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

Нижний Новгород  
2021 г.

Содержание.

1. [Введение. 3](#_Введение.)
2. [Постановка задачи. 4](#_Toc86237500)
3. [Руководство пользователя. 5](#_Toc86237501)
4. Руководство программиста. 7
5. Эксперименты 12
6. [Заключение.](#_Toc86237504) 16
7. Литература 17
8. Приложение 19

# Введение.

Программирование — процесс создания компьютерных программ.

По выражению одного из основателей языков программирования Никлауса Вирта, «Программы = алгоритмы + структуры данных».

Программирование основывается на использовании языков программирования, на которых записываются исходные тексты программ.

Существуют десятки алгоритмов сортировки, каждый из которых хорош в чём-то своём. На одних легко объяснять принципы сортировки, другие хороши при работе с большими массивами, третьи оптимизированы по скорости, четвёртые — по числу процессорных циклов, компактности кода и т. д. В данной лабораторной работе на можно будет наглядно увидеть какие сортировки в чем хороши.

# Постановка задачи.

Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных “float”) 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 данных из исходного массива и ввод новых данных с клавиатуры, выход), выполняет выбранные действия.

# Руководство пользователя.

Используется две программы. После запуска первой программы выводится сообщение на экран: «Введите кол-во элементов в массиве». Далее на экран выводится сообщение: «Введите минимальный элемент». Пользователь должен ввести минимальное значение диапазона случайных чисел и нажать «Enter». Следующим шагом выводится сообщение: «Введите максимальный элемент». Необходимо ввести с клавиатуры максимальное значение диапазона случайных чисел и снова нажать клавишу «Enter». (см. рис. 1)

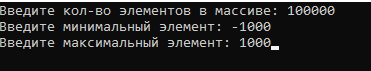


Рисунок 1. Ввод данных.

Если пользователь неверно введет данные программа предложит ввести их еще раз, до тех пор, пока данные не будут верны. (см. рис. 2)

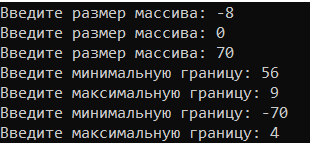


Рисунок 2. Ошибки при вводе данных.

После все данные будут записаны в файл:

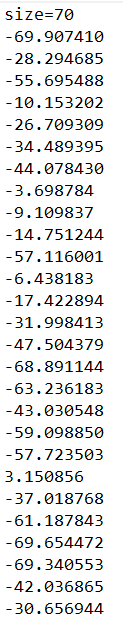


Рисунок 3. Записанные числа в файле.

После завершения первой программы следует запустить вторую. После ее запуска на экран будет выведен консольный интерфейс.

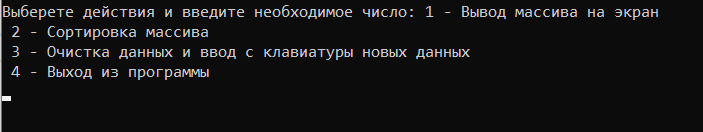


Рисунок 4. Консольный интерфейс.

Консольный интерфейс представляет собой набор из четырех команд:

1. Печать массива на экран

2. Сортировка массива

3. Ввод новых данных с клавиатуры

4. Выход

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

При выборе команды «Сортировка» пользователю предоставляется выбор типа сортировки.

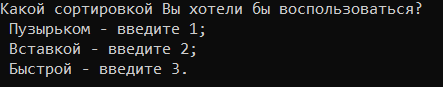


Рисунок 5. Типы сортировки.

После выполнения какой-либо сортировки на экран будет выведено сообщение о времени занимаемом этим типом сортировки.

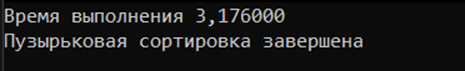


Рисунок 6. Время сортировки.

# 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 массив.**

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

{

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

}

**2. Алгоритм записи данных из массива в файл.**

FILE\* file = 0;

fopen\_s(&file, "file.txt", "w");

fprintf(file, "size=%d\n", size);

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

{

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

}

fclose(file);

**3. Алгоритм считывания чисел из файла.**

**4. Алгоритм консольного интерфейса.**

do {

printf\_s("Выберете действия и введите необходимое число: 1 - Вывод массива на экран\n 2 - Сортировка массива\n 3 - Очистка данных и ввод с клавиатуры новых данных\n 4 - Выход из программы\n");

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

switch (x) {

case 1:

Print(mas, size);

break;

case 2:

printf\_s("Какой сортировкой Вы хотели бы воспользоваться?\n Пузырьком - введите 1;\n Вставкой - введите 2;\n Быстрой - введите 3.\n");

do

{

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

if (y < 1 || y > 3)

printf\_s("\nВы неверно ввели номер. Введите снова\n");

} while (y < 1 || y > 3);

if (y == 1) {

startTime = clock();

BubbleSort(mas, size);

endTime = clock();

printf\_s("Время выполнения %lf\n", (endTime - startTime) / 1000);

printf\_s("Пузырьковая сортировка завершена\n");

Print(mas, size);

}

if (y == 2) {

startTime = clock();

InsertSort(mas, size);

endTime = clock();

printf\_s("Время выполнения %lf\n", (endTime - startTime) / 1000);

printf\_s("Cортировка вставкой завершена\n");

Print(mas, size);

}

if (y == 3) {

startTime = clock();

QuicklySort(mas, 0, size - 1);

endTime = clock();

printf\_s("Время выполнения %lf\n", (endTime - startTime) / 1000);

printf\_s("Быстрая сортировка завершена\n");

Print(mas, size);

}

break;

case 3:

memset(mas, 0, sizeof(float) \* size);

printf\_s("Введите новый размер массива: \n");

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

mas = (float\*)malloc(size \* sizeof(float));

printf\_s("Введите новые элементы массива: \n");

Scanf(mas, size);

break;

case 4:

break;

default:

printf\_s("Неправильный ввод данных\n");

}

} while (x != 4);

**5. Сортировки.**

**5.1. Сортировка пузырьком.**

void BubbleSort(float\* arr, int size) {

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

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

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

int tmp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = tmp;

}

}

}

}

**5.2. Сортировка вставкой.**

void InsertSort(float\* arr, int size) {

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

{

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

{

float tmp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = tmp;

}

}

}

**5.3. Быстрая сортировка.**

void QuicklySort(float\* arr, int l, int u) {

int i = l;

int j = u;

int tmp = 0;

int x = arr[(int)((l + u) / 2)];

do

{

while (arr[i] < x)

++i;

while (arr[j] > x)

--j;

if (i <= j)

{

tmp = arr[i];

arr[i] = arr[j];

arr[j] = tmp;

i++;

j--;

}

} while (i < j);

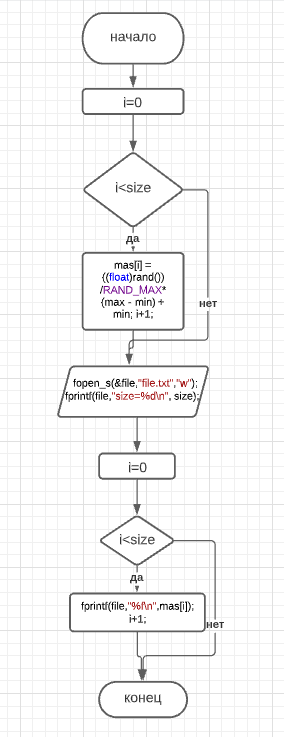
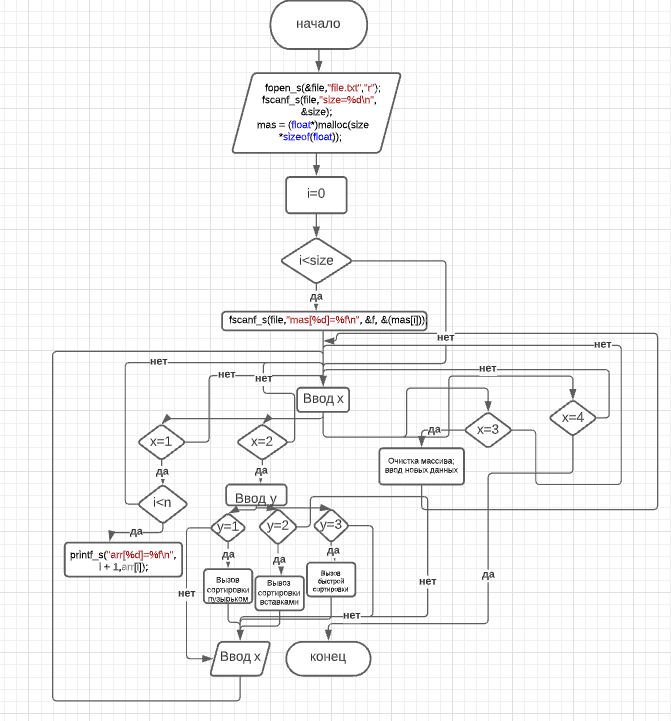
if (l < j)

QuicklySort(arr, l, j);

if (i < u)

QuicklySort(arr, i, u);

}

# Эксперименты.

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

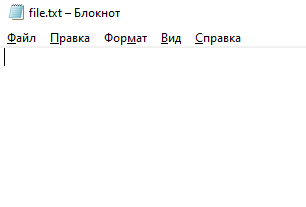


Рисунок 7. Чистый файл.

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



Рисунок 8. Выполнение программы.

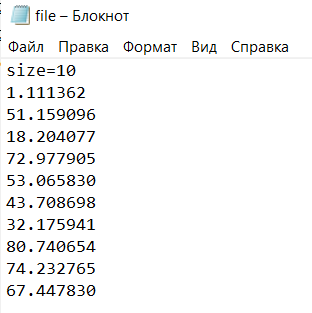


Рисунок 9. Записанные числа в файл.

Действительно, программа записала 10 случайных чисел в правильном диапазоне.

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

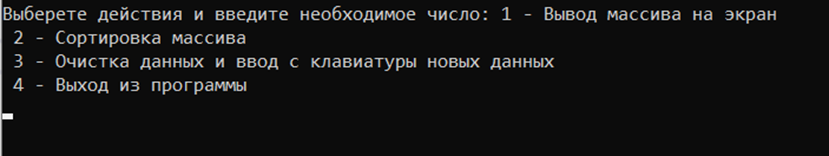


Рисунок 10. Консольный интерфейс.

Выберем команду «Вывод массива на экран». Для этого введем «1» и нажмем «Enter». На экран будут выведены числа, записанные в файле:

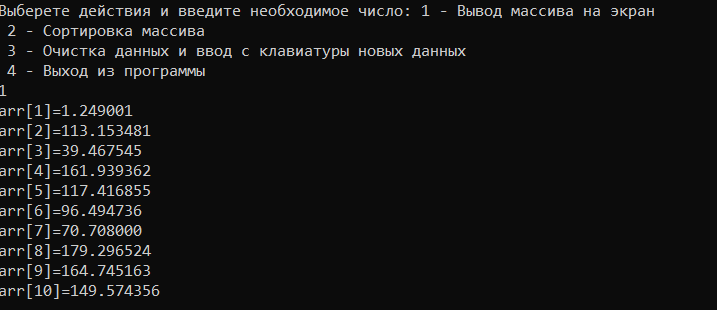


Рисунок 11. Печать.

Теперь отсортируем их методом пузырька и выведем результат:

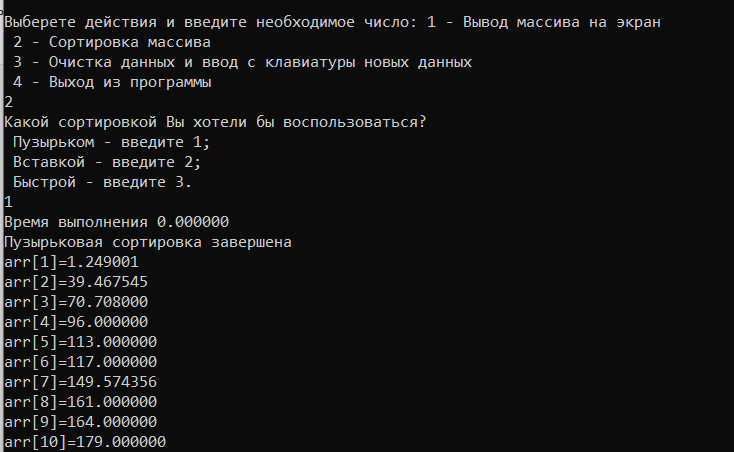


Рисунок 12. Сортировка пузырьком и вывод результата.

Теперь выберем команду «Очистка данных и ввод с клавиатуры новых данных», после «Вывод массива на экран»:

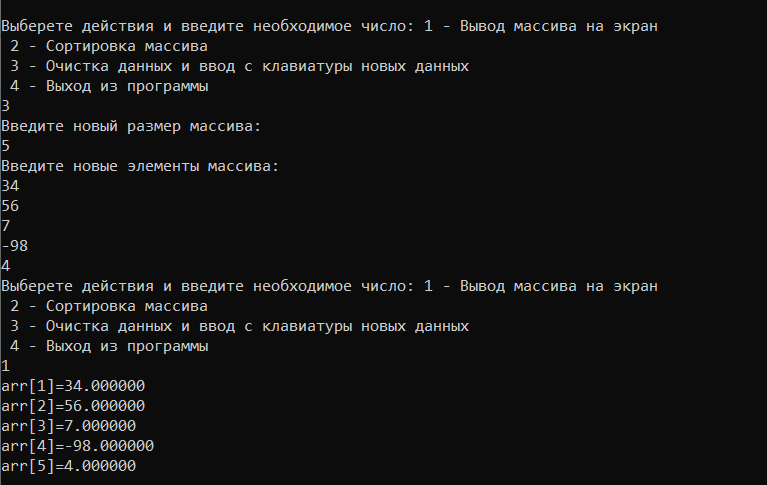


Рисунок 13. Ввод и вывод новых данных

Проверим оставшиеся две сортировки.

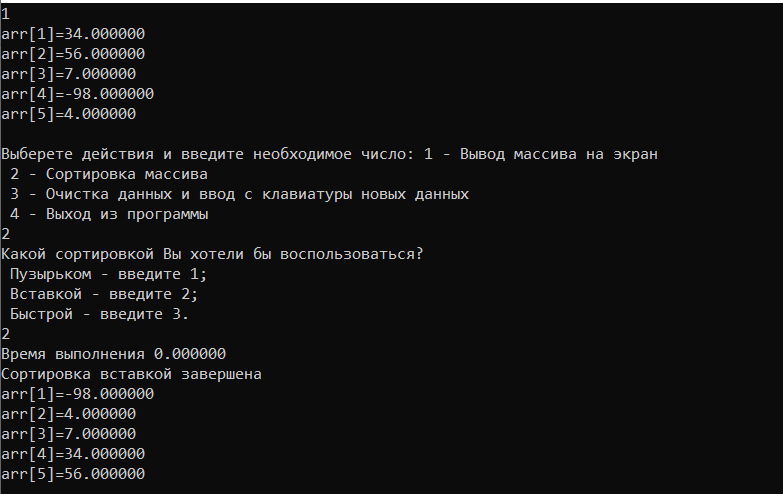


Рисунок 14. Сортировки.

Проверим программу на введение некорректного номера команды. Для этого введем номер, не присвоенный не одной из команд:

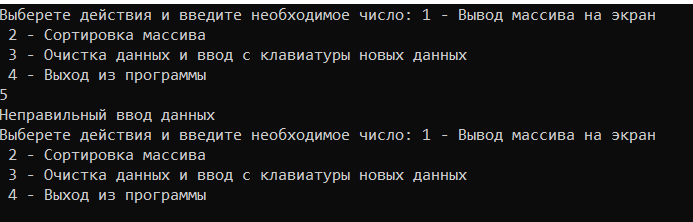


Рисунок 16. Некорректный номер команды.

Сравним скорости сортировок на 50000 чисел. (см. рис. 25, 26, 27)



Рисунок 18. Время работы сортировки пузырьком.



Рисунок 19. Время работы сортировки вставками.



Рисунок 20. Время работы быстрой сортировки.

# 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 “float”) задаваемых случайно.

Реализовать сортировки: пузырьком, вставкой, быстрая.

Сравнить время работы, сделать выводы.

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

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

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

7. 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>

# 8. Приложение.

Приложение 1.

#include <stdio.h>

#include <stdlib.h>

int main () {

system("chcp 1251");

system("cls");

int size = 0, max = 0, min = 0;

float\* mas = 0;

int count = 0;

do {

int ch = 0;

printf("Введите размер массива: ");

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

while (ch = getchar() != '\n');

} while ((count < 1) || (size < 0) || (size == 0));

mas = (float\*)malloc(size \* sizeof(float));

do {

do {

int ch = 0;

printf("Введите минимальную границу: ");

count = scanf\_s("%d", &min);

while (ch = getchar() != '\n');

} while (count < 1);

do {

int ch = 0;

printf("Введите максимальную границу: ");

count = scanf\_s("%d", &max);

while (ch = getchar() != '\n');

} while (count < 1);

} while ((min > max) || (min == max));

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

{

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

}

FILE\* file = 0;

fopen\_s(&file, "file.txt", "w");

fprintf(file, "size=%d\n", size);

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

{

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

}

fclose(file);

return 0;

}

Приложение 2.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <conio.h>

void Scanf(float\* arr, int size) {

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

scanf\_s("%f", &arr[i]);

}

}

void Print(float\* arr, int size) {

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

printf\_s("arr[%d]=%f\n", i + 1, arr[i]);

}

printf\_s("\n");

}

void BubbleSort(float\* arr, int size) {

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

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

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

int tmp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = tmp;

}

}

}

}

void InsertSort(float\* arr, int size) {

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

{

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

{

float tmp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = tmp;

}

}

}

void QuicklySort(float\* arr, int l, int u) {

int i = l;

int j = u;

int tmp = 0;

int x = arr[(int)((l + u) / 2)];

do

{

while (arr[i] < x)

++i;

while (arr[j] > x)

--j;

if (i <= j)

{

tmp = arr[i];

arr[i] = arr[j];

arr[j] = tmp;

i++;

j--;

}

} while (i < j);

if (l < j)

QuicklySort(arr, l, j);

if (i < u)

QuicklySort(arr, i, u);

}

int main() {

system("chcp 1251");

system("cls");

double startTime, endTime;

int size=0;

float\* mas=0;

FILE\* file = 0;

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

fscanf\_s(file, "size=%d\n", &size);

mas = (float\*)malloc(size \* sizeof(float));

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

{

mas[i] = 0;

}

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

{

int f = 0;

fscanf\_s(file, "mas[%d]=%f\n", &f, &(mas[i]));

}

fclose(file);

int x = 0;

int y = 0;

do {

printf\_s("Выберете действия и введите необходимое число: 1 - Вывод массива на экран\n 2 - Сортировка массива\n 3 - Очистка данных и ввод с клавиатуры новых данных\n 4 - Выход из программы\n");

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

switch (x) {

case 1:

Print(mas, size);

break;

case 2:

printf\_s("Какой сортировкой Вы хотели бы воспользоваться?\n Пузырьком - введите 1;\n Вставкой - введите 2;\n Быстрой - введите 3.\n");

do

{

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

if (y < 1 || y > 3)

printf\_s("\nВы неверно ввели номер. Введите снова\n");

} while (y < 1 || y > 3);

if (y == 1) {

startTime = clock();

BubbleSort(mas, size);

endTime = clock();

printf\_s("Время выполнения %lf\n", (endTime - startTime) / 1000);

printf\_s("Пузырьковая сортировка завершена\n");

Print(mas, size);

}

if (y == 2) {

startTime = clock();

InsertSort(mas, size);

endTime = clock();

printf\_s("Время выполнения %lf\n", (endTime - startTime) / 1000);

printf\_s("Cортировка вставкой завершена\n");

Print(mas, size);

}

if (y == 3) {

startTime = clock();

QuicklySort(mas, 0, size - 1);

endTime = clock();

printf\_s("Время выполнения %lf\n", (endTime - startTime) / 1000);

printf\_s("Быстрая сортировка завершена\n");

Print(mas, size);

}

break;

case 3:

memset(mas, 0, sizeof(float) \* size);

printf\_s("Введите новый размер массива: \n");

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

mas = (float\*)malloc(size \* sizeof(float));

printf\_s("Введите новые элементы массива: \n");

Scanf(mas, size);

break;

case 4:

break;

default:

printf\_s("Неправильный ввод данных\n");

}

} while (x != 4);

}