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  
Вьюнов Д.В.  
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 интересный, полезный и увлекательный процесс, благодаря которому мы, с помощью специальных команд, заставляем компьютер, выполнять для нас различные задачи, от выполнения операций с числами и навигации, до управления самолетами, спутниками и прочей техникой.

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.

# 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.

# Руководство пользователя.

Используется две программы. После запуска первой программы выводится сообщение на экран: «Выберите способ задания». (см. рис. 1)

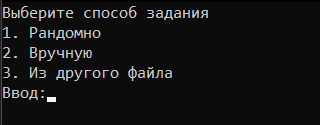


Рисунок 1.Выбор способа ввода.

При выборе рандомного способа задания следует ввести с клавиатуры кол-во случайных чисел, которые в дальнейшем будут записаны в файл. После ввода необходимо нажать клавишу «Enter». 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. 2)

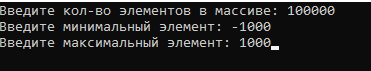


Рисунок 2. Ввод значений.

Если пользователь введет отрицательное значение для кол-ва случайных чисел, на экран будет выведено сообщение: «Вы неверно ввели данные. Введите снова». (см. рис. 3)

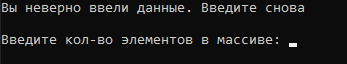


Рисунок 3. Ошибка в кол-ве случайных чисел.

Если же пользователь введет такие значения для минимального и максимального чисел, что максимальное будет меньше минимального, то на экран будет выведено следующее сообщение: «Вы неверно ввели данные. Введите снова». (см. рис. 4)

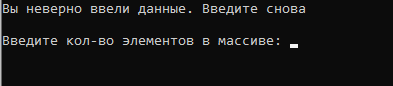


Рисунок 4. Ошибка в диапазоне.

При выборе второго способа задания на экран выведется сообщение «Введите кол-во чисел». (см. рис. 5)

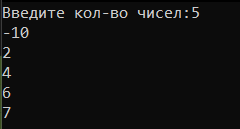


Рисунок 5. Способ задания вручную.

При выборе третьего способа на экран выведется сообщение «Массив задан». (см. рис. 6)

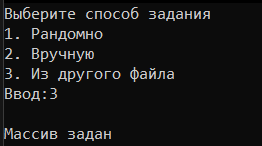


Рисунок 6. Способ задания из другого файла.

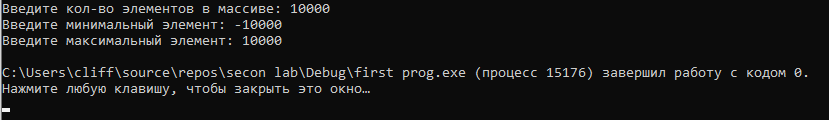


Рисунок 7. Завершение первой программы.

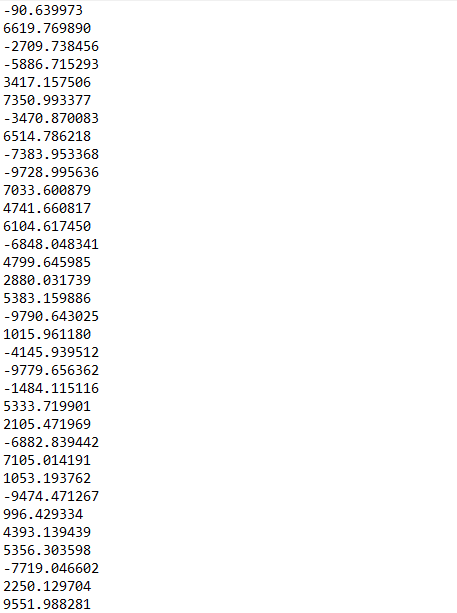


Рисунок 8. Записанные числа в файле.

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

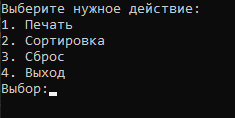


Рисунок 9. Консольный интерфейс.

Консольный интерфейс представляет собой набор из четырех команд:

1. Печать

2. Сортировка

3. Сброс

4. Выход

* Команда «Печать» осуществляет вывод на экран чисел, записанных в файле.
* Команда «Сортировка» сортирует числа.
* 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. 10)

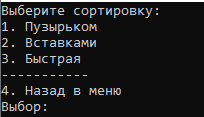


Рисунок 10. Типы сортировки.

После выполнения какой-либо сортировки на экран будет выведено сообщение о времени занимаемом этим типом сортировки. (см. рис. 11)

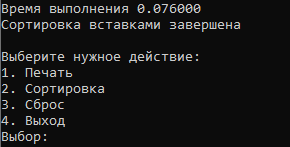


Рисунок 11. Время сортировки.

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. 12)

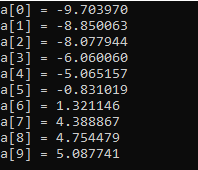


Рисунок 12. Отсортированные числа.

# 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("file.txt", "w");

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

{

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

}

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

FILE\* file = fopen("../first prog/file.txt", "r");

double\* b, \* a;

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

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

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

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

**3. Алгоритм консольного интерфейса.**

clrscr();

memcpy(b, a, sizeof(double) \* size);

do

{

printf\_s("Выберите нужное действие:\n1. Печать\n2. Сортировка\n3. Сброс\n4. Выход\nВыбор:");

do

{

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

if (checkbox < 1 || checkbox > 4)

{

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

}

} while (checkbox < 1 || checkbox > 4);

if (checkbox == 1)

{

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

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

printf\_s("\n");

}

else if (checkbox == 2)

{

clrscr();

printf\_s("Выберите сортировку:\n1. Пузырьком\n2. Вставками\n3. Быстрая\n-----------\n4. Назад в меню\nВыбор:");

do

{

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

if (sortCheck < 1 || sortCheck > 4)

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

} while (sortCheck < 1 || sortCheck > 4);

if (sortCheck == 1)

{

clrscr();

startTime = clock();

bubbleSort(b, size);

endTime = clock();

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

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

memcpy(b, a, sizeof(double) \* size);

}

else if (sortCheck == 2)

{

clrscr();

startTime = clock();

insertionSort(b, size);

endTime = clock();

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

printf\_s("Сортировка вставками завершена\n\n");

memcpy(b, a, sizeof(double) \* size);

}

else if (sortCheck == 3)

{

clrscr();

startTime = clock();

quickSort(b, 0, size - 1);

endTime = clock();

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

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

memcpy(b, a, sizeof(double) \* size);

}

else

{

clrscr();

}

}

else if (checkbox == 3)

{

clrscr();

}

else

{

printf\_s("\nВы вышли из программы\n");

return 0;

}

} while (checkbox == 3 || checkbox == 1 || checkbox == 2 || checkbox == 4);

**4. Сортировки.**

**4.1. Сортировка пузырьком.**

void bubbleSort(double\* arrbs, int size)

{

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

{

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

{

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

{

double buff = arrbs[j];

arrbs[j] = arrbs[j + 1];

arrbs[j + 1] = buff;

}

}

}

}

**4.2. Сортировка вставкой.**

void insertionSort(double\* mas, int size)

{

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

{

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

{

double buff = mas[j];

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

mas[j - 1] = buff;

}

}

}

**4.3. Быстрая сортировка.**

void quickSort(double\* mas, int start, int end)

{

int left = start;

int right = end;

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

while (left <= right)

{

while (mas[left] < pivot)

{

left++;

}

while (mas[right] > pivot)

{

right--;

}

if (left <= right)

{

double buff = mas[left];

mas[left] = mas[right];

mas[right] = buff;

left++;

right--;

}

}

if (start < right)

quickSort(mas, start, right);

if (end > left)

quickSort(mas, left, end);

}

**Блок-схема 1. Алгоритм записи случайных чисел в файл.**

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

i++

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

True

i < n

i = 0

False

True

a = 0

a = (double\*)malloc(size \* sizeof(double))

**Блок-схема 2.** **Алгоритм считывания чисел из файла.**

a = (double\*)malloc(size \* sizeof(double))

True

i < size

i++

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

i = 0

return 1

False

a == 0

# Эксперименты.

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

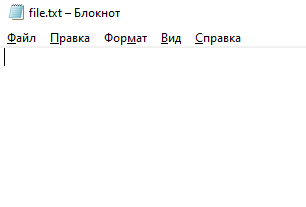


Рисунок 13. Чистый файл.

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

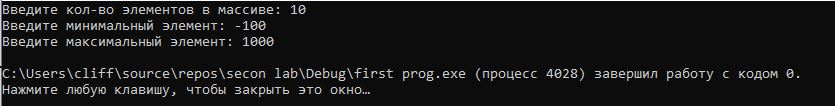


Рисунок 14. Выполнение программы.

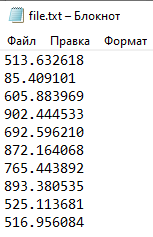


Рисунок 14. Записанные числа в файл.

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:



Рисунок 15. Некорректное кол-во чисел.



Рисунок 16. Некорректный диапазон.

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

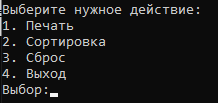


Рисунок 17. Консольный интерфейс.

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

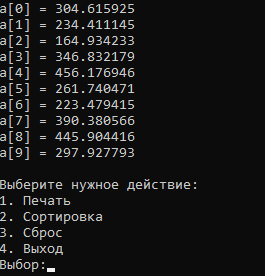
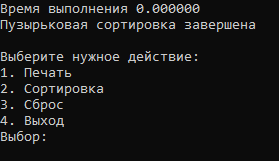


Рисунок 18. Печать.

Теперь отсортируем их методом пузырька и выведем результат:



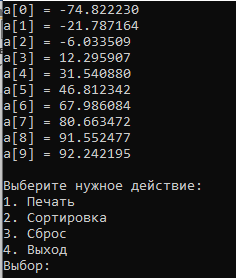
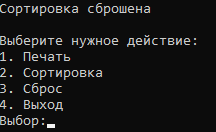


Рисунок 19. Сортировка пузырьком и вывод результата.

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



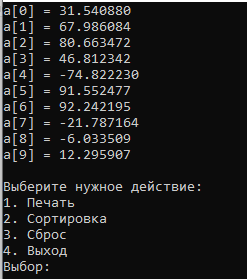
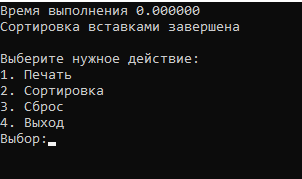


Рисунок 20. Сброс сортировки.

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



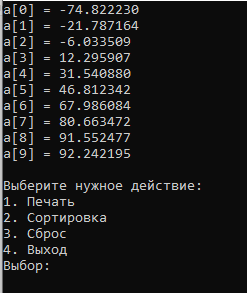
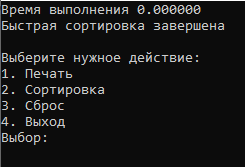


Рисунок 21. Сортировка вставками и вывод результата.



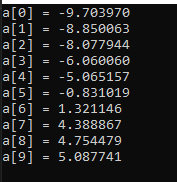


Рисунок 22. Быстрая сортировка и вывод результата.

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

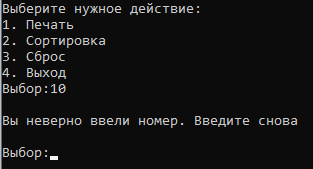


Рисунок 23. Некорректный номер команды.

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

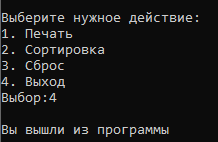


Рисунок 24. Завершение работы программы.

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



Рисунок 25. Время работы сортировки пузырьком.



Рисунок 26. Время работы сортировки вставками.



Рисунок 27. 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 “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 (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».

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>

# Приложение.

Приложение 1.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

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

void clrscr()

{

system("@cls||clear");

}

int main()

{

srand(time(NULL));

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

int size, max, min;

do

{

printf("Введите кол-во элементов в массиве: ");

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

printf("Введите минимальный элемент: ");

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

printf("Введите максимальный элемент: ");

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

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

{

clrscr();

printf\_s("Вы неверно ввели данные. Введите снова\n\n");

}

} while (size <= 0 || min > max);

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

{

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

}

fclose(file);

}

Приложение 2.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <conio.h>

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

int stringCount(FILE\* strC) {

int result = 0;

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

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

result++;

}

rewind(strC);

return result;

}

void clrscr()

{

system("@cls||clear");

}

void bubbleSort(double\* arrbs, int size)

{

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

{

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

{

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

{

double buff = arrbs[j];

arrbs[j] = arrbs[j + 1];

arrbs[j + 1] = buff;

}

}

}

}

void quickSort(double\* mas, int start, int end)

{

int left = start;

int right = end;

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

while (left <= right)

{

while (mas[left] < pivot)

{

left++;

}

while (mas[right] > pivot)

{

right--;

}

if (left <= right)

{

double buff = mas[left];

mas[left] = mas[right];

mas[right] = buff;

left++;

right--;

}

}

if (start < right)

quickSort(mas, start, right);

if (end > left)

quickSort(mas, left, end);

}

void insertionSort(double\* mas, int size)

{

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

{

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

{

double buff = mas[j];

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

mas[j - 1] = buff;

}

}

}

int main()

{

srand(time(NULL));

FILE\* file = fopen("../first prog/file.txt", "r");

int size, checkbox = 0, sortCheck = 0;

double startTime, endTime;

printf\_s("Чтение из файла\n");

size = stringCount(file);

double\* b, \* a;

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

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

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

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

clrscr();

memcpy(b, a, sizeof(double) \* size);

do

{

printf\_s("Выберите нужное действие:\n1. Печать\n2. Сортировка\n3. Сброс\n4. Выход\nВыбор:");

do

{

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

if (checkbox < 1 || checkbox > 4)

{

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

}

} while (checkbox < 1 || checkbox > 4);

if (checkbox == 1)

{

clrscr();

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

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

printf\_s("\n");

}

else if (checkbox == 2)

{

clrscr();

printf\_s("Выберите сортировку:\n1. Пузырьком\n2. Вставками\n3. Быстрая\n-----------\n4. Назад в меню\nВыбор:");

do

{

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

if (sortCheck < 1 || sortCheck > 4)

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

} while (sortCheck < 1 || sortCheck > 4);

if (sortCheck == 1)

{

clrscr();

startTime = clock();

bubbleSort(b, size);

endTime = clock();

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

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

}

else if (sortCheck == 2)

{

clrscr();

startTime = clock();

insertionSort(b, size);

endTime = clock();

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

printf\_s("Сортировка вставками завершена\n\n");

}

else if (sortCheck == 3)

{

clrscr();

startTime = clock();

quickSort(b, 0, size - 1);

endTime = clock();

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

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

}

else

{

clrscr();

}

}

else if (checkbox == 3)

{

clrscr();

memcpy(b, a, sizeof(double) \* size);

printf\_s("Сортировка сброшена\n\n");

}

else

{

printf\_s("\nВы вышли из программы\n");

return 0;

}

} while (checkbox == 3 || checkbox == 1 || checkbox == 2 || checkbox == 4);

fclose(file);

free(a);

free(b);

}