**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: Работа с массивами

|  |  |  |
| --- | --- | --- |
| Студент гр. 9893 |  | Росихин А.Ю. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2019

**Цель работы.**

Научиться обрабатывать одномерные массивы данных. Изучение разных видов сортировок. Изучение генератора случайных чисел и работа со временем.

**Основные теоретические положения.**

Объявление статического массива.

Генератор случайных чисел в языке С++. Функция rand() стандартной библиотеки генерирует случайное число. Для того чтобы масштабировать интервал генерации чисел нужно воспользоваться, операцией нахождения остатка от деления «%». В нашем случае по условиям задачи это rand() % 100.

Сортировка «расческой» (Combsort). Основная идея «расчёски» в том, чтобы первоначально брать достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива сужать это расстояние вплоть до минимального. Таким образом, мы как бы причёсываем массив, постепенно разглаживая на всё более аккуратные пряди. Первоначальный разрыв между сравниваемыми элементами лучше брать с учётом специальной величины, называемой фактором уменьшения, оптимальное значение которой равно примерно 1,247. Сначала расстояние между элементами равно размеру массива, разделённого на фактор уменьшения (результат округляется до ближайшего целого). Затем, пройдя массив с этим шагом, необходимо поделить шаг на фактор уменьшения и пройти по списку вновь. Так продолжается до тех пор, пока разность индексов не достигнет единицы. В этом случае массив досортировывается обычным пузырьком [1].

Поиск минимального и максимального элемента массива сводится к выводу 1-го и последнего элемента массива.

Для подсчета скорости работы программы и ее частей используется функция clock\_tclock( void ). Она возвращает количество временных тактов, прошедших с начала запуска программы. С помощью макроса CLOCKS\_PER\_SEC функция получает количество пройденных тактов за 1 секунду. Таким образом, зная, сколько выполняется тактов в секунду, зная время запуска программы можно посчитать время работы всей программы или отдельного её фрагмента, что и делает данная функция [2].

**Постановка задачи.**

Разработать алгоритм и написать программу:

1) которая создает массив размерности *N*, заполненный случайными числами в диапазоне от 0 до 99;

2) сортирует элементы массива Combsort сортировкой;

3) ищет максимальный и минимальный элемент массива;

4) выводит количество элементов, равных среднему значения *max* и *min*;

5) выводит количество элементов, которые меньше числа *a*;

6) выводит количество элементов, которые больше числа *b*;

7) выводит скорость удаление, вставки и поиска значения;

8) выводит скорость обмена элементов массива, номера пользователем;

**Выполнение работы.**

Перед началом выполнения работы разобьем основную задачу на подзадачи для уменьшения сложности ее восприятия:

1. Получить на входе число равное размеру массива.
2. В цикле заполнить массив случайными числами.
3. Отсортировать массив методом «расческа».
4. Досортировать массив методом «пузырька».
5. Вывести 1-й и последний элемент массива.
6. Найти среднее значение *min* и max и с помощью счетчика в цикле найти количество элементов равное среднему значению.
7. Получить на входе числа *a* и *b*.
8. Найти позицию последнего элемента меньшего *a* и большего *b* и вывести на экран.
9. Присвоисть переменной типа clock\_tstart значение clock() в начале программы, в начале работы с массивом и в конце. Потом вычесть из конечного значения начальное.

Исходный код программы представлен в приложении А.

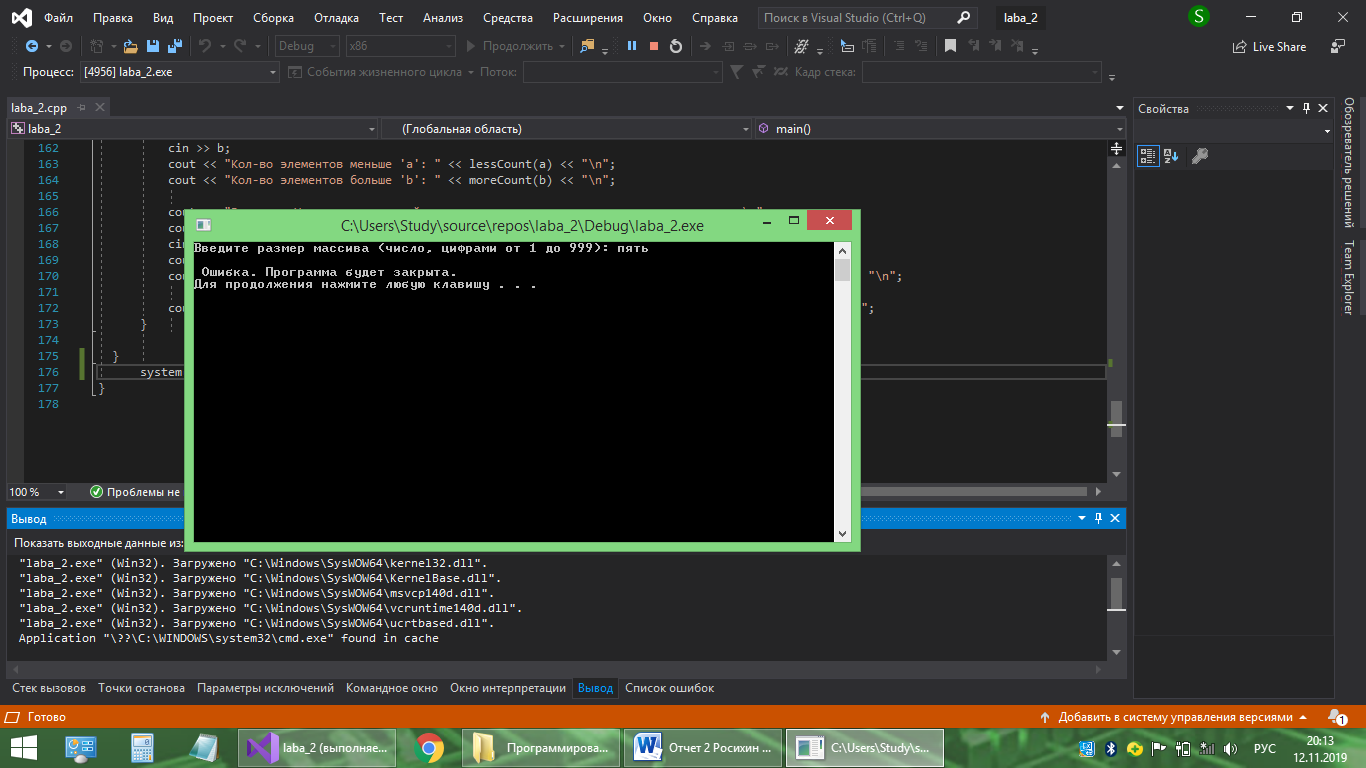
Скриншоты работы программы с разными значениями «см. рис. с 1 по 10».

Рисунок 1 — Скриншот работы программы с текстовым значением

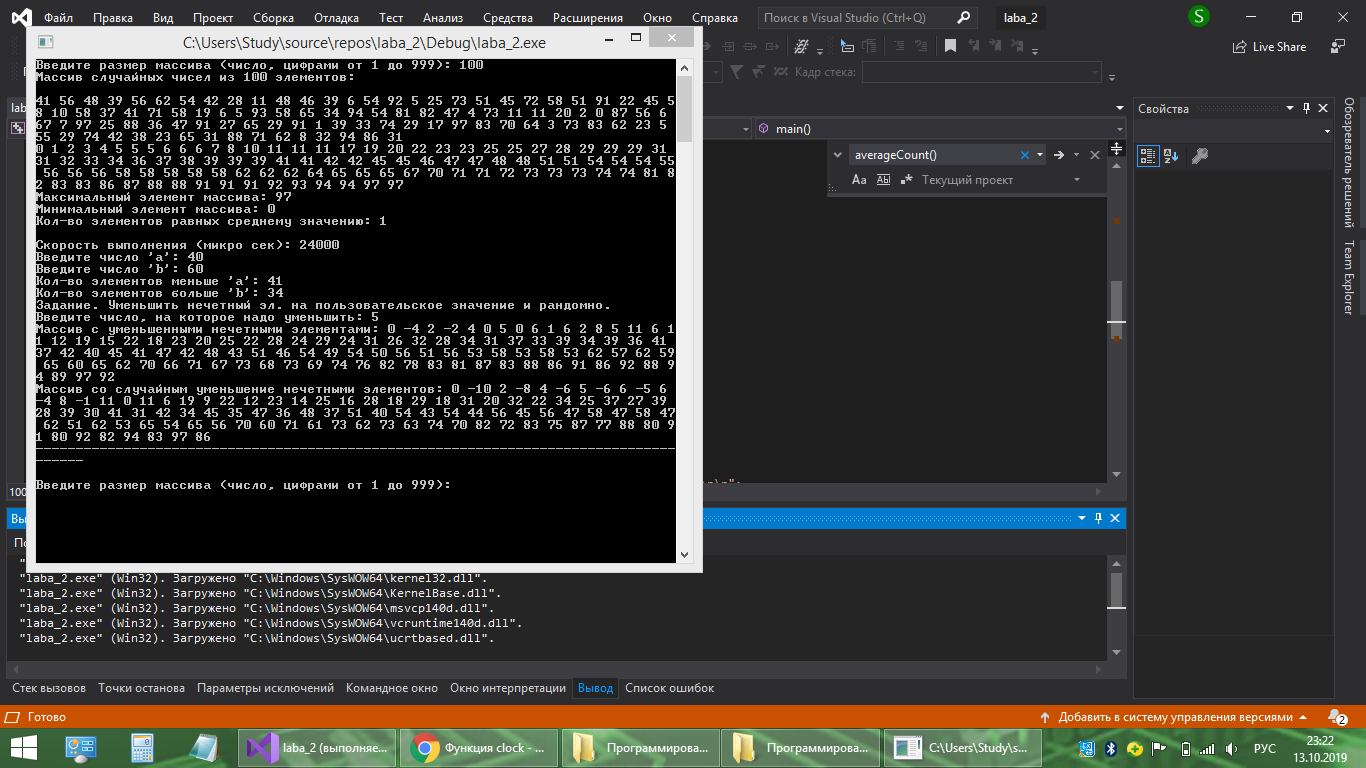
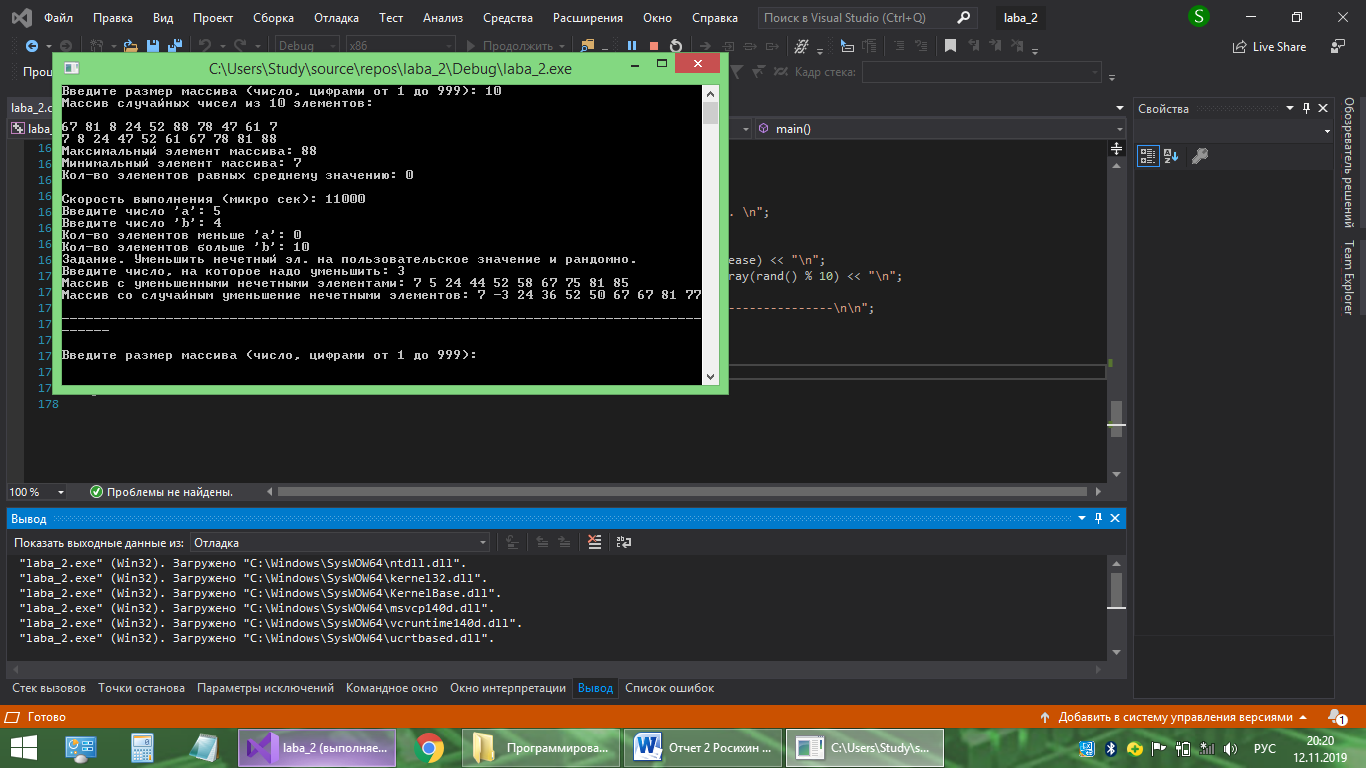
****Рисунок 2 — Скриншот работы программы со значением 100

Рисунок 3 — Скриншот работы программы со значением 10

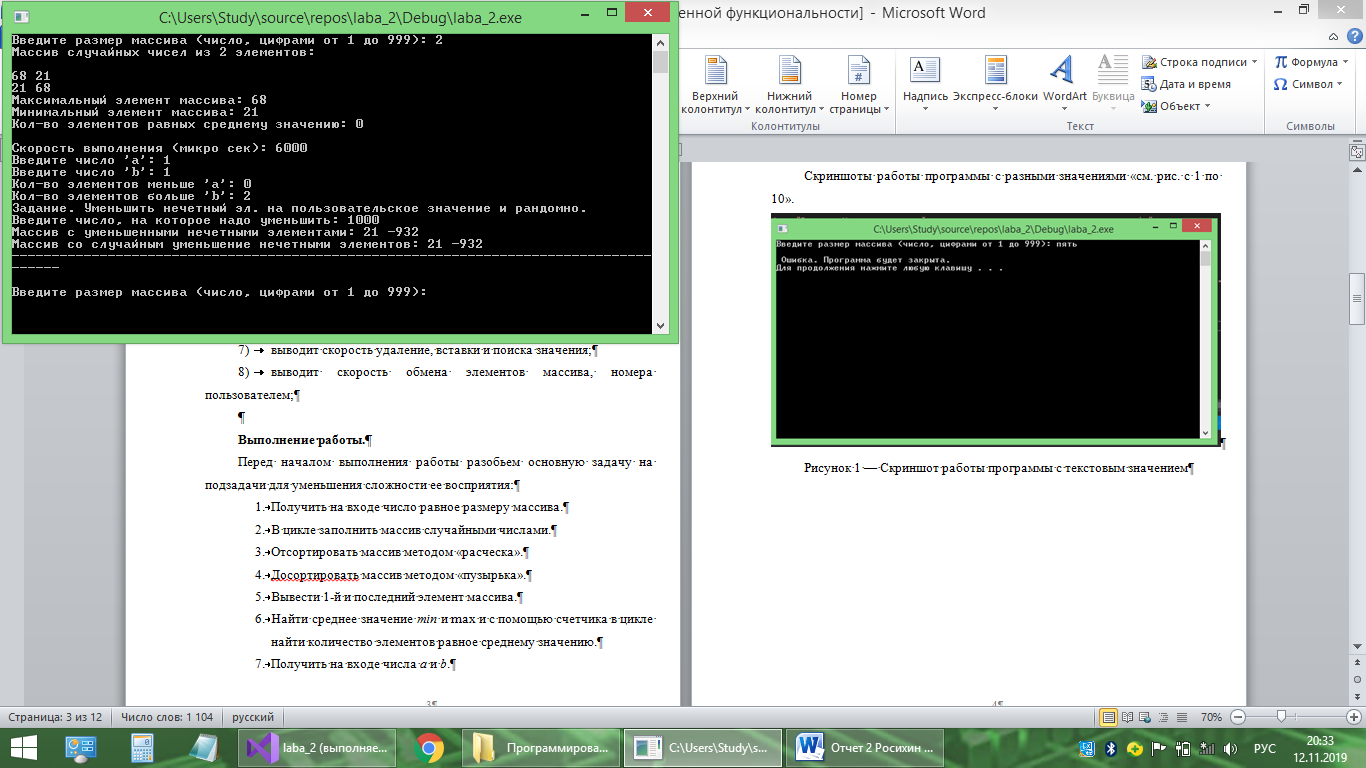
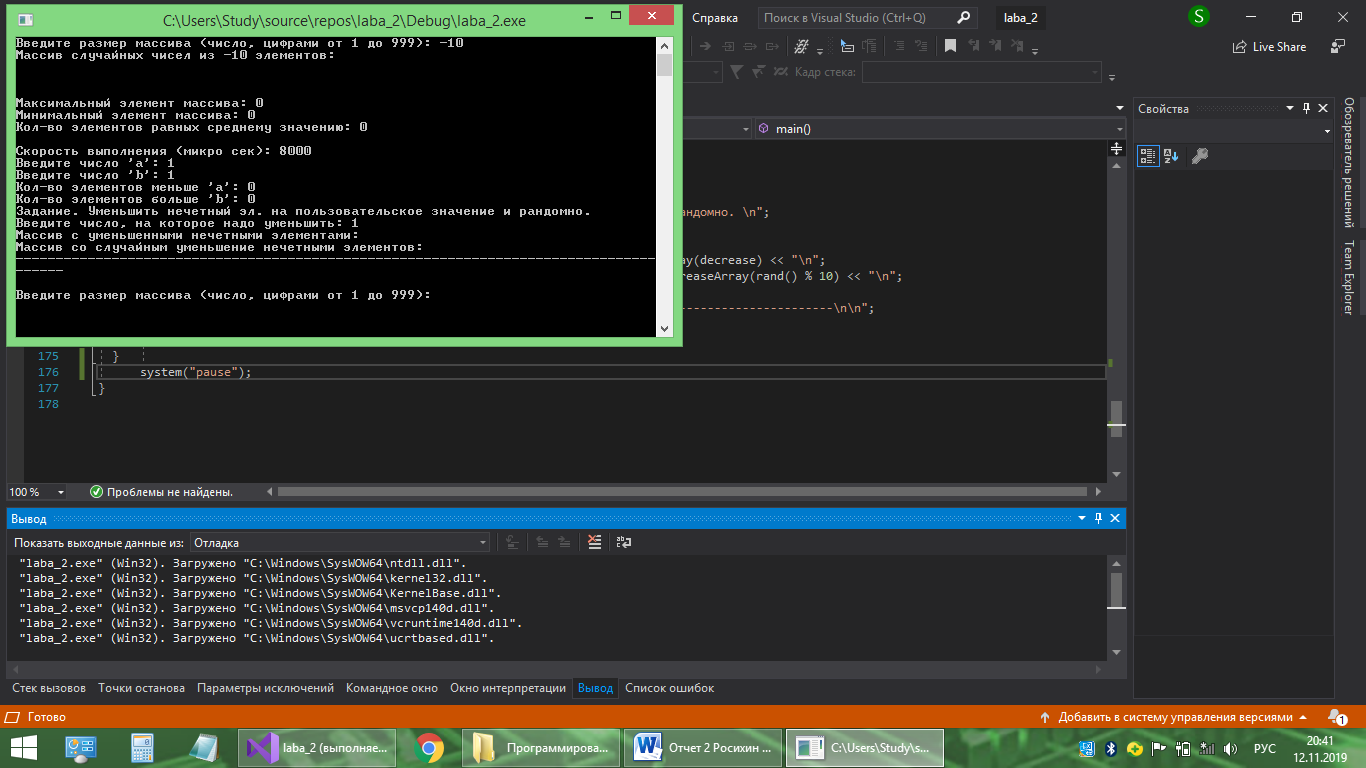
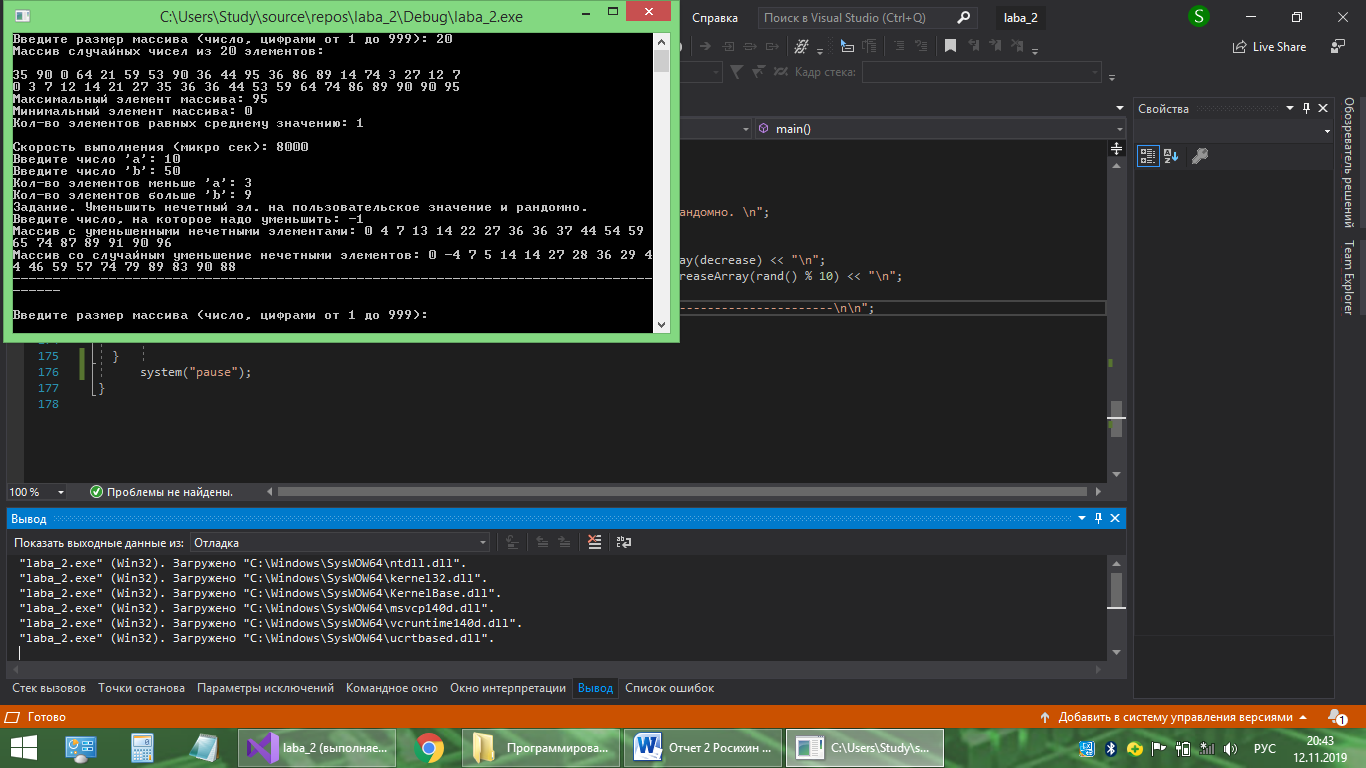
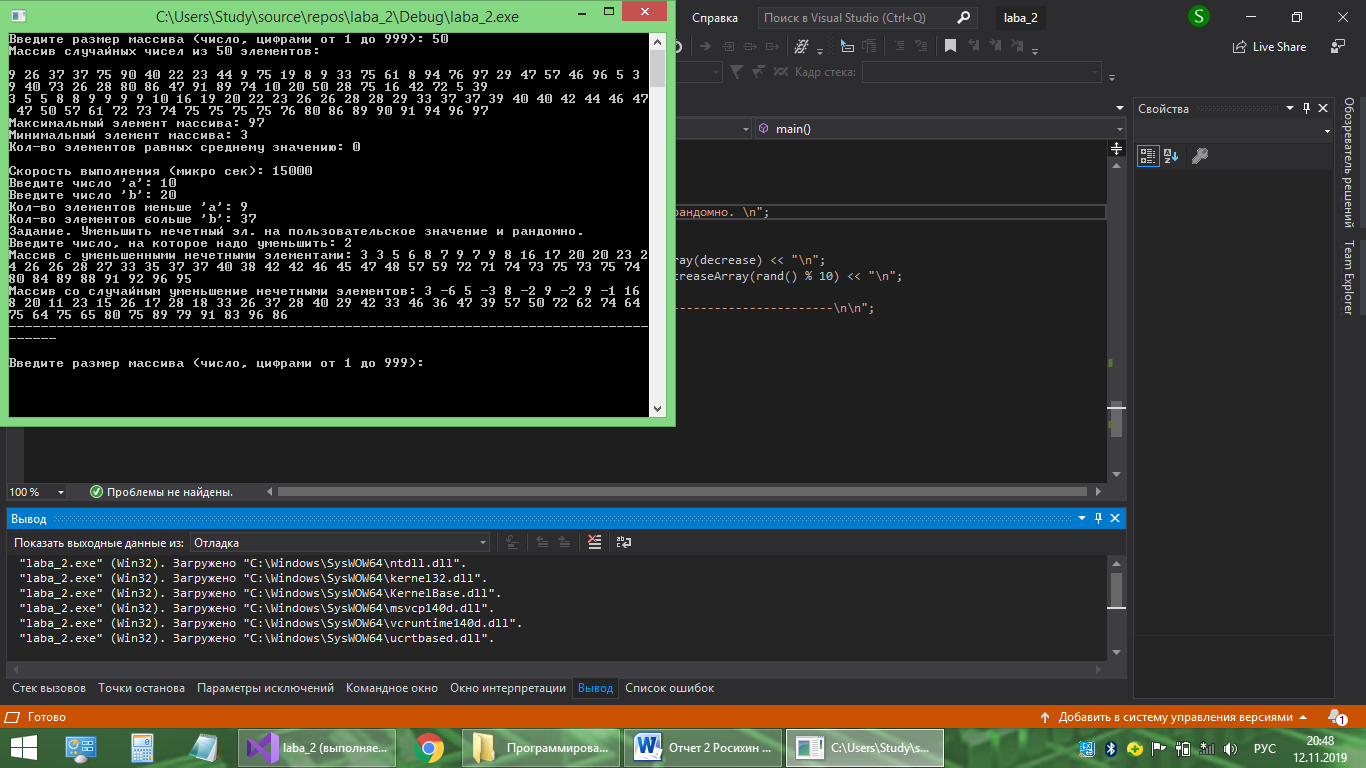
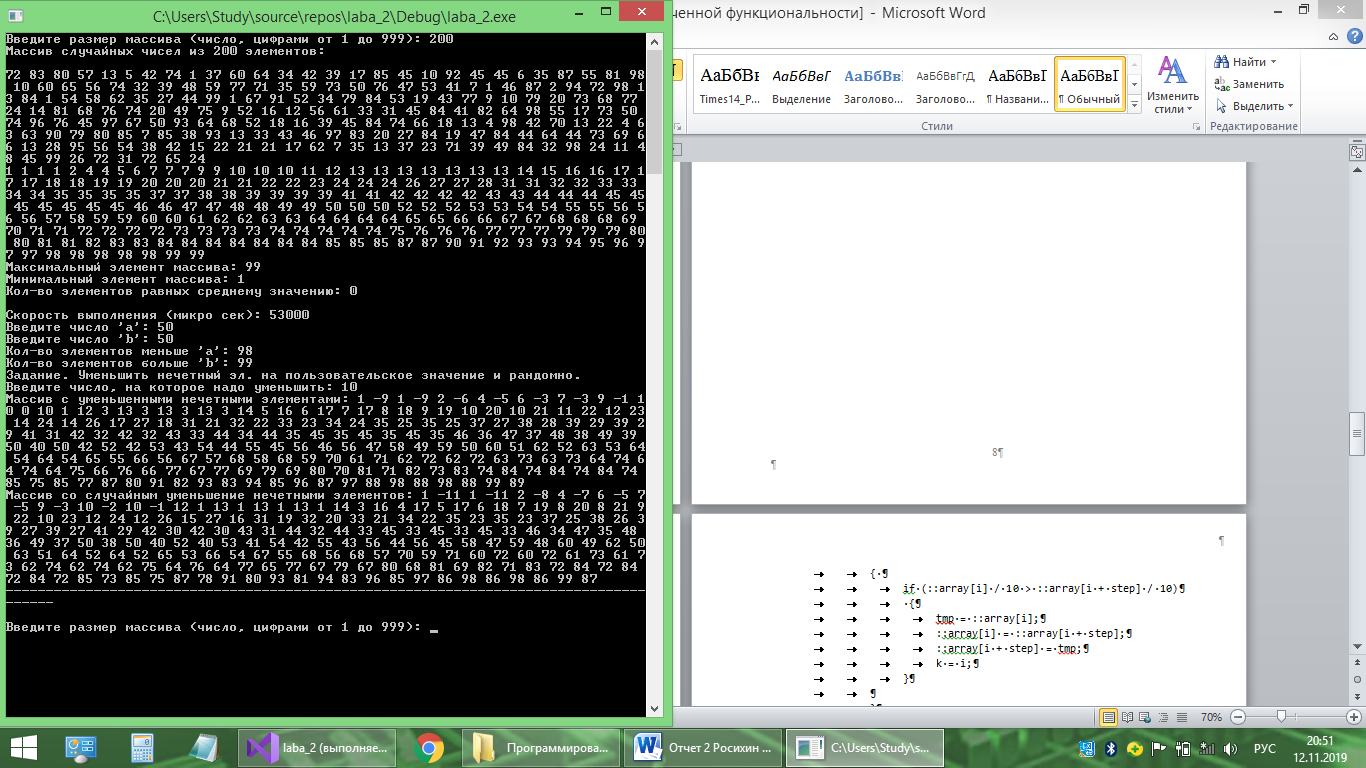
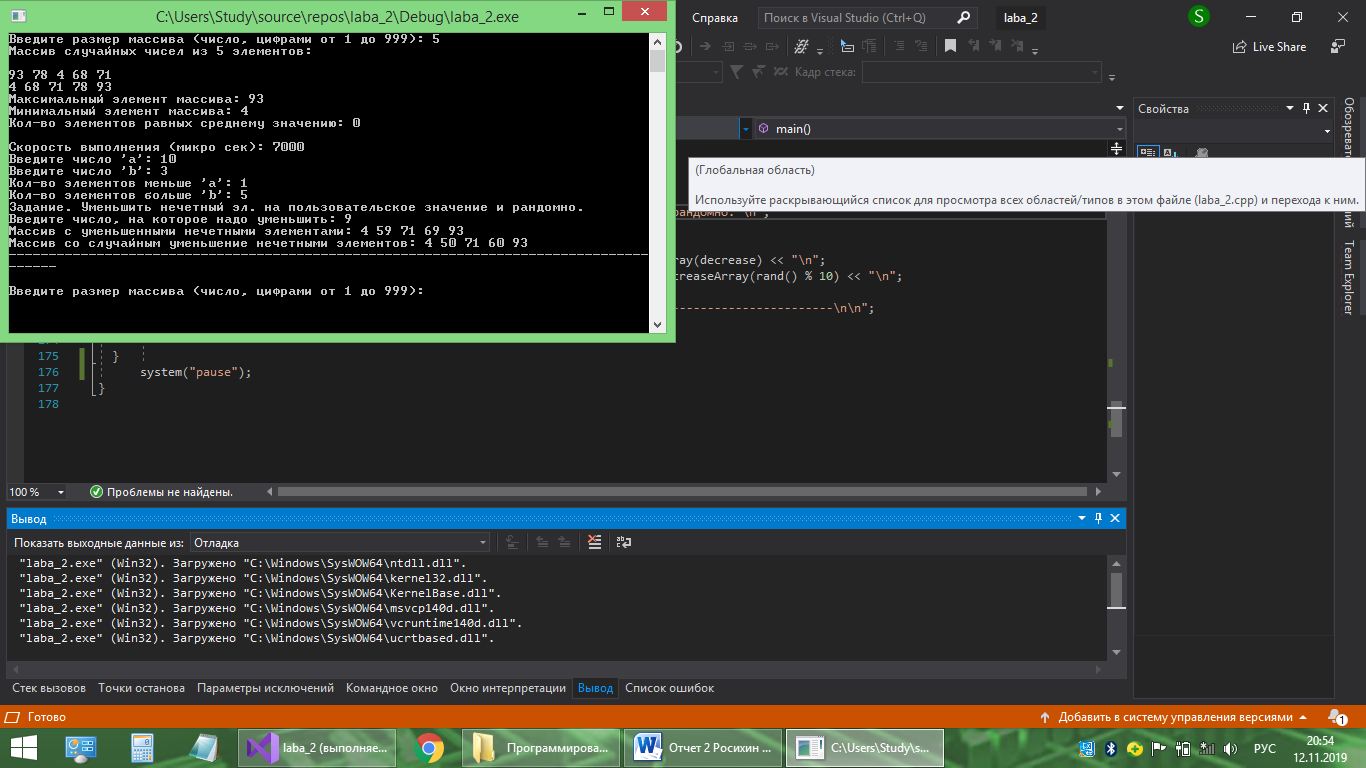
****Рисунок 4 — Скриншот работы программы со значением 2

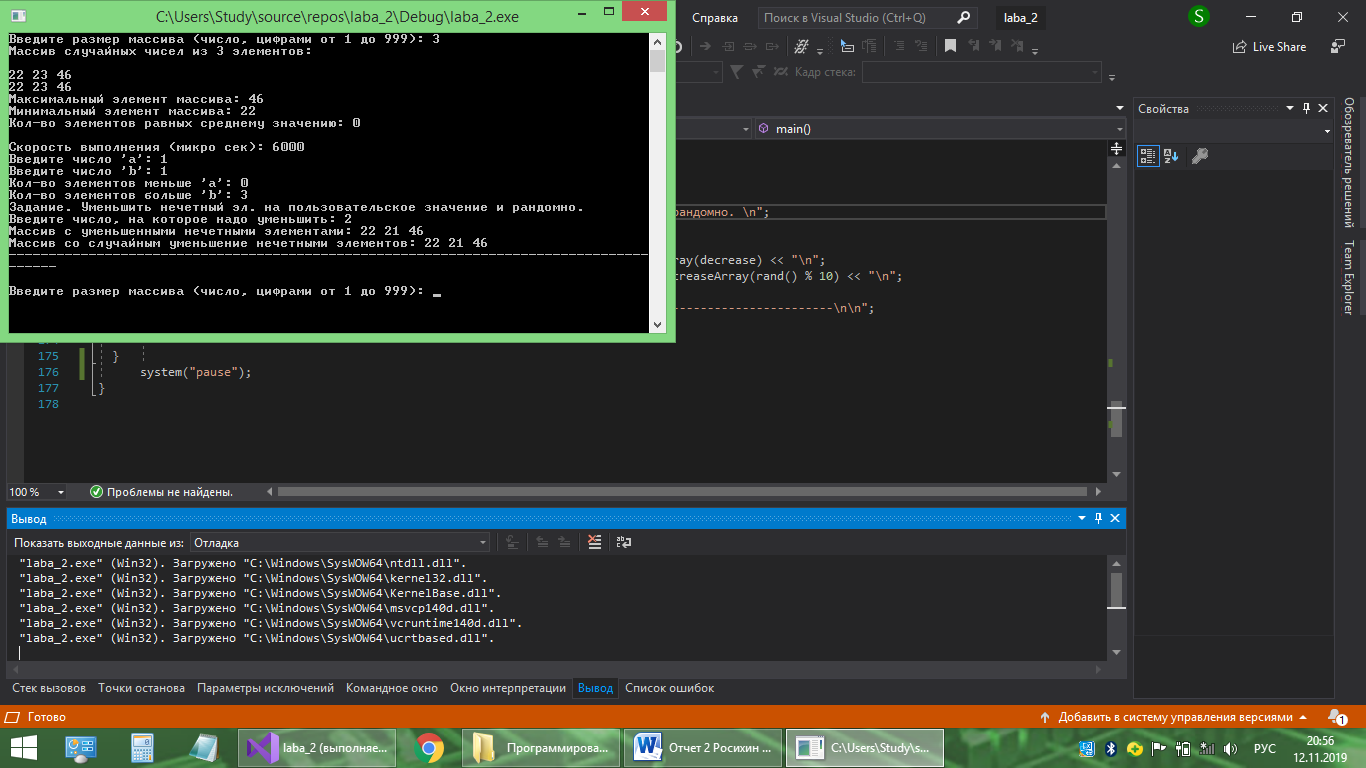
Рисунок 5 — Скриншот работы программы со значением -10

****Рисунок 6 — Скриншот работы программы со значением 20

****Рисунок 7 — Скриншот работы программы со значением 50

Рисунок 8 — Скриншот работы программы со значением 200

Рисунок 9 — Скриншот работы программы со значением 5

Рисунок 10 — Скриншот работы программы со значением 3

**Выводы.**

Время работы программы растет пропорционально размеру массива.

Сортировка «расческой» гораздо эффективнее простой сортировки «пузырьком».

Для нахождения минимального и максимально элемента массива, достаточно получить значение 1-го и последнего элемента отсортированного массива.

В сортировке «расческой» производятся неоднократные прогоны по массиву, при которых сравниваются пары элементов. Если они не отсортированы друг относительно друга, то производится обмен этих элементов. В результате «крупные элементы» перемещаются в конец массива, а «малые» в начало. Наилучший вариант этой сортировки, если элементы исходного массива уже отсортированы и перестановка элементов не требуется, соответственно наихудший вариант если элементы исходного массива отсортированы в обратном порядке.

Плюс этого вида сортировки состоит в том, чтобы устранить «малые» элементы в конце массива, которые крайне замедляют скорость простой пузырьковой сортировки. Также, несомненным плюсом этого вида сортировки является ее простота в понимании и реализации, грубо говоря, сортировка «расческой» это модифицированный вид сортировки «пузырьком».

Минус этого вида сортировки состоит в том, что если в массиве много одинаковых элементов, эта сортировка все равно будет проверять все эти одинаковые элементы.

Приложение А

Название приложения

#include<iostream>

#include <string>

#include <ctime>

using namespace std;

int array[1000], arraySize=0, decrease;

stringshowArray(int size)

{

string result;

srand(time(0));

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

{

::array[i] = rand() % 100;

result += to\_string(::array[i]) + " ";

}

return result;

}

voidsortc(int n)

{

inttmp, k;

intstep = n; // готовимначальныйшаг

while (n > 1) {

step /= 1.247f; // шаг на этом проходе. На первом проходе получается примерно 80% от размера массива,

// и легкие элементы в хвосте переносятся в первые 20%

if (step< 1) step = 1; // 0 быть не может, присвоим 1

k = 0; // нетперестановок

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

{

if (::array[i] / 10 > ::array[i + step] / 10)

{

tmp = ::array[i];

::array[i] = ::array[i + step];

::array[i + step] = tmp;

k = i;

}

}

if (step == 1) // шаг 1, как в обычном пузырьке. Включаемконтроль

n = k + 1;

}

}

stringshowSort(int size)

{

string result;

sortc(size);

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

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

if (::array[j] > ::array[j + 1])

{

int temp = ::array[j];

::array[j] = ::array[j + 1];

::array[j + 1] = temp;

}

for (int i = 0; i < size; i++) result += to\_string(::array[i]) + " ";

return result;

}

intaverageCount()

{

int count=0;

int average = (::array[0] + ::array[::arraySize]) / 2 ;

for (int i = 0; i < ::arraySize; i++) if (::array[i] == average) count++;

return count;

}

intlessCount(int x)

{

int count = 0;

for (int i = 0; i < ::arraySize; i++)

if (::array[i] < x) count = i+1;

else break;

return count;

}

intmoreCount(int x)

{

int count = 0;

for (int i = ::arraySize-1; i>=0; i--)

if (::array[i] > x) count = ::arraySize-i;

else break;

return count;

}

stringshowDecreaseArray(int d)

{

string result="";

for (int i = 0; i < ::arraySize; i++)

{

if (i % 2 != 0) ::array[i] = ::array[i] - d;

result += to\_string(::array[i]) + " ";

}

return result;

}

int main()

{

setlocale(0, "");

int a, b;

while (true)

{

cout<< "Введите размер массива (число, цифрами от 1 до 999): ";

cin>>arraySize;

if (cin.fail())

{

cout<< "\n Ошибка. Программа будет закрыта.\n";

cin.clear();

break;

}

else

{

clock\_tstart = clock();

cout<< "Массив случайных чисел из " <<arraySize<< " элементов:\n\n";

cout<<showArray(arraySize) + "\n";

cout<<showSort(arraySize) + "\n";

cout<< "Максимальный элемент массива: " << ::array[arraySize - 1] << "\n";

cout<< "Минимальный элемент массива: " << ::array[0] << "\n";

cout<< "Кол-во элементов равных среднему значению: " <<averageCount() << "\n\n";

double speed = ((clock() - start) \* 1000000) / CLOCKS\_PER\_SEC;

cout<< "Скорость выполнения (микро сек): " <<speed<< "\n";

cout<< "Введите число 'a': ";

cin>> a;

cout<< "Введите число 'b': ";

cin>> b;

cout<< "Кол-во элементов меньше 'a': " <<lessCount(a) << "\n";

cout<< "Кол-во элементов больше 'b': " <<moreCount(b) << "\n";

cout<< "Задание. Уменьшить нечетный эл. на пользовательское значение и рандомно. \n";

cout<< "Введите число, на которое надо уменьшить: ";

cin>>decrease;

cout<< "Массив с уменьшенными нечетными элементами: " <<showDecreaseArray(decrease) << "\n";

cout<< "Массив со случайным уменьшение нечетными элементов: " <<showDecreaseArray(rand() % 10) << "\n";

cout<< "--------------------------------------------------------------------------------------\n\n";

}

}

}

**Список использованных источников.**

1. <https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D1%80%D0%B0%D1%81%D1%87%D1%91%D1%81%D0%BA%D0%BE%D0%B9>
2. <http://cppstudio.com/post/561/>