Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Моделирование работы сортировщика файлов ОС»**

**Выполнил**:

студент групп 381803-1

Колесников Г. Р.

**Проверил**:

доцент кафедры МОСТ,к.т.н.,

Сысоев А.В.

Нижний Новгород

2018

Содержание

[Введение 3](#_Toc530730517)

[1. Постановка задачи 4](#_Toc530730518)

[2. Руководство пользователя 5](#_Toc530730519)

[3.1. Описание структуры программы 7](#_Toc530730520)

[3.2. Описаниеалгоритмов 7](#_Toc530730521)

[4. Результаты экспериментов 9](#_Toc530730522)

[Заключение 10](#_Toc530730523)

[Литература 11](#_Toc530730524)

[Приложение 12](#_Toc530730525)

# Введение

Что такое сортировка и зачем она нужна?

Под сортировкой элементов множества мы подразумеваем процесс перегруппировки этих элементов в некотором определенном порядке. Для числовых множеств, например, можно провести сортировку по убыванию или по возрастанию; множество слов можно расположить в алфавитном порядке и т.д. Очевидно, что если массив отсортирован по какому-то правилу, с ним удобнее работать. В таком массиве проще найти нужные элементы, совершать с ними какие-либо операции; отсортированные массивы проще сравнивать друг с другом, находить в них общие элементы. Существуют разные варианты сортировок, которые различаются принципов работы, скоростью. Знание скорости выполнения сортировок полезно для создания быстродействующих программ.

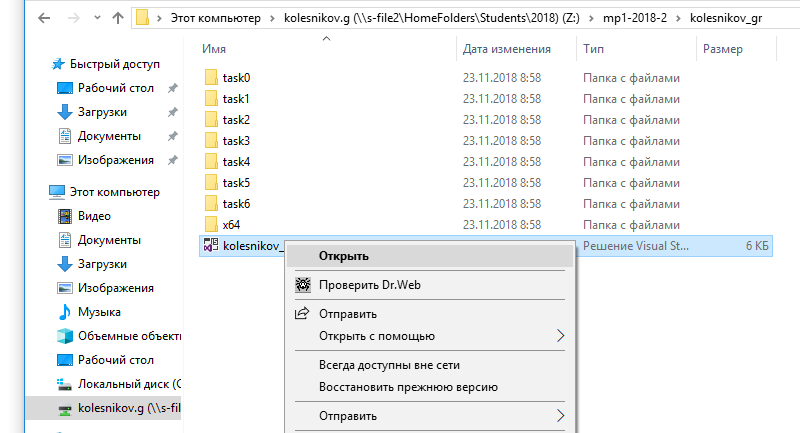
# Постановка задачи

Задача: Разработать программу, которая, получая на вход путь к каталогу с файлами, могла выводить эти файлы, отсортированные по их размеру. При этом программа должна уметь выполнять разные виды сортировок, а так же давать возможность изменять метод сортировки указанного каталога.

Дополнительно мы проведем сравнение скорости работы различных типов сортировок.

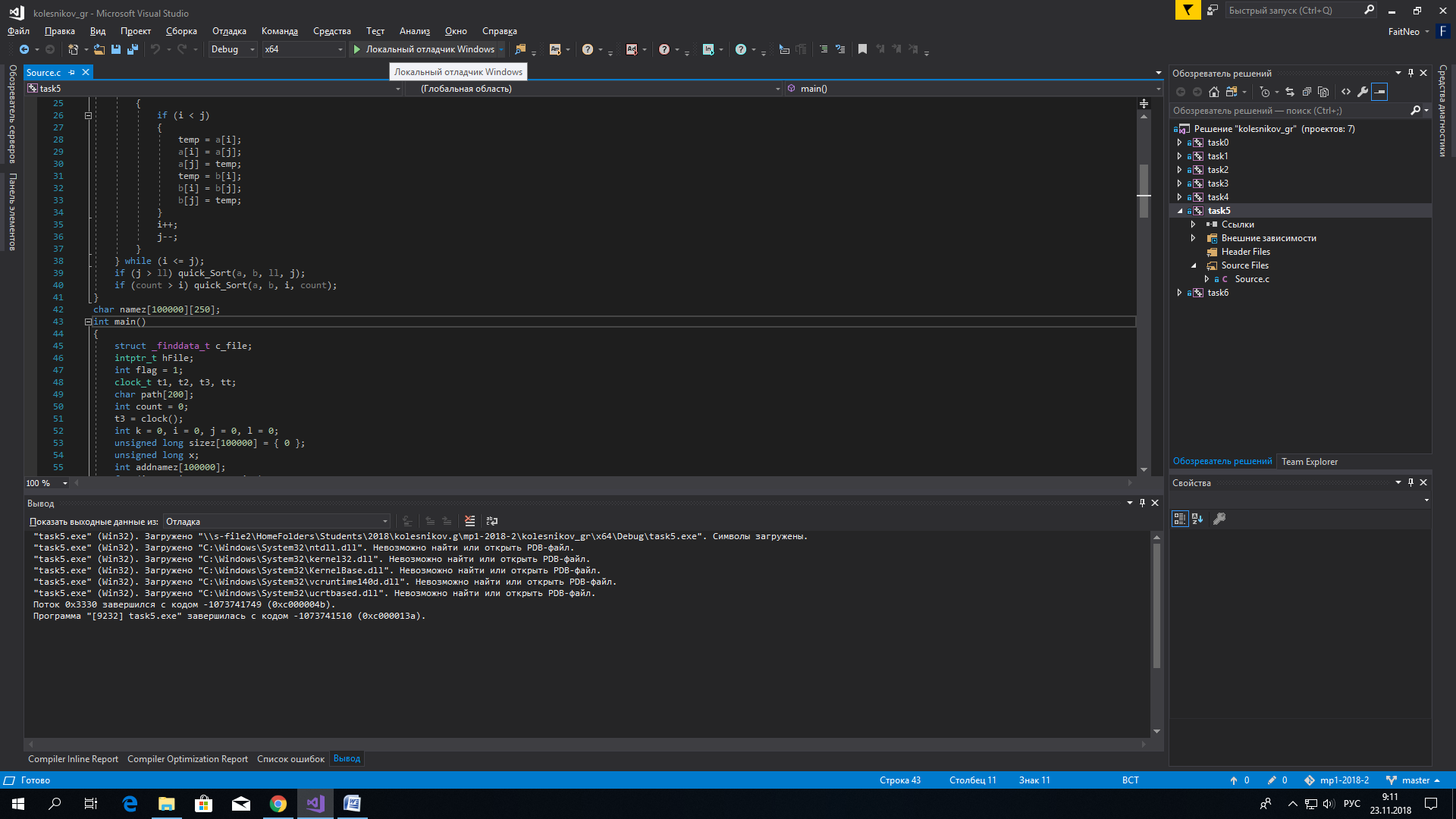
# Руководство пользователя

Во-первых, Вам нужно открыть программу, для этого нужно щелкнуть два раза по файлу kolesnikov\_gr левой кнопкой мыши, или правой кнопкой, после чего в появшемся окне нажать “Открыть”.

****

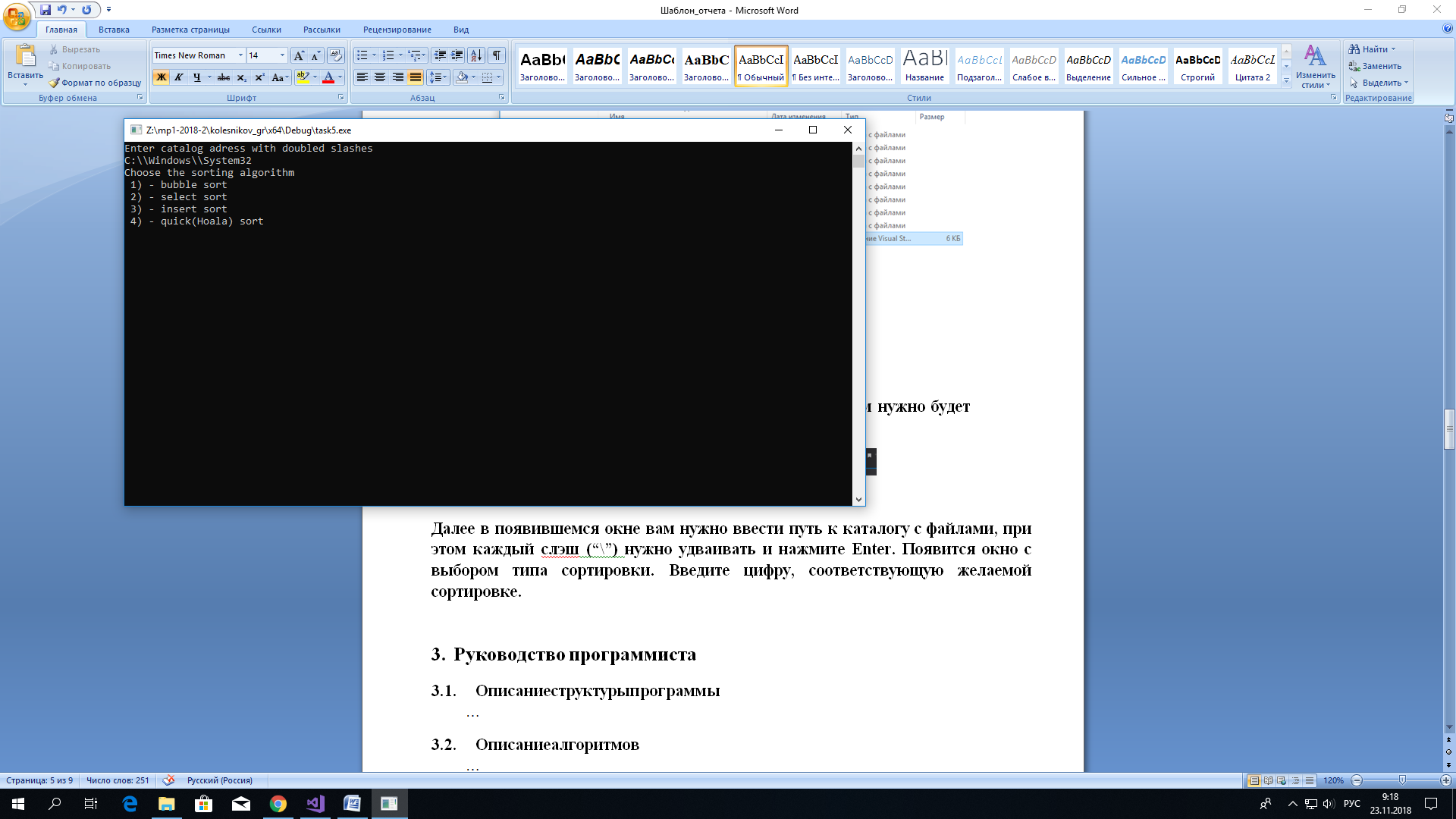
**Рис.1 Открытие программы**

После этого у вас откроется окно Visual Studio, в котором нужно будет нажать на кнопку “Локальный отладчик Windows”(убедившись перед этим, что исполняемым решением стоит task5).

****

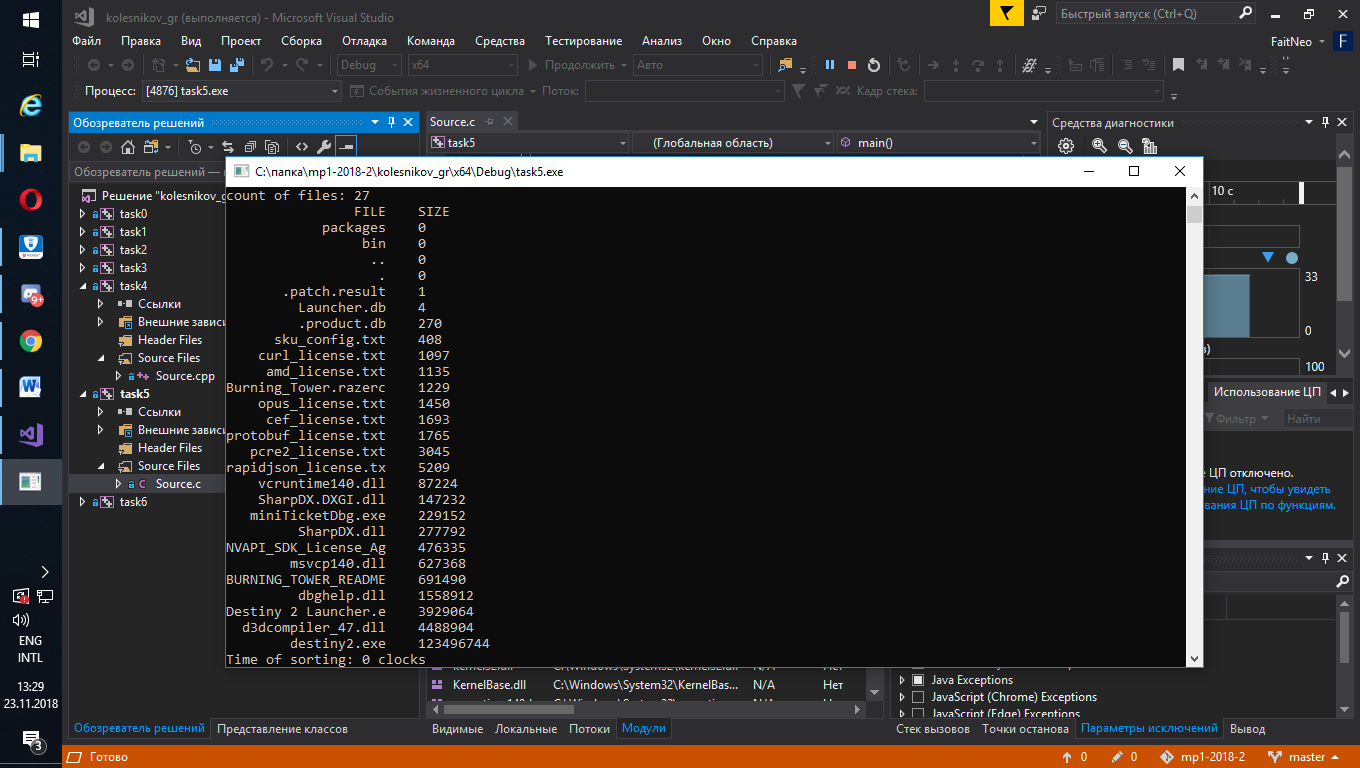
**Рис.2 Запуск программы**

Далее в появившемся окне вам нужно ввести путь к каталогу с файлами, при этом каждый слэш (“\”) нужно удваивать и нажмите Enter. Появится окно с выбором типа сортировки. Введите цифру, соответствующую желаемой сортировке.

****

**Рис.3 Выбор сортировки**

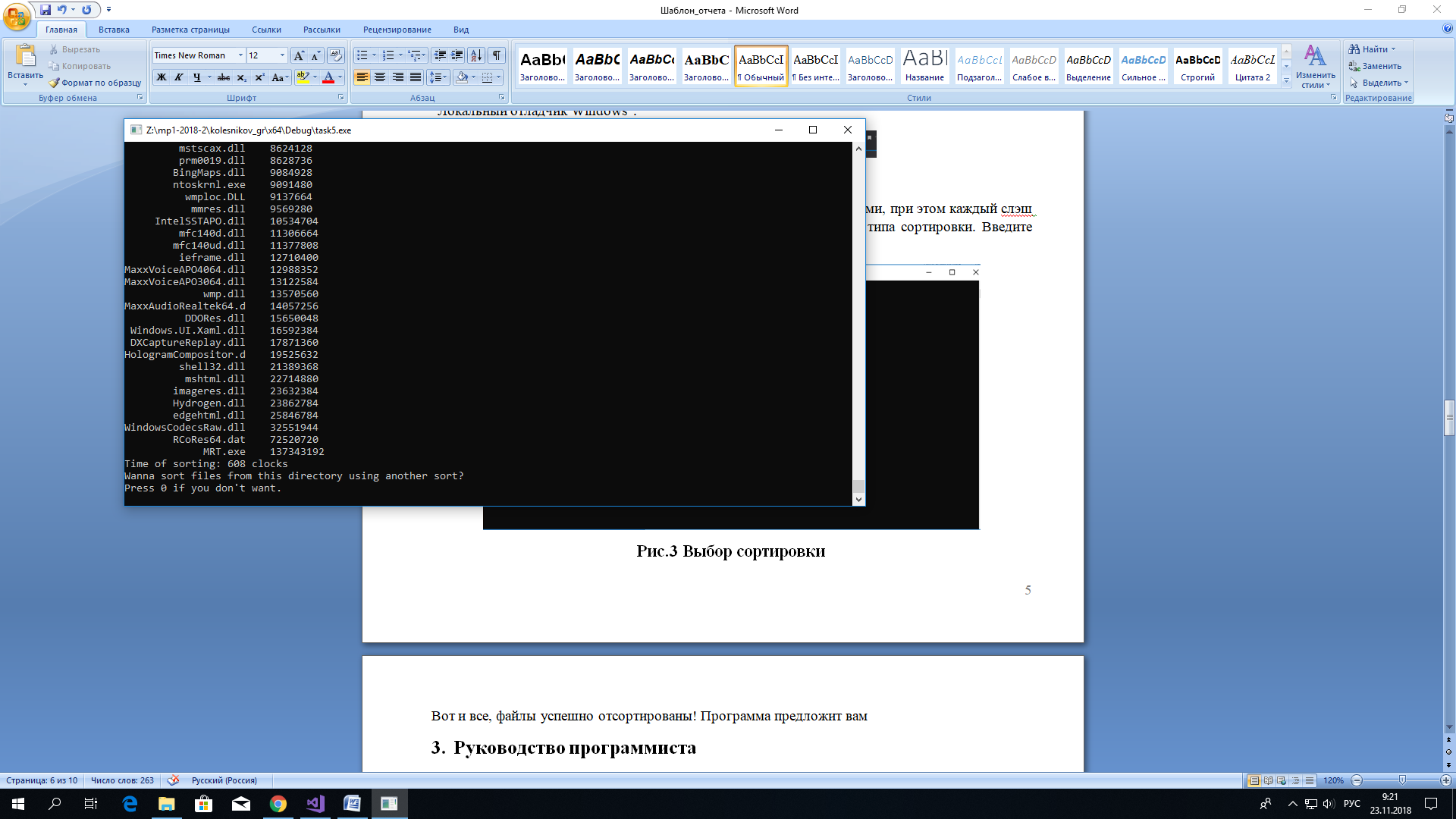
Вот и все, файлы успешно отсортированы! Результат самой сортировки программы выводит примерно так:



**Рис.4 Отсортированные файлы**

После сортировки программа выводит время сортировки в “клоках” – количество временных тактов, прошедших за время сортировки. Клоки не нужно округлять и выводить как число с запятой, поэтому они будут более наглядны для измерения скорости сортировки.

Программа предложит отсортировать каталог еще раз, сменив тип сортировки. Введите 0, если вы хотите закончить сортировки и выйти из программы. Введите что-то кроме 0, если хотите сортировать дальше. Для повторных сортировок работает такой же алгоритм действия, который применялся к первой.



**Рис.5 Повторные сортировки**

Когда закончите сортировки, введите любой символ для завершения программы.

1. **Руководство программиста**

## Описание структуры программы

Программа состоит из одного файла. В этом файле есть три функции

* Функция switch1

Меняет строки местами. Используется при сортировках.

* Функция quick\_Sort

Реализует один из типов сортировки, быструю сортировку. Функция является рекурсивной.

Код см. в приложении.

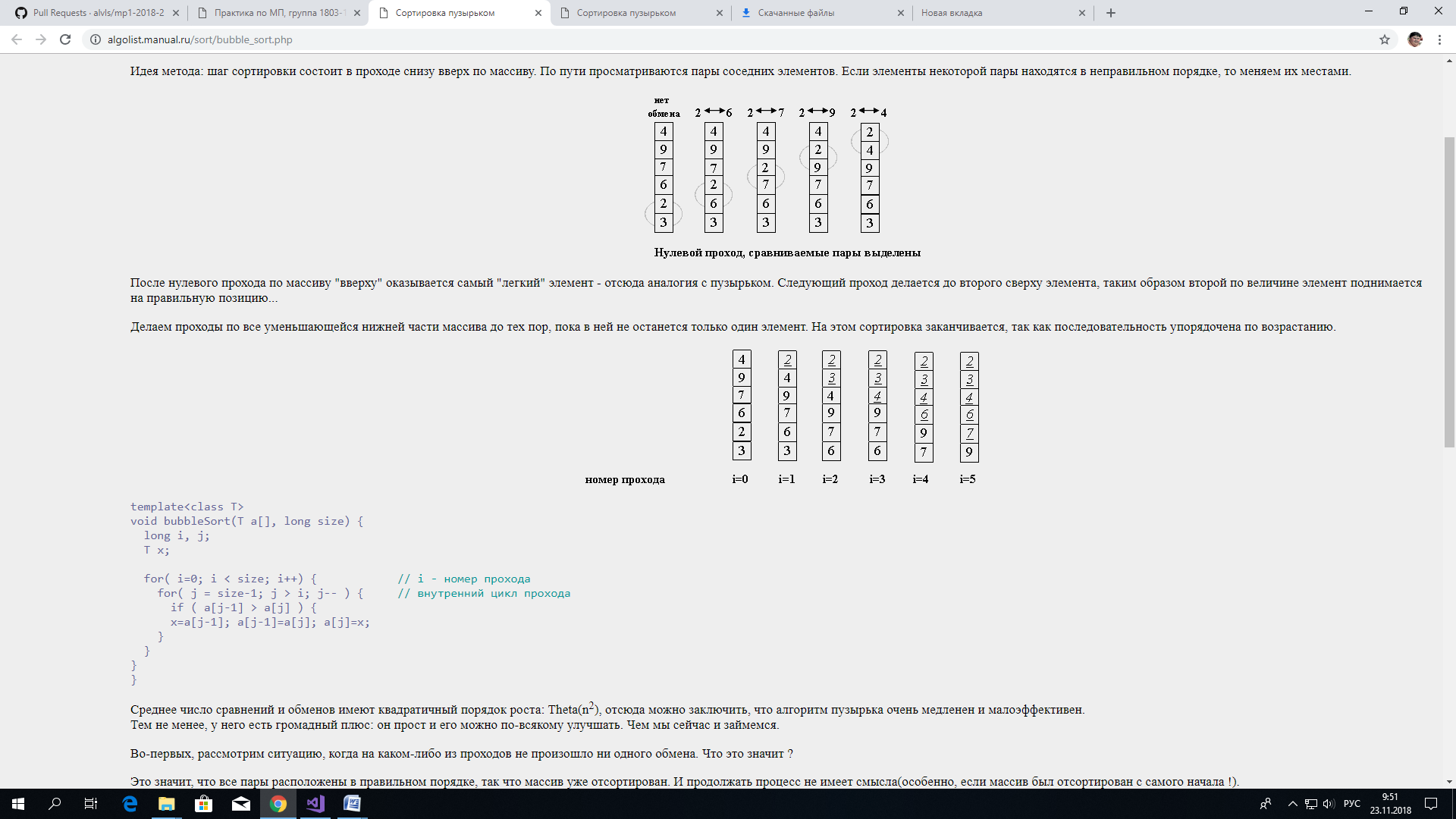
* Функция main

Основная функция, здесь выполняются операции, ввод-вывод файлов, вызов функций. Код см. в приложении.

## Описание алгоритмов

### Сортировка пузырьком

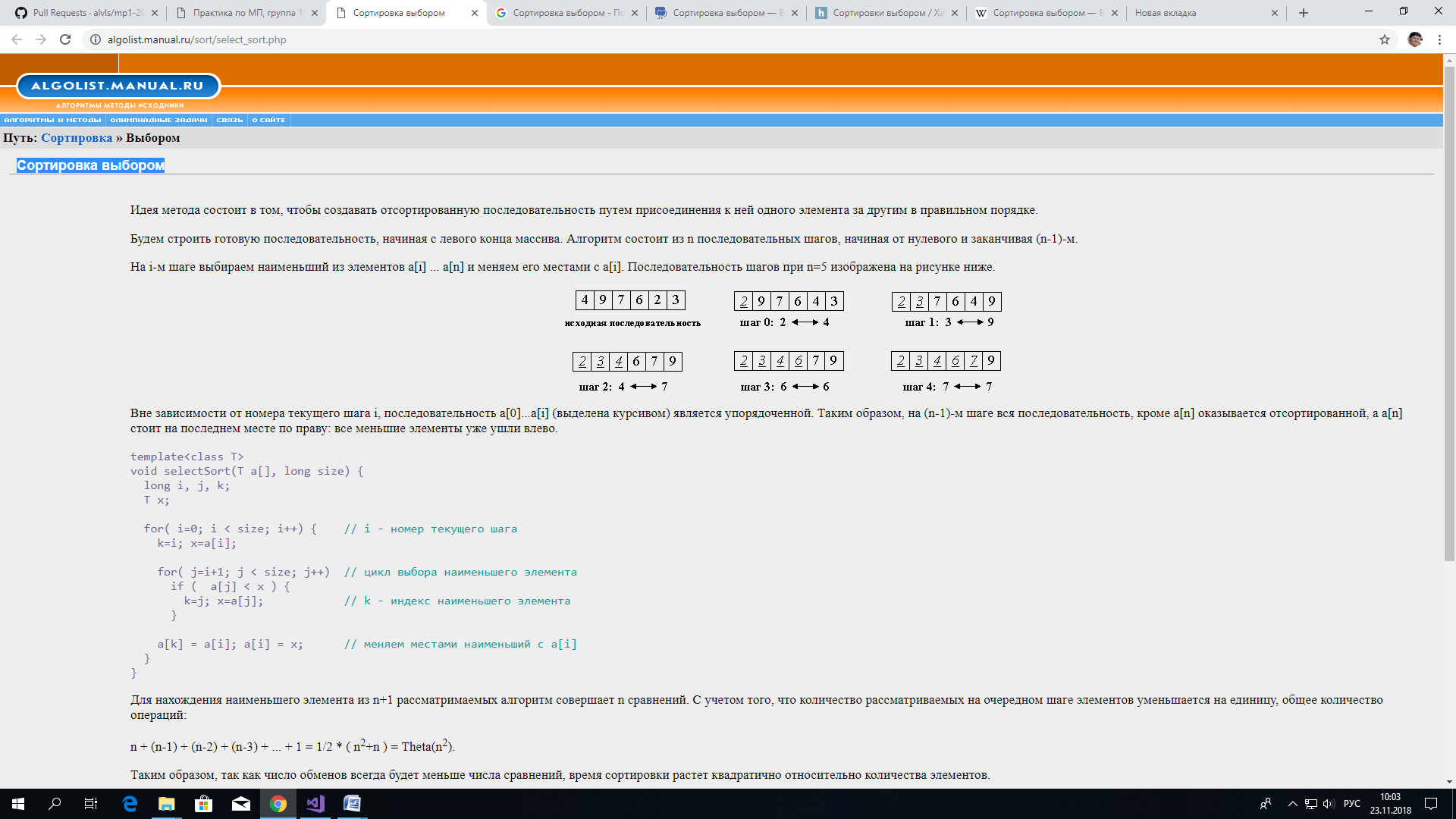
Принцип сортировки: просматриваем массив снизу вверх, рассматривая соседние пары элементов. Если порядок элементов в паре неправильный, меняем элементы местами. Проходы совершаются до полной сортировки массива.



**Рис.6 Пример работы пузырька**

### Сортировка выбором

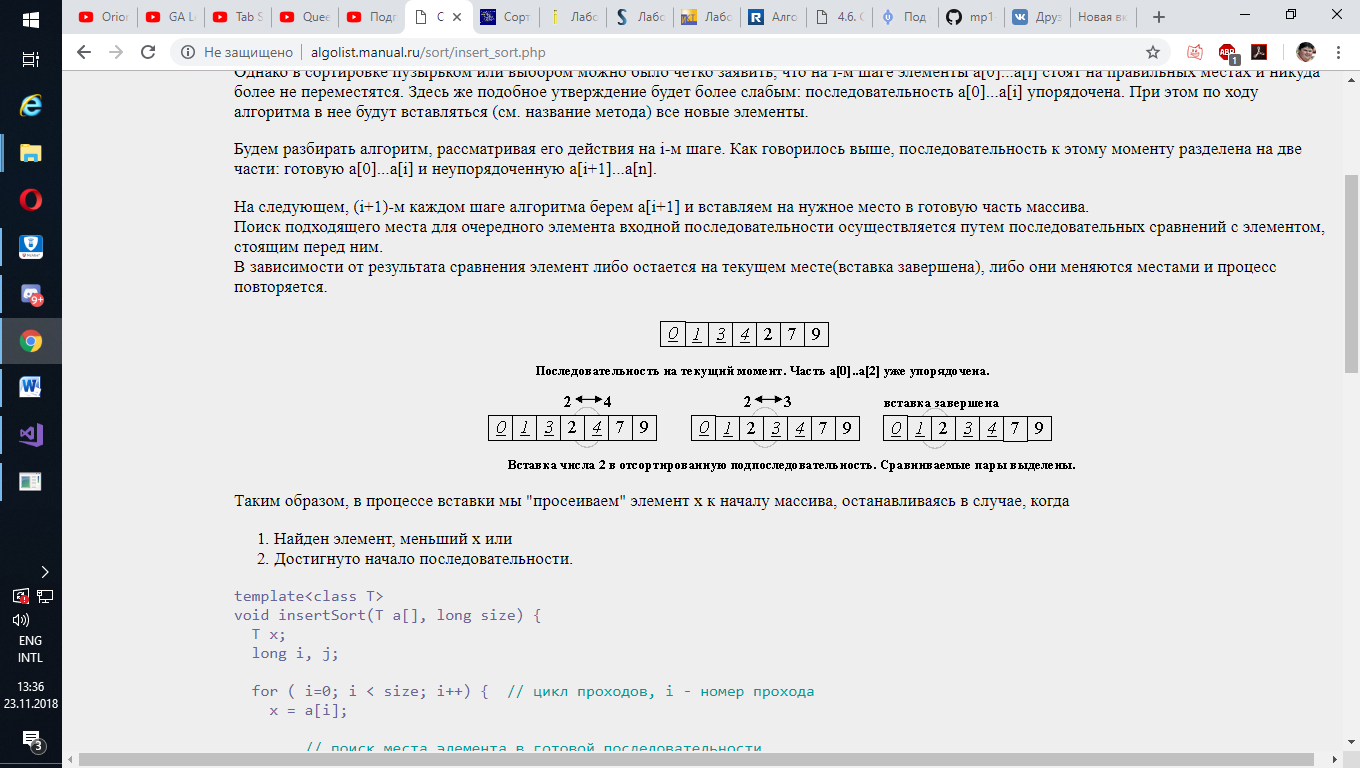
Принцип сортировки: на каждом i-ом шаге алгоритма находим i-ый минимальный элемент и меняем его местами с i-ым элементом в массиве.



**Рис.6 Пример работы сортировки выбором**

### Сортировка вставками

Принцип сортировки: массив условно делится на отсортированную и неотсортированную часть, перебираем неотсортированные элементы, каждый элемент вставляем в отсортированную часть на то место, где он должен находиться.



**Рис.7 Пример сортировки вставками**

### Сортировка Хоара

Принцип сортировки: выбираем из массива какой-то элемент. Оставшиеся элементы нужно сравнивать с опорным и переставить их так, чтобы сначала стояли элементы, меньшие опорного, затем элементы, равные ему, а за ними элементы, большие опорного. После этот алгоритм нужно рекурсивно применить к “меньшей” части и к “большей”, если длина этих частей больше единицы.

# Результаты экспериментов

Прим 1. Из-за высокой скорости работы компьютера сортировки нет смысла проводить на каталоги с малым количеством файлов, ведь при любой сортировки программа выполнится почти мгновенно, и результаты экспериментов не будут отражать реальную скорость сортировок.

Прим 2. Даже на большом количестве файлов время в секундах будет слишком малым, поэтому для измерения времени пользуемся клоками (См. руководство пользователя стр. 6).

Результаты выполнения сортировок. Каталок с 4744 файлами

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Пузырьком | Выбором | Вставками | Хаора |
| Время 1-ого прогона, клок | 626 | 20 | 508 | 2 |
| Время 2-ого прогона, клок | 594 | 20 | 512 | 1 |
| Время 3-его прогона, клок | 607 | 22 | 518 | 1 |
| Время 4-ого прогона, клок | 598 | 19 | 507 | 1 |

# 

# Заключение

Выполнил поставленную задачу. Разработал программу, сортирующую файлы. Провел эксперименты по определению скорости сортировок. Быстрее всех работает сортировка Хаора, самая медленная - пузырьковая

# Литература

1. Кнут Д. Э. Искусство программирования. Том 3. Сортировкаипоиск = The Art of Computer Programming. Volume 3. Sorting and Searching / подред. В. Т. Тертышного (гл. 5) иИ. В. Красикова (гл. 6). – 2-еизд. –Москва: Вильямс, 2007. –Т. 3. – 832 с.
2. СайтAlgolist.Сортировка выбором –<http://algolist.manual.ru/sort/select_sort.php>.
3. СайтAlgolist.Сортировка пузырьком –<http://algolist.manual.ru/sort/bubble_sort.php>.
4. СайтKvodo Сортировка выбором -http://kvodo.ru/sortirovka-vyiborom-2.html

# Приложение

1. Функция Main

int main()

{

struct \_finddata\_t c\_file;

intptr\_t hFile;

int flag = 1;

clock\_t t1, t2, t3, tt;

char path[200];

int count = 0;

t3 = clock();

int k = 0, i = 0, j = 0, l = 0;

unsigned long sizez[100000] = { 0 };

unsigned long x;

int addnamez[100000];

for (i = 0; i < 100000; i++)

addnamez[i] = i;

printf("Enter catalog adress with doubled slashes\n");

gets(path);

int a = strlen(path);

path[a + 3] = '\0';

path[a + 2] = '\*';

path[a] = '\\';

path[a + 1] = '\\';

i = 0;

while (flag)

{

printf("Choose the sorting algorithm\n");

printf(" 1) - bubble sort\n 2) - select sort\n 3) - insert sort\n 4) - quick(Hoala) sort\n");

while ((k < 1) || (k > 5))

scanf("%d", &k);

if ((hFile = \_findfirst(path, &c\_file)) == -1L)

printf("No files fould!\n");

else

{

printf("Listing of files\n\n");

printf("FILE DATE %24c SIZE\n", ' ');

printf("---- ---- %24c ----\n", ' ');

do {

char buffer[30];

if (i > -1)

{

sizez[i] = c\_file.size;

count++;

strncpy(namez[addnamez[i]], c\_file.name, 250);

}

ctime\_s(buffer, \_countof(buffer), &c\_file.time\_write);

if (count < 20);

printf("%-12.12s %.24s %10u\n", c\_file.name, buffer, c\_file.size);

i++;

} while (\_findnext(hFile, &c\_file) == 0);

\_findclose(hFile);

printf("\ncount of files: %d\n", count);

}

t1 = clock();

switch (k)

{

case 1:

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

{

for (j = count - 1; j > i; j--)

{

if (sizez[j - 1] > sizez[j])

{

x = sizez[j - 1];

sizez[j - 1] = sizez[j];

Switch1(namez[j - 1], namez[j]);

sizez[j] = x;

}

}

}

t2 = clock();

break;

case 2:

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

{

k = i;

x = sizez[i];

for (j = i + 1; j < count; j++)

if (sizez[j] < x)

{

k = j;

x = sizez[j];

}

sizez[k] = sizez[i];

Switch1(namez[k], namez[i]);

sizez[i] = x;

}

t2 = clock();

break;

case 3:

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

{

x = sizez[i];

for (j = i - 1; j >= 0 && sizez[j] > x; j--)

{

sizez[j + 1] = sizez[j];

Switch1(namez[j], namez[j + 1]);

}

sizez[j + 1] = x;

}

t2 = clock();

break;

case 4:

quick\_Sort(sizez, addnamez, 0, count - 1);

t2 = clock();

break;

}

tt = t2 - t1;

printf("%16cFILE SIZE\n", ' ');

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

{

printf("%20.20s\t", namez[addnamez[i]]);

printf("%u\n", sizez[i]);

}

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

addnamez[i] = i;

printf("Time of sorting: %u clocks\n", tt);

printf("Wanna sort files from this directory using another sort?\nPress 0 if you don't want.\n");

scanf("%d", &flag);

k = 0;

i = 0;

count = 0;

j = 0;

}

\_getch();

}

1. Функция quick\_Sort

void quick\_Sort(unsigned long a[], int b[], int ll, int count)

{

long i = ll, j = count;

long p = a[(count + ll) / 2];

unsigned long temp;

do {

while (a[i] < p) i++;

while (a[j] > p) j--;

if (i <= j)

{

if (i < j)

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

temp = b[i];

b[i] = b[j];

b[j] = temp;

}

i++;

j--;

}

} while (i <= j);

if (j > ll) quick\_Sort(a, b, ll, j);

if (count > i) quick\_Sort(a, b, i, count);

}