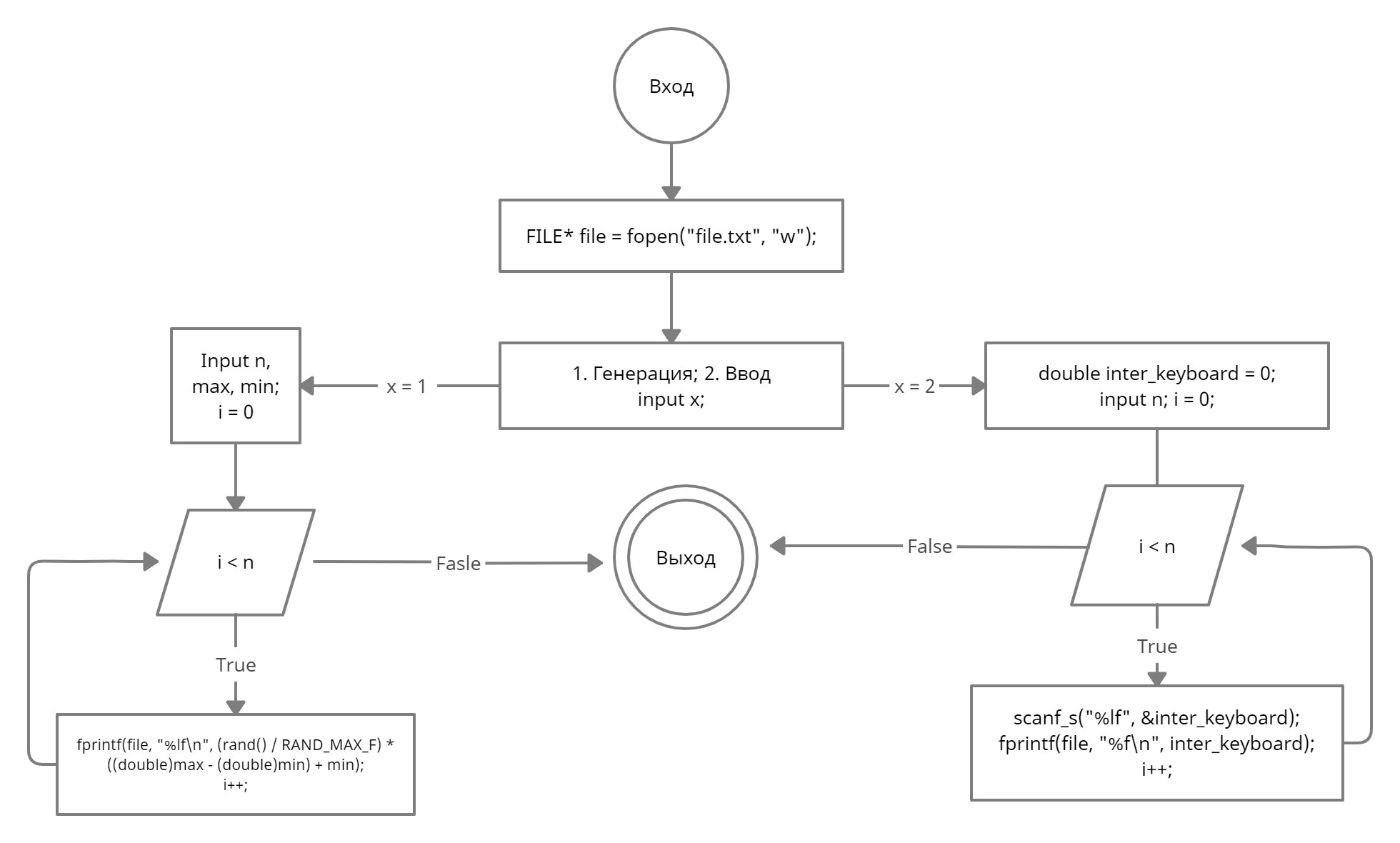
### 3.2.4. Сортировка вставкой.

|  |
| --- |
| 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;  }  } |

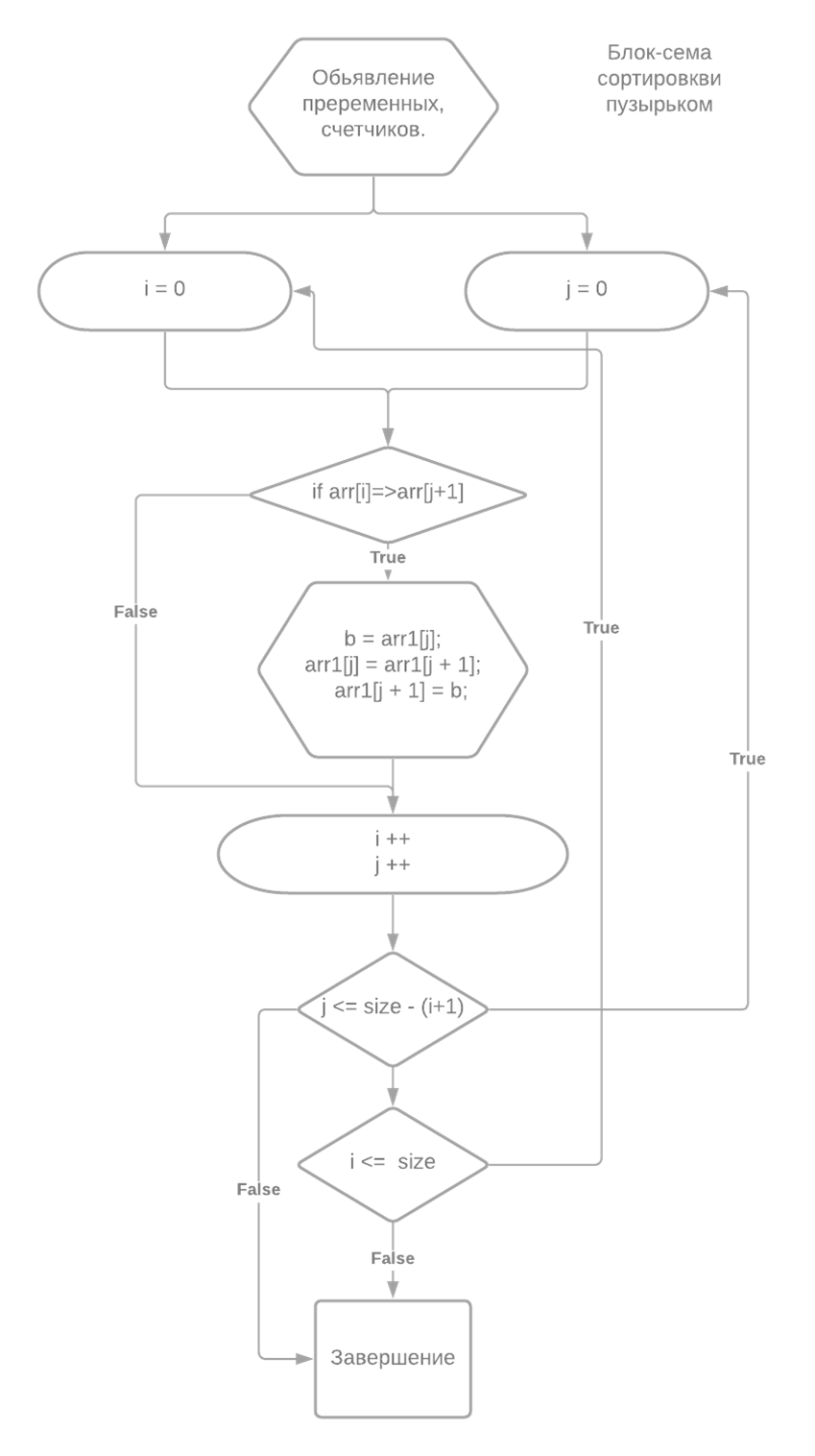
### 3.2.5. Быстрая сортировка.

|  |
| --- |
| 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-схема программы 1.**



**Блок-схема программы 2.**



# 4. Эксперименты

Для начала убедимся, что первая программа записывает числа в файл. Изначально «Блокнот» чист (см. рис 16):

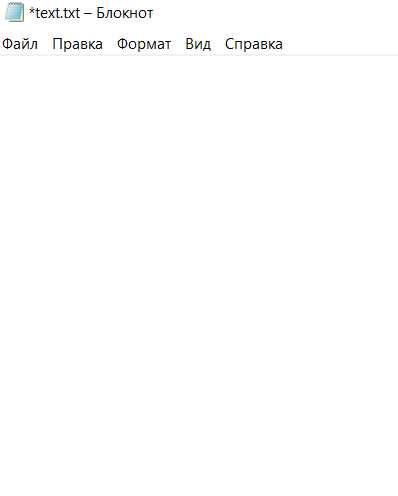


Рис. 16. Чистый файл.

Далее запустим программу для записи чисел в файл и проверим его на наличие записанных чисел (см. рис 17):

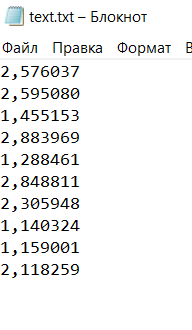


Рис. 17. Одна часть записанных чисел.

Действительно, программа записала 10 случайных чисел в правильном диапазоне.

Заметим, если будут введены некорректные данные, то программа выведет на экран сообщение с указанием на ошибку при вводе (см. рис 18):

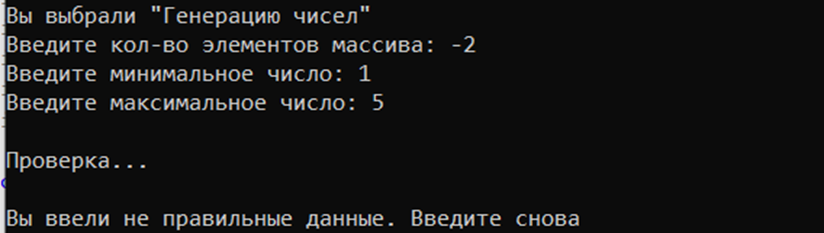


Рис. 18. Некорректный диапазон / некорректное кол-во чисел.

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

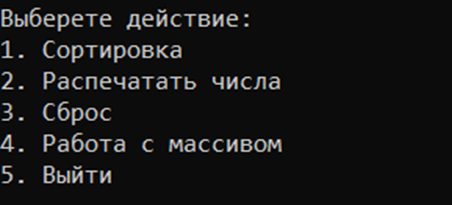


Рис. 19. Консольный интерфейс.

Выберем команду «Печать». Для этого введем «2» и нажмем «Enter». На экран будут выведены числа, записанные в файле, а также возвращение в меню:

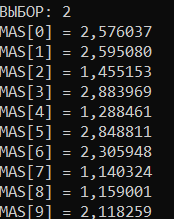


Рис. 20. Печать и возврат в меню.

Теперь вернемся в меню и отсортируем их методом пузырька и выведем результат:

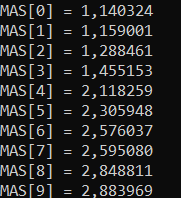


Рис. 21. Вывод отсортированного списка.

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

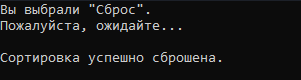


Рис. 22. Сброс сортировки.

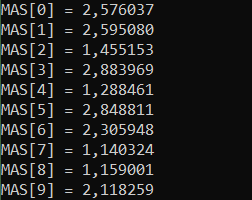


Рис. 23. Печать после сброса.

Проведем замер времени у оставшихся видов сортировок.

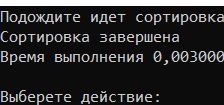


Рис. 24. Время на сортировку вставками.



Рис. 25. Время на быструю сортировку.

Путем проведения экспериментов было выявлено и зафиксировано, что сортировка «Быстрая» самая быстрая из трех

Таблица времени

|  |  |  |
| --- | --- | --- |
| Вид сортировки | Кол-во элементов в файле | Время сортировки |
| Пузырьком | 1000 | 0,40000 |
| Вставками | 1000 | 0,03000 |
| Быстрая | 1000 | 0,00000 |

# 5. 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 “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».

# **6. 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

http://codeforces.com/

# 7. Приложение

## 7.1 Приложение 1

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <locale.h>

#include <Windows.h>

#include <time.h>

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

int main() {

int x = 0, i = 0, case\_1 = 0, j;

int size\_gener, len\_inter\_keyboard;

double min\_gener, max\_gener;

double inter\_keyboard;

double\* mas = 0;

int n;

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

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

do

{

system("cls");

printf("Меню \"Сортировка\". Выбирете действие?\n");

printf("1. Сгенерировать числа\n2. Ввести с клавиатуры\n3. Копирование из файла\n");

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

switch (x)

{

case 1:

do

{

system("cls");

printf("Вы выбрали \"Генерацию чисел\"\n");

printf("Введите кол-во элементов массива: ");

scanf\_s("%d", &size\_gener);

printf("Введите минимальное число: ");

scanf\_s("%lf", &min\_gener);

printf("Введите максимальное число: ");

scanf\_s("%lf", &max\_gener);

printf("\nПроверка...\n\n");

Sleep(1500);

if ((min\_gener > max\_gener) || (size\_gener <= 0))

{

printf("Вы ввели не правильные данные. Введите снова\n");

Sleep(1000);

}

else

{

system("cls");

printf("Данные корректны.\nСейчас произойдет генерации...\n\n");

Sleep(1000);

case\_1++;

}

} while (case\_1 != 1);

printf("Ожидайте...\n\n");

Sleep(1000);

for (j = 0; j < size\_gener; j++)

{

fprintf(file, "%lf\n", (rand() / RAND\_MAX\_F) \* ((double)max\_gener - (double)min\_gener) + min\_gener);

}

printf("Генерация чисел произошла\n");

i++;

break;

case 2:

system("cls");

printf("Вы выбрали \"Ввести с клавиатуры\"\n");

printf("Пожалуйста, введите кол-во элементов массива: ");

scanf\_s("%d", &len\_inter\_keyboard);

printf("Пожалуйста, введите все нужные Вам числа (числа вводить с запятой, а не с точкой. К примеру: 6,66):\n");

for (j = 0; j < len\_inter\_keyboard; j++)

{

scanf\_s("%lf", &inter\_keyboard);

fprintf(file, "%f\n", inter\_keyboard);

}

printf("\nОжидайте...\n\n");

Sleep(1000);

printf("Числа записаны в текстовый файл\n");

i++;

break;

case 3:

system("cls");

printf(" Вы выбрали копирование массива из файла\n");

printf(" Массив успешно скопирован из файла\n");

FILE\* file1 = 0;

fopen\_s(&file1,"text1.txt", "r");

printf("Сколько элементов содержит файл?\n");

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

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

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

{

mas[i] = 0;

}

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

{

fscanf\_s(file1, "%lf\n", &(mas[i]));

}

fclose(file1);

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

{

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

}

fclose(file);

i++;

break;

default:

printf("Вы ввели не правильное число, повторите попытку\n");

Sleep(1000);

break;

}

} while (i != 1);

return 0;

}

## 7.2 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>

#include<math.h>

clrscr()

{

system("@cls||clear");

}

void bSort(double\* arr1, int size)

{

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

int i, j;

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(double\* arr2, int size)

{

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

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(double\* arr3, int start, int end)

{

int left = start;

int right = end;

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);

}

int the\_first\_norm\_of\_vector(int size\_file, double b[])

{

int answer;

double vichisl = 0;

for (int k = 0; k < size\_file; k++)

{

vichisl += fabs(pow(b[k], 1));

}

printf\_s("\nНорма вектора = %lf\n\n", pow(vichisl, (1 / 1)));

printf("\nХотите подсчитать другую норму фектора?\n");

printf\_s("1. Да\n2. Нет\n");

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

if (answer == 1)

return 0;

else

return 1;

return 0;

}

int the\_second\_norm\_of\_vector(int size\_file, double b[])

{

int answer;

double vichisl = 0;

vichisl = fabs(b[size\_file - 1]);

printf("\nНорма вектора = %lf\n\n", vichisl);

printf("\nХотите подсчитать другую норму фектора?\n");

printf\_s("1. Да\n2. Нет\n");

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

if (answer == 1)

return 0;

else

return 1;

return 0;

}

int the\_Helder\_norm\_of\_vector(int size\_file, double b[])

{

int answer;

double sum = 0;

for (int index = 0; index < size\_file; index++)

{

sum += pow(abs(b[index]), size\_file);

}

printf("\nГельдерова норма вектора равна: %lf\n", pow(sum, 1 / size\_file));

printf("\nХотите подсчитать другую норму фектора?\n");

printf\_s("1. Да\n2. Нет\n");

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

if (answer == 1)

return 0;

else

return 1;

return 0;

}

int infinite\_norm\_of\_vector(int size\_file, double b[])

{

double max = 0;

int answer;

for (int index = 0; index < size\_file; index++)

{

if (b[index] > max)

{

max = abs(b[index]);

}

}

printf("\nБесконечная норма вектора равна: %lf\n", max);

printf("\nХотите подсчитать другую норму фектора?\n");

printf\_s("1. Да\n2. Нет\n");

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

if (answer == 1)

return 0;

else

return 1;

return 0;

}

int normalization\_of\_the\_array\_vector(int size\_file, double b[])

{

int answer;

double sum = 0, sum\_sqrt;

for (int index = 0; index < size\_file; index++)

{

sum += pow(abs(b[index]), 2);

}

sum\_sqrt = sqrt(sum);

printf("\nНормировка вектора: {");

for (int index = 0; index < size\_file; index++)

{

printf("%lf, ", b[index] / sum\_sqrt);

}

printf("}\n");

printf("\nХотите подсчитать другую норму фектора?\n");

printf\_s("1. Да\n2. Нет\n");

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

if (answer == 1)

return 0;

else

return 1;

return 0;

}

const 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, x\_1, user\_run;

double startTime, endTime, time\_on\_prog;

double\* MAS, \* MAS1;

FILE\* file = fopen("C://Users//anton//source//repos//Project5//Project5//text.txt", "r");

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

printf\_s("Чтение из файла...\n\n");

size = stringCount(file);

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

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

do

{

printf("Выберете действие:\n1. Сортировка\n2. Распечатать числа\n3. Сброс\n4. Работа с массивом\n5. Выйти\n\nВЫБОР: ");

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

if (flag < 1 or flag > 5)

{

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

}

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(double) \* size);

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

}

if (flag == 4)

{

do

{

system("cls");

printf("1. Вычислить первую норму вектора\n2. Вычислить вторую норму вектора\n3. Вычислить Гельдерову норму вектора\n");

printf("4. Вычислить бесконечную норму вектора\n5. Нормировка вектора.\n");

scanf\_s("%d", &x\_1);

if (x\_1 < 1 or x\_1>5)

{

printf("Вы ввели не правильные данные. Повторите\n");

}

if (x\_1 == 1)

{

user\_run = the\_first\_norm\_of\_vector(size, MAS1);

}

if (x\_1 == 2)

{

user\_run = the\_second\_norm\_of\_vector(size, MAS1);

}

if (x\_1 == 3)

user\_run = the\_Helder\_norm\_of\_vector(size, MAS1);

if (x\_1 == 4)

user\_run = infinite\_norm\_of\_vector(size, MAS1);

if (x\_1 == 5)

user\_run = normalization\_of\_the\_array\_vector(size, MAS1);

} while (user\_run != 1);

}

if (flag == 5)

{

printf("Программа завершена\n");

return 0;

}

} while (flag == 1 or flag == 2 or flag == 3 or flag == 4 or flag == 5);

fclose(file);

free(MAS);

free(MAS1);

}