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. Взаимодействие двух программ.

Выполнил:

студент группы 3821Б1ФИ3

Васин А.В.

Проверил:

заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г

Нижний Новгород 2021г.

# Содержание

[Содержание 1](#_Toc88648576)

[Введение 2](#_Toc88648577)

[Постановка задачи 3](#_Toc88648578)

[Руководство пользователя 4](#_Toc88648579)

[Первая программа 4](#_Toc88648580)

[Вторая программа 5](#_Toc88648581)

[Руководство программиста 8](#_Toc88648582)

[*Описание структуры кода программ* 8](#_Toc88648583)

[Первая программа 8](#_Toc88648584)

[Вторая программа 11](#_Toc88648585)

[Описание структур данных 16](#_Toc88648586)

[Описание алгоритмов 16](#_Toc88648587)

[Сортировка «пузырьком» 16](#_Toc88648588)

[Сортировка «вставками» 16](#_Toc88648589)

[Быстрая сортировка 17](#_Toc88648590)

[Эксперименты 20](#_Toc88648591)

[Заключение 21](#_Toc88648592)

[Список литературы 22](#_Toc88648593)

[Приложение 1 23](#_Toc88648594)

[Приложение 2 26](#_Toc88648594)

# Введение

**Си** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *C*) — [компилируемый](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) [статически типизированный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [язык 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) общего назначения, разработанный в 1969—1973 годах сотрудником [Bell Labs](https://ru.wikipedia.org/wiki/Bell_Labs" \o "Bell Labs) [Деннисом Ритчи](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D1%82%D1%87%D0%B8,_%D0%94%D0%B5%D0%BD%D0%BD%D0%B8%D1%81" \o "Ритчи, Деннис) как развитие языка [Би](https://ru.wikipedia.org/wiki/%D0%91%D0%B8_(%D1%8F%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)). Первоначально был разработан для реализации [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) [UNIX](https://ru.wikipedia.org/wiki/UNIX), но впоследствии был [перенесён](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) на множество других платформ. Согласно дизайну языка, его конструкции близко сопоставляются типичным [машинным инструкциям](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), благодаря чему он нашёл применение в проектах, для которых был свойственен [язык ассемблера](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0), в том числе как в [операционных системах](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), так и в различном [прикладном программном обеспечении](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) для множества устройств — от [суперкомпьютеров](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D0%B5%D1%80%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) до [встраиваемых систем](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D1%82%D1%80%D0%B0%D0%B8%D0%B2%D0%B0%D0%B5%D0%BC%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). Язык программирования Си оказал существенное влияние на развитие индустрии программного обеспечения, а его [синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%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%D0%B5)) стал основой для таких языков программирования, как [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [C#](https://ru.wikipedia.org/wiki/C_Sharp), [Java](https://ru.wikipedia.org/wiki/Java" \o "Java) и [Objective-C](https://ru.wikipedia.org/wiki/Objective-C" \o "Objective-C).

**Алгоритм сортировки** — это [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) для упорядочивания элементов в массиве. В случае, когда элемент в массиве имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма. В данной лабораторной работе была продемонстрирована работа трёх сортировок: «Сортировка пузырьком», «Быстрая сортировка», и «Сортировка вставкой».

# Постановка задачи

Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных – unsigned 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 возможность запуска сортировки через параметры командной строки.  
Добавить вычисление первой номы вектора (массива);  
Добавить вычисление второй нормы вектора (массива);  
Добавить вычисление Гельдеровой нормы вектора (массива);  
Добавить вычисление бесконечной нормы вектора (массива);  
Добавить нормировку вектора (массива);

# Руководство пользователя

## Первая программа

После запуска первой программы будет отображено меню программы, в котором необходимо выбрать нужный вам пункт (см рис 1.).

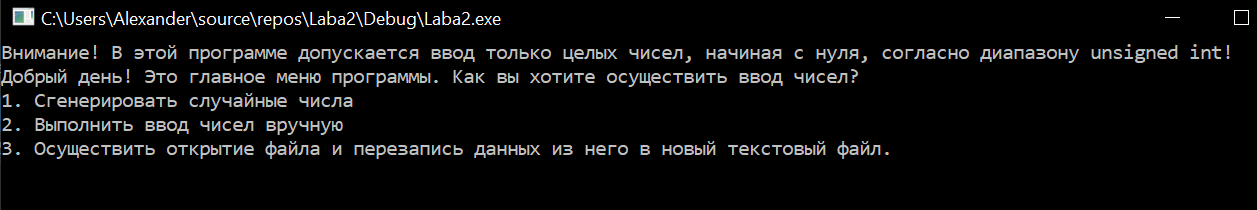


Рис. 1 Программа после запуска

Если пользователь выберет первый пункт, ему высветится меню генерации массива (см. рис. 2).

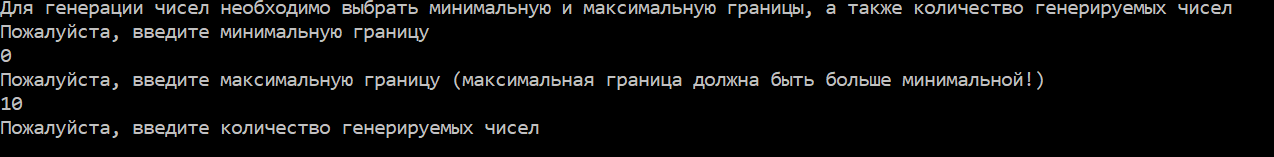


Рис. 2 Генерация массива из случайных чисел.

В случае выбора второго варианта, пользователь сможет ввести элементы самостоятельно (см рис 3.)

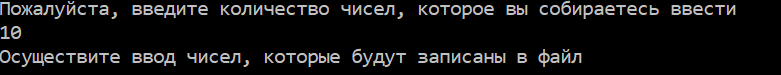


Рис. 3 Ввод массива с клавиатуры.

Если пользователь выберет третий вариант, то будет произведено копирование из файла «input.txt» (ваш файл, который нужно переименовать в это название и переместить в корневую папку программы) (см. рис. 4).

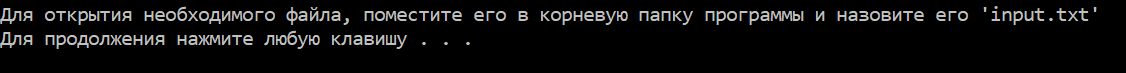


Рис. 4 Копирование массива из файла

В результате работы программы будет создан текстовый файл под названием «massive.txt», наполненный элементами созданного массива (см. рис. 5).

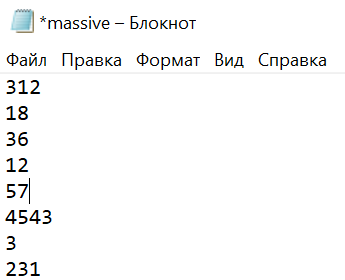


Рис. 5 Текстовый файл с сгенерированными числами

## Вторая программа

При запуске второй программы перед пользователем будет выведено окно меню, в котором ему необходимо выбрать нужное действие (см. рис. 6).

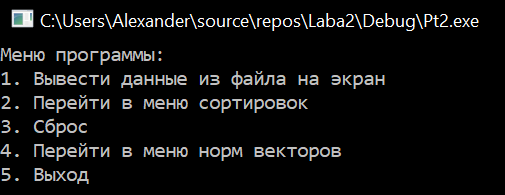


Рис.6 Меню программы

Если пользователь выберет первый вариант, то на экране будут напечатаны все элементы массива из текстового файла (см. рис.7).

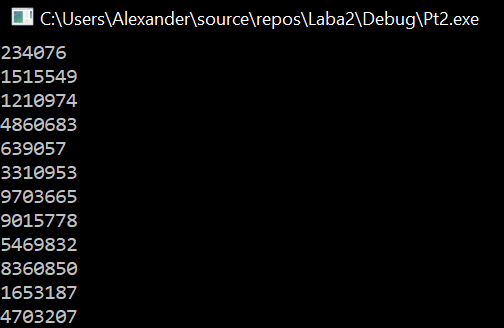


Рис. 7 Печать массива

Если пользователь выберет второй вариант, то он попадёт в меню выбора сортировок (см. рис. 8).

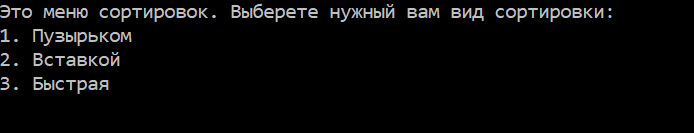


Рис. 8 Выбор сортировки

При выборе любой сортировки, на экран будет выводиться процесс работы, чтобы можно было наглядно увидеть всю работу алгоритма. И, в конце программы, откроется изначальное меню (см. рис. 9).

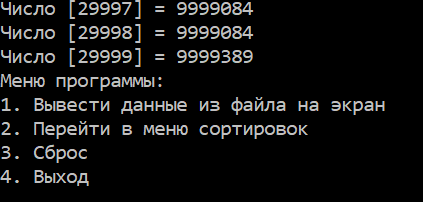


Рис. 9 Процесс работы

Также при выполнении сортировки, в отдельный файл в корневой папке под названием «ftime.txt» ведётся запись времени в миллисекундах, за которое осуществилась сортировка (см. рис. 10).

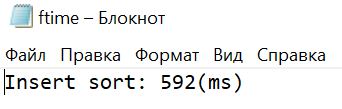


Рис. 10 Время сортировки вставкой массива из 30000 элементов.

При выборе третьего пункта меню будет произведён сброс последней сортировки, массив будет возвращён в изначальное состояние.

При выборе четвёртого пункта, пользователь попадёт в меню норм векторов, где ему будет предложено осуществить вычисления норм и нормировки векторов (см. рис. 10.1)

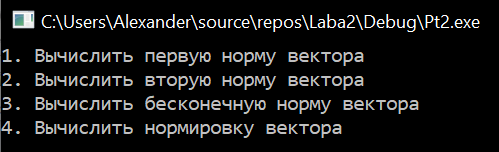


Рис. 10.1 Меню норм векторов.

Выход из программы осуществляется вводом цифры «5» в главном меню.

# Руководство программиста

**Описание структуры кода программ**

**Первая программа**

Подключение библиотек:

#include <stdlib.h>

#include <stdio.h>

#include <Windows.h>

#include <locale.h>

Объявление функции main() и всех переменных:

int main()

{

setlocale(LC\_ALL, "Russian");

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

int menu, n, x, y, z, count; // xzy - побочные

int i;

unsigned int min, max, keyboard;

unsigned int\* Massive = 0;

unsigned int\* Mass = 0;

char \* fpath;

char \* str;

Вывод главного меню программы:

printf("Внимание! В этой программе допускается ввод только целых чисел, начиная с нуля, согласно диапазону unsigned int!\n");

printf("Добрый день! Это главное меню программы. Как вы хотите осуществить ввод чисел?\n");

printf("1. Сгенерировать случайные числа\n");

printf("2. Выполнить ввод чисел вручную\n");

printf("3. Осуществить открытие файла и перезапись данных из него в новый текстовый файл.\n");

scanf("%d", &menu);

Операторы блоки кода, вызываемые в зависимости от выбора пользователя:

if (menu == 1)

{

printf("Для генерации чисел необходимо выбрать минимальную и максимальную границы, а также количество генерируемых чисел\n");

printf("Пожалуйста, введите минимальную границу\n");

scanf("%d", &min);

if (min < 0) goto err;

printf("Пожалуйста, введите максимальную границу (максимальная граница должна быть больше минимальной!)\n");

scanf("%d", &max);

if (max >= 4294967295)

if (max < min) goto err;

printf("Пожалуйста, введите количество генерируемых чисел\n");

scanf("%d", &n);

if (n < 0) goto err;

system("cls");

printf("Генерация чисел начинается. Процесс работы будет выводится на экран.");

Sleep(5000);

system("cls");

Massive = (unsigned int\*)malloc(n \* sizeof(unsigned int));

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

{

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

}

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

{

printf("Число %d = %d\n", i, Massive[i]);

}

printf("Генерация прошла успешно. Чтобы записать полученные данные в файл, нажмите '1', для закрытия пограммы нажмите '2'\n");

scanf("%d", &x);

if (x == 1) // GEN

{

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

{

fprintf(file, "%d\n", Massive[i]);

}

if (file != 0)

{

printf("Запись произошла успешно!\n");

goto stop;

}

else

{

goto err;

}

}

if (x == 2)

{

goto stop;

}

else

{

goto err;

}

}

if (menu == 2) // KLAVA

{

printf("Пожалуйста, введите количество чисел, которое вы собираетесь ввести\n");

scanf("%d", &n);

if (n < 0) goto err;

printf("Осуществите ввод чисел, которые будут записаны в файл\n");

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

{

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

fprintf(file, "%d\n", keyboard);

}

printf("Запись прошла успешно!\n");

goto stop;

}

if (menu == 3) // CHTENIE

{

printf("Для открытия необходимого файла, поместите его в корневую папку программы и назовите его 'input.txt'\n");

//scanf\_s(" %c", &fpath);

system("pause");

fpath = 0;

FILE\* file1 = fopen("..\\input.txt", "r");

if (file1 == 0) goto err;

printf("Запись в файл 'massive.txt' началась!\n");

Sleep(1000);

while (!feof(file1))

{

count = fread(&str, sizeof(char), 1, file1);

if (count != 0)

fwrite(&str, sizeof(char), 1, file);

}

if (file != 0)

{

system("cls");

printf("Запись прошла успешно!\n");

}

else

{

goto err;

}

}

else

{

err:

system("cls");

printf("Произошла неизвестная ошибка! Закрытие программы...\n");

Sleep(1000);

goto stop;

}

stop:

fclose(file);

free(Massive);

system("pause");

return 0;

}

**Вторая программа**

Подключение библиотек:

#include <stdlib.h>

#include <stdio.h>

#include <Windows.h>

#include <locale.h>

#include <malloc.h>

#include <time.h>

#include <math.h>

#include <stdbool.h>

Объявление функций:

void quickSort(int \*numbers, int left, int right)

{

int i, j;

int tmp, pivot;

i = left;

j = right;

pivot = numbers[(left + (right - left) / 2)];

do {

while (numbers[i] < pivot) {

i++;

}

while (numbers[j] > pivot) {

j--;

}

if (i <= j) {

if (numbers[i] > numbers[j]) {

tmp = numbers[i];

numbers[i] = numbers[j];

numbers[j] = tmp;

}

i++;

if (j > 0) {

j--;

}

}

} while (i <= j);

if (i < right) {

quickSort(numbers, i, right);

}

if (j > left) {

quickSort(numbers, left, j);

}

}

void norma1(unsigned int\* numbers, int size)

{

int i = 0, kol = 0;

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

{

kol += numbers[i];

}

printf("Первая норма вектора = %d\n", kol);

}

void norma2(unsigned int\* numbers, int size)

{

int i = 0, kol = 0;

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

{

kol += pow(numbers[i], 2);

}

kol = sqrt(kol);

printf("Вторая норма вектора = %d\n", kol);

}

void normaInfinity(unsigned int\* numbers, int size)

{

int i = 0, last = 0;

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

{

if (last < numbers[i])

{

last = numbers[i];

}

}

printf("Бесконечная норма вектора = %d\n", last);

}

void normir(unsigned int\* numbers, int size)

{

int i = 0;

double k = 0, x = 0;

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

{

x += pow(numbers[i], 2);

}

x = sqrt(x);

x = round(x);

printf("Результат нормировки:\n");

Sleep(2000);

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

{

k = numbers[i] / x;

printf("%lf\n", k);

}

}

Объявление переменных и функция main():

int main()

{

menu:

setlocale(LC\_ALL, "Russian");

clock\_t fTimeStart1, fTimeStop1;

clock\_t fTimeStart2, fTimeStop2;

clock\_t fTimeStart3, fTimeStop3;

int m, m2, i, j, N, y, k, x;

char q, s;

FILE\* file = 0;

unsigned char \* fdata;

unsigned int \* mass = 0;

unsigned int \* mass2 = 0;

N = 0;

x = 0;

Вывод меню программы:

printf("Меню программы:\n");

printf("1. Вывести данные из файла на экран\n");

printf("2. Перейти в меню сортировок\n");

printf("3. Сброс\n");

printf("4. Выход\n");

scanf("%d", &m);

system("cls");

Блок, в котором происходит перевод данных из файла в массив, а также подсчёт количества элементов:

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

while (!feof(file))

{

if (fgetc(file) == '\n') N++;

}

fclose(file);

mass = (unsigned int\*)malloc(N \* sizeof(unsigned int));

mass2 = (unsigned int\*)malloc(N \* sizeof(unsigned int));

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

{

mass[i] = 0;

}

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

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

{

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

}

fclose(file);

Блок кода, в котором происходит вызов необходимых групп команд, исходя из выбора пользователя. В него включено функция выхода из программы, вывод элемента массива из текстового файла на экран, меню сортировок, алгоритмы вставочной и пузырьковой сортировки (быстрая вынесена в отдельную функцию), а также меню нормировок:

if (m == 5) goto stop;

if (m == 1)

{

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

while ((q = fgetc(file)) != EOF)

{

putchar(q);

}

fclose(file);

printf("Вывод произошёл успешно!\n");

goto menu;

}

if (m == 2)

{

printf("Это меню сортировок. Выберете нужный вам вид сортировки:\n");

printf("1. Пузырьком\n");

printf("2. Вставкой\n");

printf("3. Быстрая\n");

scanf("%d", &m2);

if (m2 == 1)

{

fTimeStart2 = clock() \* 1000 / CLOCKS\_PER\_SEC;

int tmp;

bool noSwap;

for (int i = N - 1; i >= 0; i--)

{

noSwap = 1;

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

{

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

{

tmp = mass[j];

mass[j] = mass[j + 1];

mass[j + 1] = tmp;

noSwap = 0;

}

}

if (noSwap == 1)

break;

}

fTimeStop2 = clock() \* 1000 / CLOCKS\_PER\_SEC;

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

fprintf(ftime, "Bubble sort: %i(ms)", fTimeStop2 - fTimeStart2);

fclose(ftime);

printf("Результат обработки:\n");

Sleep(3000);

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

{

printf("Число [%d] = %d\n", i, mass[i]);

}

goto menu;

}

if (m2 == 2)

{

int newElement, location;

fTimeStart3 = clock() \* 1000 / CLOCKS\_PER\_SEC;

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

{

newElement = mass[i];

location = i - 1;

while (location >= 0 && mass[location] > newElement)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = newElement;

}

fTimeStop3 = clock() \* 1000 / CLOCKS\_PER\_SEC;

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

fprintf(ftime, "Insert sort: %i(ms)", fTimeStop3 - fTimeStart3);

fclose(ftime);

printf("Результат обработки:\n");

Sleep(3000);

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

{

printf("Число [%d] = %d\n", i, mass[i]);

}

goto menu;

}

if (m2 == 3)

{

fTimeStart1 = clock() \* 1000 / CLOCKS\_PER\_SEC;

quickSort(mass, 0, N - 1);

fTimeStop1 = clock() \* 1000 / CLOCKS\_PER\_SEC;

FILE\* ftime1 = fopen("..\\ftime.txt", "w");

fprintf(ftime1, "Quick sort: %i(ms)", fTimeStop1 - fTimeStart1);

fclose(ftime1);

printf("Результат обработки:\n");

Sleep(3000);

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

{

printf("Число [%d] = %d\n", i, mass[i]);

}

goto menu;

}

}

if (m == 3)

{

memcpy(mass, mass2, sizeof(unsigned int) \* N);

goto menu;

}

if (m == 4)

{

printf("1. Вычислить первую норму вектора\n");

printf("2. Вычислить вторую норму вектора\n");

printf("3. Вычислить бесконечную норму вектора\n");

printf("4. Вычислить нормировку вектора\n");

scanf\_s("%d", &m3);

if (m3 < 1 || m3 > 4)

{

goto err;

}

if (m3 == 1) norma1(mass, N);

if (m3 == 2) norma2(mass, N);

if (m3 == 3) normaInfinity(mass, N);

if (m3 == 4) normir(mass, N);

}

else

{

err:

system("cls");

printf("Произошла неизвестная ошибка! Закрытие программы...\n");

Sleep(1000);

goto stop;

}

stop:

free(mass);

free(mass2);

system("pause");

return 0;

}

**Описание структур данных**

Все действия программ крутятся вокруг создания динамического массива. Разумеется, существует несколько способов его задания: вручную, генерацией чисел и копирование из существующего файла. Все основные действия происходят в диапазоне unsigned int, однако счётчики используют обычный int. При написании этих программ были использованы библиотеки “stdlib.h”, “stdio.h”, “Windows.h”, “locale.h”, “malloc.h”, “time.h”, “stdbool.h”, “time.h”

**Описание алгоритмов**

Прежде всего, происходит создание разными способами динамического массива первой программой и запись его в отдельный текстовый файл. Вторая программа читает этот файл и создаёт свой динамический массив, основываясь на готовых данных. Далее происходят все необходимые действия, в зависимости от выбора пользователя, а именно сортировка этого массива тремя разными способами и подсчёт времени их работы. Ниже будет приведены блоки кода использованных сортировок:

«Сортировка пузырьком»

Этот алгоритм сортировки является одним из самых распространённых из-за своей простоты, однако он является и наименее эффективным.

int tmp;

bool noSwap;

for (int i = N - 1; i >= 0; i--)

{

noSwap = 1;

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

{

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

{

tmp = mass[j];

mass[j] = mass[j + 1];

mass[j + 1] = tmp;

noSwap = 0;

}

}

if (noSwap == 1)

break;

«Сортировка вставками»

Сортировка вставками по сложности примерно такой же, как и «Пузырьковая сортировка», однако такой вид сортировки куда быстрее и эффективнее

int newElement, location;

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

{

newElement = mass[i];

location = i - 1;

while (location >= 0 && mass[location] > newElement)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = newElement;

}

«Быстрая сортировка»

Быстрая сортировка – совершенно непохожий способ сортировки на два предыдущих. Его особенность заключается в его быстроте, и, следовательно, в его сложности. Этот вид сортировки превосходит в скорости предыдущие два в разы, но, однако, он отличается своей нестабильностью.

void quickSort(int \*numbers, int left, int right)

{

int i, j;

int tmp, pivot;

i = left;

j = right;

pivot = numbers[(left + (right - left) / 2)];

do {

while (numbers[i] < pivot) {

i++;

}

while (numbers[j] > pivot) {

j--;

}

if (i <= j) {

if (numbers[i] > numbers[j]) {

tmp = numbers[i];

numbers[i] = numbers[j];

numbers[j] = tmp;

}

i++;

if (j > 0) {

j--;

}

}

} while (i <= j);

if (i < right) {

quickSort(numbers, i, right);

}

if (j > left) {

quickSort(numbers, left, j);

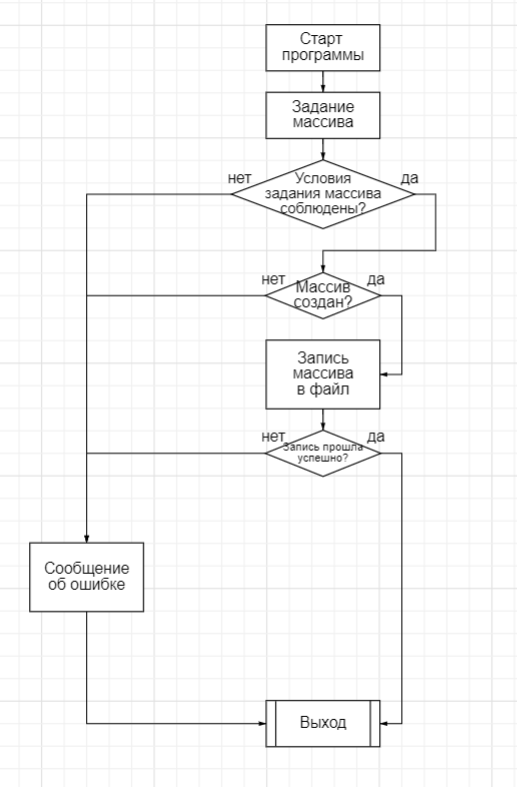
}

}

**Блок схема 1.**

**Базовый принцип работы**

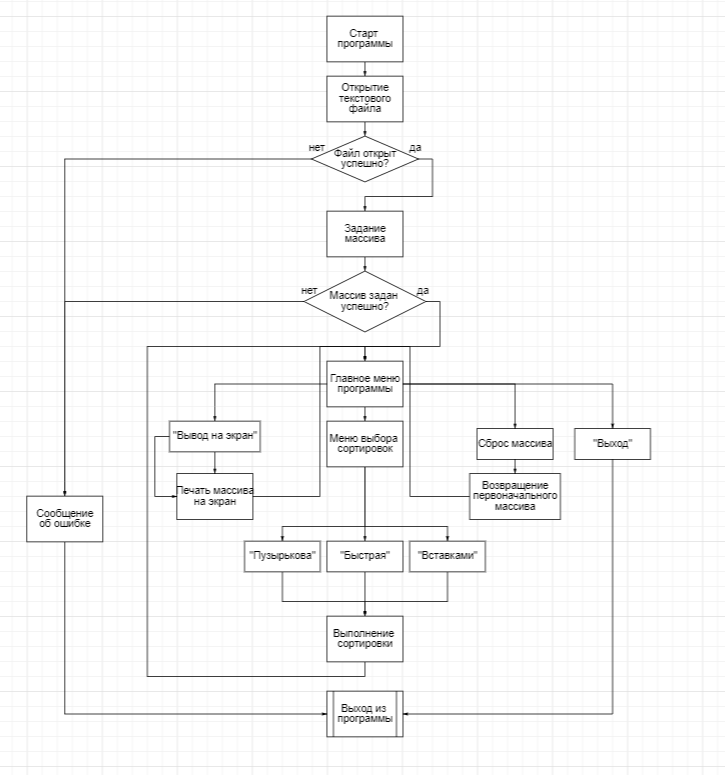
**первой программы**



**Блок-схема 2.**

**Базовый принцип работы**

**второй программы.**



**Эксперименты**

Можно провести несколько экспериментов над второй программой. А именно можно было бы замерить скорость той или иной сортировки при разных объёмах массива.

Прежде всего, сгенерируем первой программой десять тысяч чисел, а затем пятьдесят тысяч. (см рис. 11)

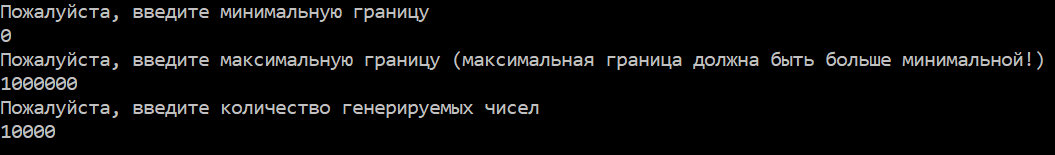


Рис. 11 Выбор условий генерации чисел

Чтобы было удобнее сравнивать, запишем результаты в таблицу:

|  |  |  |
| --- | --- | --- |
| *Вид сортировки* | *Количество элементов* | *Время сортировки (мс)* |
| «Пузырьковая» | 10000 | 256 |
| «Вставками» | 10000 | 67 |
| «Быстрая» | 10000 | 1 |
|  |  |  |
| *Вид сортировки* | *Количество элементов* | *Время сортировки (мс)* |
| «Пузырьковая» | 50000 | 7588 |
| «Вставками» | 50000 | 1727 |
| «Быстрая» | 50000 | 7 |

**Заключение**

В ходе работы на языке «С» было написано всего две программы, взаимодействующие друг с другом. Первая программа генерирует массив чисел и записывает их в текстовый файл, а вторая читает этот файл и производит необходимые сортировки с замером времени. И были проведено пару экспериментов с замером времени этих сортировок. Исходя из их результатов, можно сказать, что «Быстрая сортировка» превосходит все остальные в несколько раз по скорости, а «Пузырьковая» наоборот отстаёт от первых двух.

**Список литературы:**

* Лекции по программированию <https://github.com/LebedevIlyaG/3821>
* Брайан Керниган, Деннис Ритчи. Язык программирования C. — Москва: Вильямс, 2015.

**Приложение 1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <Windows.h>

#include <locale.h>

int main()

{

setlocale(LC\_ALL, "Russian");

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

int menu, n, x, y, z, count; // xzy - побочные

int i;

unsigned int min, max, keyboard;

unsigned int\* Massive = 0;

unsigned int\* Mass = 0;

char \* fpath;

char \* str;

if (file == 0) goto err;

printf("Внимание! В этой программе допускается ввод только целых чисел, начиная с нуля, согласно диапазону unsigned int!\n");

printf("Добрый день! Это главное меню программы. Как вы хотите осуществить ввод чисел?\n");

printf("1. Сгенерировать случайные числа\n");

printf("2. Выполнить ввод чисел вручную\n");

printf("3. Осуществить открытие файла и перезапись данных из него в новый текстовый файл.\n");

scanf("%d", &menu);

if (menu == 1)

{

printf("Для генерации чисел необходимо выбрать минимальную и максимальную границы, а также количество генерируемых чисел\n");

printf("Пожалуйста, введите минимальную границу\n");

scanf("%d", &min);

if (min < 0) goto err;

printf("Пожалуйста, введите максимальную границу (максимальная граница должна быть больше минимальной!)\n");

scanf("%d", &max);

if (max >= 4294967295)

if (max < min) goto err;

printf("Пожалуйста, введите количество генерируемых чисел\n");

scanf("%d", &n);

if (n < 0) goto err;

system("cls");

printf("Генерация чисел начинается. Процесс работы будет выводится на экран.");

Sleep(5000);

system("cls");

Massive = (unsigned int\*)malloc(n \* sizeof(unsigned int));

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

{

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

}

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

{

printf("Число %d = %d\n", i, Massive[i]);

}

printf("Генерация прошла успешно. Чтобы записать полученные данные в файл, нажмите '1', для закрытия пограммы нажмите '2'\n");

scanf("%d", &x);

if (x == 1) // GEN

{

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

{

fprintf(file, "%d\n", Massive[i]);

}

if (file != 0)

{

printf("Запись произошла успешно!\n");

goto stop;

}

else

{

goto err;

}

}

if (x == 2)

{

goto stop;

}

else

{

goto err;

}

}

if (menu == 2) // KLAVA

{

printf("Пожалуйста, введите количество чисел, которое вы собираетесь ввести\n");

scanf("%d", &n);

if (n < 0) goto err;

printf("Осуществите ввод чисел, которые будут записаны в файл\n");

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

{

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

fprintf(file, "%d\n", keyboard);

}

printf("Запись прошла успешно!\n");

goto stop;

}

if (menu == 3) // CHTENIE

{

printf("Для открытия необходимого файла, поместите его в корневую папку программы и назовите его 'input.txt'\n");

//scanf\_s(" %c", &fpath);

system("pause");

fpath = 0;

FILE\* file1 = fopen("..\\input.txt", "r");

if (file1 == 0) goto err;

printf("Запись в файл 'massive.txt' началась!\n");

Sleep(1000);

while (!feof(file1))

{

count = fread(&str, sizeof(char), 1, file1);

if (count != 0)

fwrite(&str, sizeof(char), 1, file);

}

if (file != 0)

{

system("cls");

printf("Запись прошла успешно!\n");

}

else

{

goto err;

}

}

else

{

err:

system("cls");

printf("Произошла неизвестная ошибка! Закрытие программы...\n");

Sleep(1000);

goto stop;

}

stop:

fclose(file);

free(Massive);

system("pause");

return 0;

}

**Приложение 2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdlib.h>

#include <stdio.h>

#include <Windows.h>

#include <locale.h>

#include <malloc.h>

#include <time.h>

#include <stdbool.h>

#include <math.h>

void quickSort(int \*numbers, int left, int right)

{

int i, j;

int tmp, pivot;

i = left;

j = right;

pivot = numbers[(left + (right - left) / 2)];

do {

while (numbers[i] < pivot) {

i++;

}

while (numbers[j] > pivot) {

j--;

}

if (i <= j) {

if (numbers[i] > numbers[j]) {

tmp = numbers[i];

numbers[i] = numbers[j];

numbers[j] = tmp;

}

i++;

if (j > 0) {

j--;

}

}

} while (i <= j);

if (i < right) {

quickSort(numbers, i, right);

}

if (j > left) {

quickSort(numbers, left, j);

}

}

void norma1(unsigned int\* numbers, int size)

{

int i = 0, kol = 0;

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

{

kol += numbers[i];

}

printf("Первая норма вектора = %d\n", kol);

}

void norma2(unsigned int\* numbers, int size)

{

int i = 0, kol = 0;

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

{

kol += pow(numbers[i], 2);

}

kol = sqrt(kol);

printf("Вторая норма вектора = %d\n", kol);

}

void normaInfinity(unsigned int\* numbers, int size)

{

int i = 0, last = 0;

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

{

if (last < numbers[i])

{

last = numbers[i];

}

}

printf("Бесконечная норма вектора = %d\n", last);

}

void normir(unsigned int\* numbers, int size)

{

int i = 0;

double k = 0, x = 0;

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

{

x += pow(numbers[i], 2);

}

x = sqrt(x);

x = round(x);

printf("Результат нормировки:\n");

Sleep(2000);

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

{

k = numbers[i] / x;

printf("%lf\n", k);

}

}

int main()

{

menu:

setlocale(LC\_ALL, "Russian");

clock\_t fTimeStart1, fTimeStop1;

clock\_t fTimeStart2, fTimeStop2;

clock\_t fTimeStart3, fTimeStop3;

int m, m2, m3, i, j, N, y, k, x;

char q, s;

FILE\* file = 0;

unsigned char \* fdata;

unsigned int \* mass = 0;

unsigned int \* mass2 = 0;

N = 0;

x = 0;

printf("Меню программы:\n");

printf("1. Вывести данные из файла на экран\n");

printf("2. Перейти в меню сортировок\n");

printf("3. Сброс\n");

printf("4. Перейти в меню норм векторов\n");

printf("5. Выход\n");

scanf("%d", &m);

system("cls");

// Блок, в котором происходит перевод данных из файла в массив, а также подсчёт колличества элементов

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

while (!feof(file))

{

if (fgetc(file) == '\n') N++;

}

fclose(file);

mass = (unsigned int\*)malloc(N \* sizeof(unsigned int));

mass2 = (unsigned int\*)malloc(N \* sizeof(unsigned int));

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

{

mass[i] = 0;

}

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

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

{

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

}

fclose(file);

// Блок, в котором происходят действия, исходя из выбора в меню

if (m == 5) goto stop;

if (m == 1)

{

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

while ((q = fgetc(file)) != EOF)

{

putchar(q);

}

fclose(file);

printf("Вывод произошёл успешно!\n");

goto menu;

}

if (m == 2)

{

printf("Это меню сортировок. Выберете нужный вам вид сортировки:\n");

printf("1. Пузырьком\n");

printf("2. Вставкой\n");

printf("3. Быстрая\n");

scanf("%d", &m2);

if (m2 == 1)

{

fTimeStart2 = clock() \* 1000 / CLOCKS\_PER\_SEC;

int tmp;

bool noSwap;

for (int i = N - 1; i >= 0; i--)

{

noSwap = 1;

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

{

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

{

tmp = mass[j];

mass[j] = mass[j + 1];

mass[j + 1] = tmp;

noSwap = 0;

}

}

if (noSwap == 1)

break;

}

fTimeStop2 = clock() \* 1000 / CLOCKS\_PER\_SEC;

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

fprintf(ftime, "Bubble sort: %i(ms)", fTimeStop2 - fTimeStart2);

fclose(ftime);

printf("Результат обработки:\n");

Sleep(3000);

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

{

printf("Число [%d] = %d\n", i, mass[i]);

}

goto menu;

}

if (m2 == 2)

{

int newElement, location;

fTimeStart3 = clock() \* 1000 / CLOCKS\_PER\_SEC;

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

{

newElement = mass[i];

location = i - 1;

while (location >= 0 && mass[location] > newElement)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = newElement;

}

fTimeStop3 = clock() \* 1000 / CLOCKS\_PER\_SEC;

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

fprintf(ftime, "Insert sort: %i(ms)", fTimeStop3 - fTimeStart3);

fclose(ftime);

printf("Результат обработки:\n");

Sleep(3000);

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

{

printf("Число [%d] = %d\n", i, mass[i]);

}

goto menu;

}

if (m2 == 3)

{

fTimeStart1 = clock() \* 1000 / CLOCKS\_PER\_SEC;

quickSort(mass, 0, N - 1);

fTimeStop1 = clock() \* 1000 / CLOCKS\_PER\_SEC;

FILE\* ftime1 = fopen("..\\ftime.txt", "w");

fprintf(ftime1, "Quick sort: %i(ms)", fTimeStop1 - fTimeStart1);

fclose(ftime1);

printf("Результат обработки:\n");

Sleep(3000);

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

{

printf("Число [%d] = %d\n", i, mass[i]);

}

goto menu;

}

}

if (m == 3)

{

memcpy(mass, mass2, sizeof(unsigned int) \* N);

goto menu;

}

if (m == 4)

{

printf("1. Вычислить первую норму вектора\n");

printf("2. Вычислить вторую норму вектора\n");

printf("3. Вычислить бесконечную норму вектора\n");

printf("4. Вычислить нормировку вектора\n");

scanf\_s("%d", &m3);

if (m3 < 1 || m3 > 4)

{

goto err;

}

if (m3 == 1) norma1(mass, N);

if (m3 == 2) norma2(mass, N);

if (m3 == 3) normaInfinity(mass, N);

if (m3 == 4) normir(mass, N);

}

else

{

err:

system("cls");

printf("Произошла неизвестная ошибка! Закрытие программы...\n");

Sleep(1000);

goto stop;

}

stop:

free(mass);

free(mass2);

system("pause");

return 0;

}