Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им.И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление:710400 «Программная инженерия»

ОТЧЕТ

По дисциплине: «Алгоритмы и структуры данных»

Лабораторная работа №1

Тема: «Статические массивы»

Выполнил: студент группы

ПИ(б)-2-19 Улан уулу Нурдин

Проверила: Валеева А. А.

Бишкек – 2020

**Практическое задание**

**I.Ответить на контрольные вопросы**

* 1. **Типы данных в языках программирования**
* **Что такое данные?**

**Данные** – сведения, полученные путем измерения, наблюдения, логических или арифметических операций и представленные в форме, пригодной для постоянного хранения, передачи и обработки.

* **Физическое и логическое представление данных**

**Логическое представление**имеет дело с тем, как пользователи организуют, просматривают, понимают данные и их отношения.

**Физическое представление** имеет дело с тем, как и где данные физически размещаются и хранятся на дисках, магнитных лентах и других носителях.

* **Какие типы данных представлены в СИ++? Описание и операции с этими данными.**

**int, long**- целочисленные; операции сложение, вычитание, деление, умножение, деление с остатком, сравнения, инкремент, декремент.

**double, float** - вещественные; операции сложение, вычитание, деление, умножение, сравнения.

**string, char** – символьные; операции сравнения.

**bool** – логический.

* 1. **Структурированные типы данных –массивы. Описание массива.**

**Массив** - это пронумерованная последовательность величин одинакового типа данных, обозначаемая одним именем. Элементы массива располагаются в последовательных ячейках памяти, обозначаются именем массива и индексом. Каждое из значений, составляющих массив, называется его элементом.

* 1. **Представление массивов в памяти компьютера. Основные операции с массивами.**

Элементы массива размещаются в памяти в последовательных ячейках. Массив занимает количество байт, равное произведению количества элементов массива на размер одного элемента:

**Размер массива = количество элементов в массиве \* размер типа данных элемента массива**

**Основные операции с массивами:** Суммирование, вычитание, деление, поиск, сортировка, циклический сдвиг.

* 1. **Основные инструкции С++.**

Основными инструкциями в C++ являются: =, if(), else if(), switch() case, while(), do{ }while(), for().

**II. Выполнить упражнения**

1. **Постановка задачи**

Найти сумму всех элементов массива (А=10), значения которых больше среднего арифметического всех элементов массива.

1. **Описание входных и выходных данных**

Исходные данные:

А **- массив, подлежащий обработке**;

n - количество реально введенных элементов массива A.

Выходные данные:

average – переменная типа **double,** в которой храниться средняя арифметическая массива A.

A – элементы массива которые больше average.

sum – переменная для хранения суммы всех элементов массива.

Промежуточные данные:

check – переменная для проверки, есть ли элементы больше average или нет.

1. **Описание алгоритма**

Сначала инициализируя все элементы массива, идет сложение всех элементов в переменную sum, и после в переменную average присваивается **sum / n**. После мы определяем, есть ли хотя бы один элемент больше average, и, если есть увеличиваем check и останавливаем цикл. Дальше для красивого вывода проверяем были ли элементы больше average, если да, то через цикл проверяем каждый элемент массива и сразу его выводим.

Начало

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

cin >> A[i]

sum += A[i];

average = sum / n;

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

A[i] > average

True

False

check++

break

True

False

check != 0

Not have

sum = 0

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

True

False

A[i] > average

cout >> A[i]

sum

Конец

1. **Набор тестов**

Исходные данные:

**A=1 5 2 4 7 4 3 9 8 2**

**n=10**

Выходные данные:

**average = 4.5**

**А=5, 7, 9, 8**

Промежуточные данные:

**sum = 29**

**check > 0**

Исходные данные:

**A=0 0 0 0 0 0 0 0 0 0**

**n=10**

Выходные данные:

**average = 0**

Промежуточные данные:

**sum = 0**

**check < 0**

Исходные данные:

**A=-1 -1 -1 -1 -1 -1 -1 -1 -1 -1**

**n=10**

Выходные данные:

**average = -1**

Промежуточные данные:

**sum = 0**

**check < 0**

1. **Текст программы**

#include <iostream>

using namespace std;

int main()

{

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int n = 10;

int A[n], check = 0;

double average, sum = 0;

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

{

cin >> A[i];

sum += A[i];

}

average = sum / n;

cout << "\nСредняя арифметическая = " << average << endl;

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

{

if (A[i] > average)

{

check++;

break;

}

}

sum = 0;

if (check != 0)

{

cout << "\nЭлементы больше средней арифметической" << endl;

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

{

if (A[i] > average)

{

cout << "A[" << i + 1 << "] = " << A[i] << endl;

}

}

}

else

{

cout << "\nЭлементов больше средней арифметической нет" << endl;

}

cout << "Сумма элементов массива, значение которых больше среднего арифметического равна: " << sum;

return 0;

}

1. **Анализ результатов и выводы**

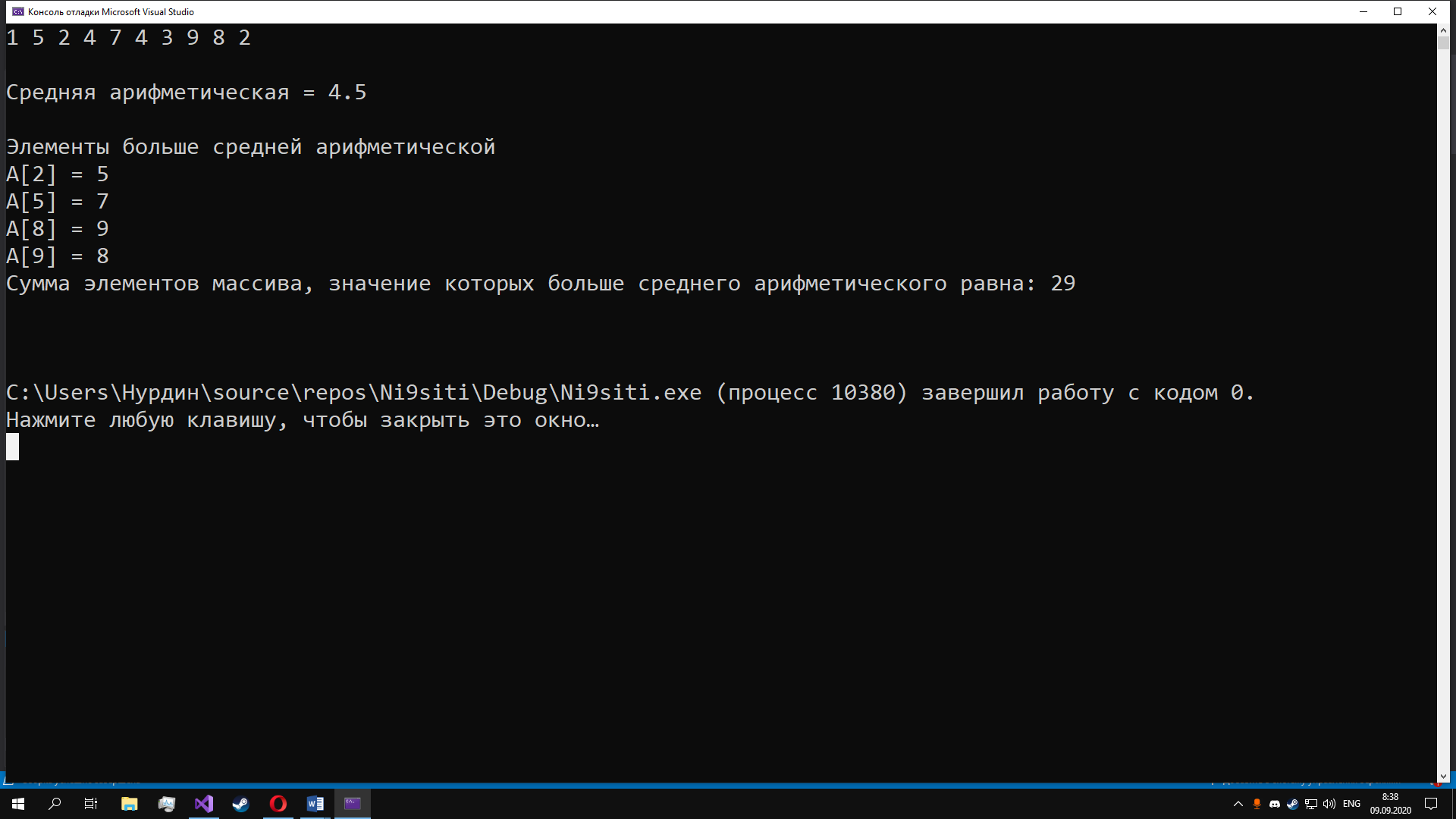


Рис 1.

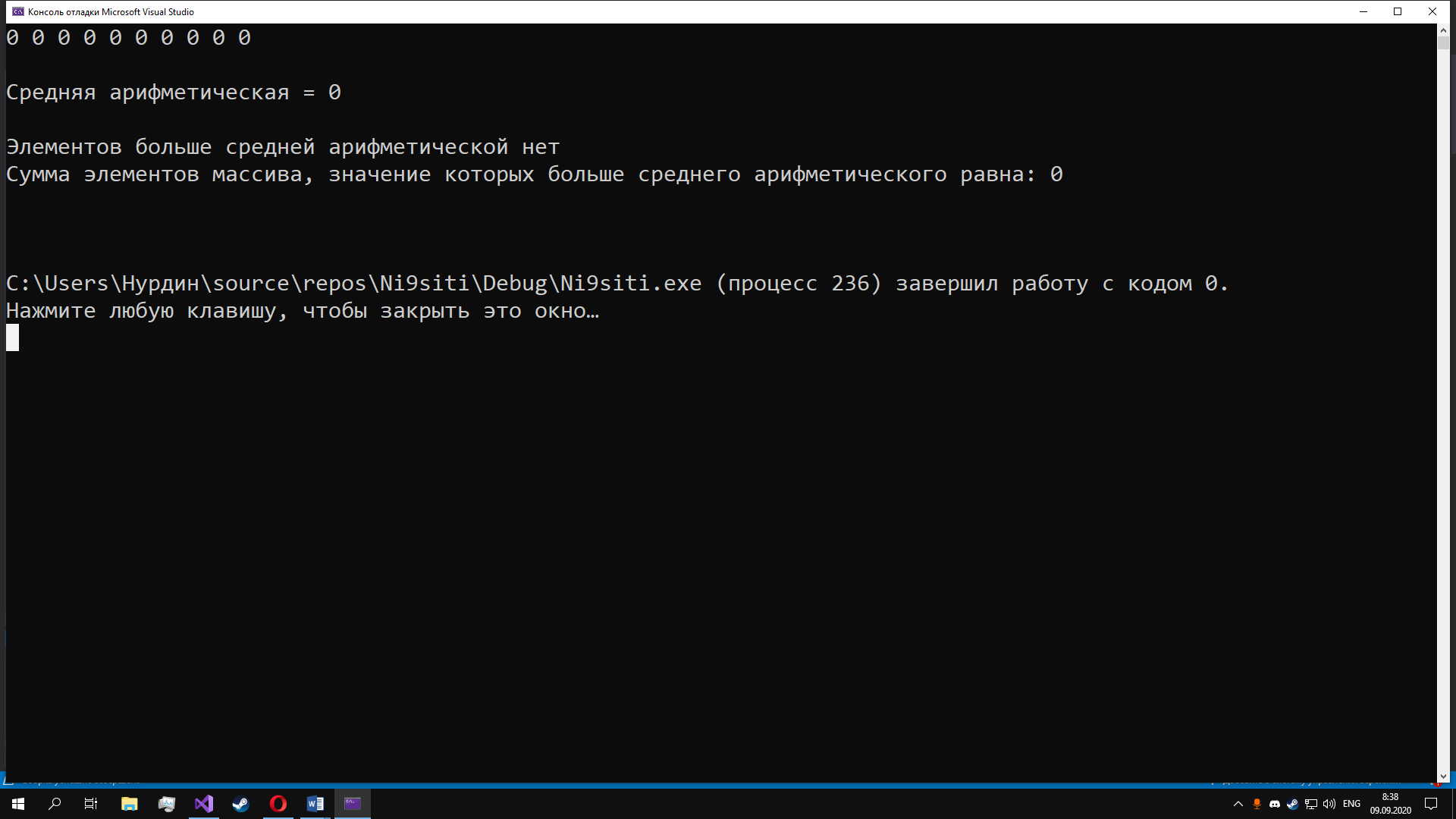


Рис 2.

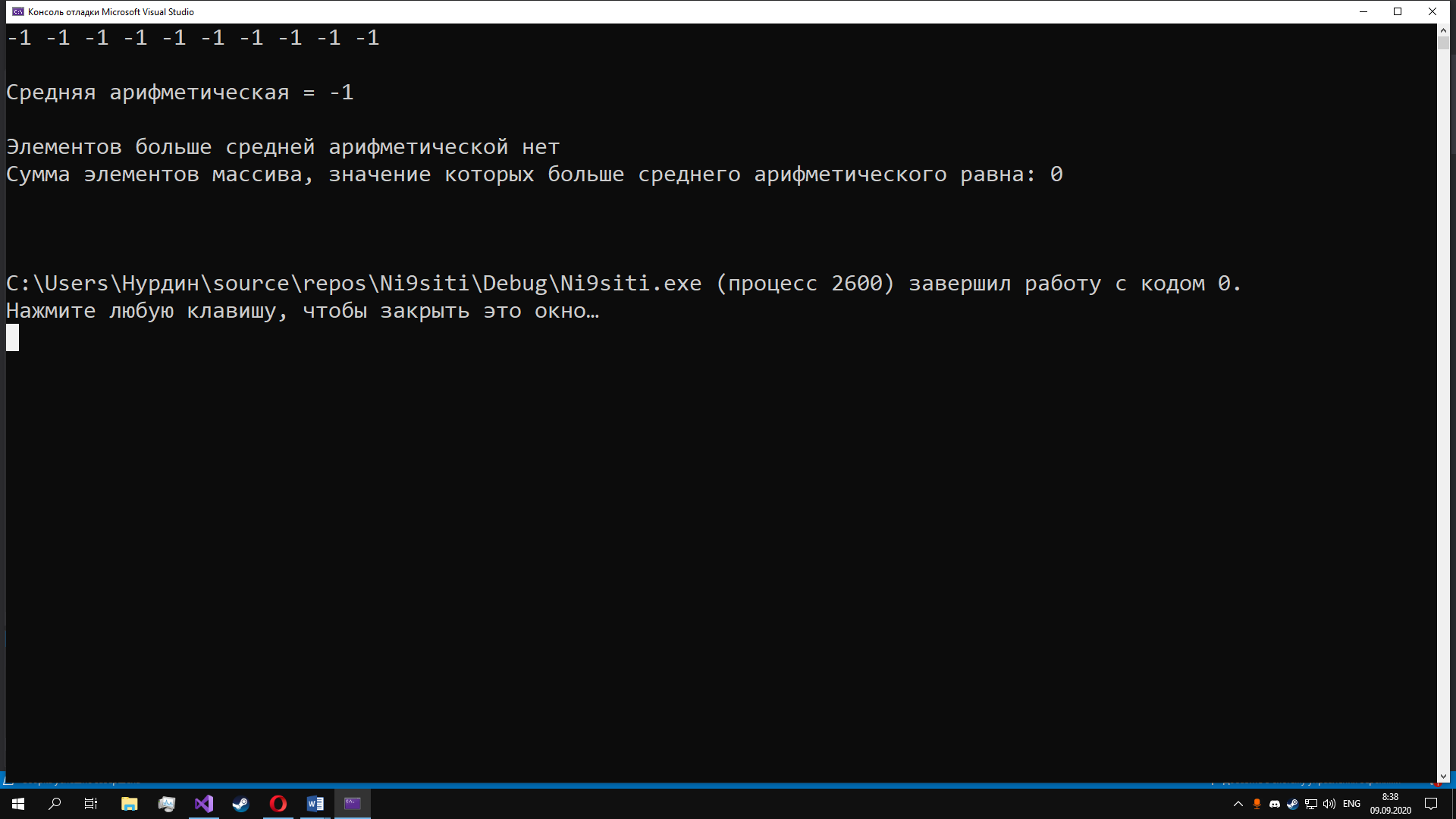


Рис 3.

**1. Постановка задачи**

Найти количество элементов массива (А=10), находящихся после наименьшего элемента.

**2. Описание входных и выходных данных**

Исходные данные:

arr **- массив, подлежащий обработке**;

n - количество реально введенных элементов массива A.

Выходные данные:

min – переменная для хранения минимального элемента массива

imin – переменная для хранения индекса минимального элемента массива

arr – элементы массива находящихся после наименьшего элемента.

kol – переменная для хранения количества элементов находящихся после наименьшего элемента.

**3. Описание алгоритма**

Сначала инициализируем все элементы массива. После в переменную **min** присваиваем **arr[0]**, то - есть первый элемент массива. После мы ищем минимальный элемент массива. Далее для красивого вывода проверяем находиться ли минимальный элемент в конце, если да, то выводим что элементов после минимального нет, иначе через цикл, выводим элементы массива, и через инкремент увеличиваем переменную **kol** для подсчета количества элементов, находящихся после наименьшего элемента. Далее это количество выводим.

Начало

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

cin >> A[i]

min = arr[0];

imin = 0;

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

True

False

arr[i] < min

min = arr[i];

imin = i;

min;  
imin+1;

imin == (n – 1)

False

True

Not have

for (i = (imin+1); i < n; i++)

arr[i]

kol++;

kol

Конец

**4. Набор тестов**

Исходные данные:

**arr = 1 5 2 4 7 4 3 9 8 2**

**n=10**

Выходные данные:

**min = 1**

**imin = 1**

**arr = 5, 2, 4, 7, 4, 9, 8, 2**

**kol = 9**

Исходные данные:

**arr = 0 0 0 0 0 0 0 0 0 0**

**n=10**

Выходные данные:

**min = 0**

**imin = 1**

**arr = 0 0 0 0 0 0 0 0 0**

**kol = 9**

Исходные данные:

**arr = 9 5 2 4 7 4 3 9 8 1**

**n=10**

Выходные данные:

**min = 1**

**imin = 1**

**arr = not have**

**kol = 9**

**5. Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

#include "Arithmetic.h"

using namespace std;

int main()

{

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int n = 10;

int arr[n], kol = 0;

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

{

cin >> arr[i];

}

int min = arr[0];

int imin = 0;

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

{

if (arr[i] < min)

{

min = arr[i];

imin = i;

}

}

cout << "\nМинимальный элемент = " << min << endl;

cout << "\nИндекс минимального элемента = " << (imin + 1) << endl;

if (imin == (n - 1))

{

cout << "\nЭлементов после минимального элемента нет" << endl;

}

else

{

cout << "\nЭлементы после минимального элемента: " << endl;

for (int i = (imin + 1); i < n; i++)

{

cout << "A[" << i + 1 << "] = " << arr[i] << endl;

kol++;

}

}

cout << "Количество элементов массива находящихся после наименьшего элемента равно: " << kol << "\n\n\n";

return 0;

}

**6. Анализ результатов и выводы**

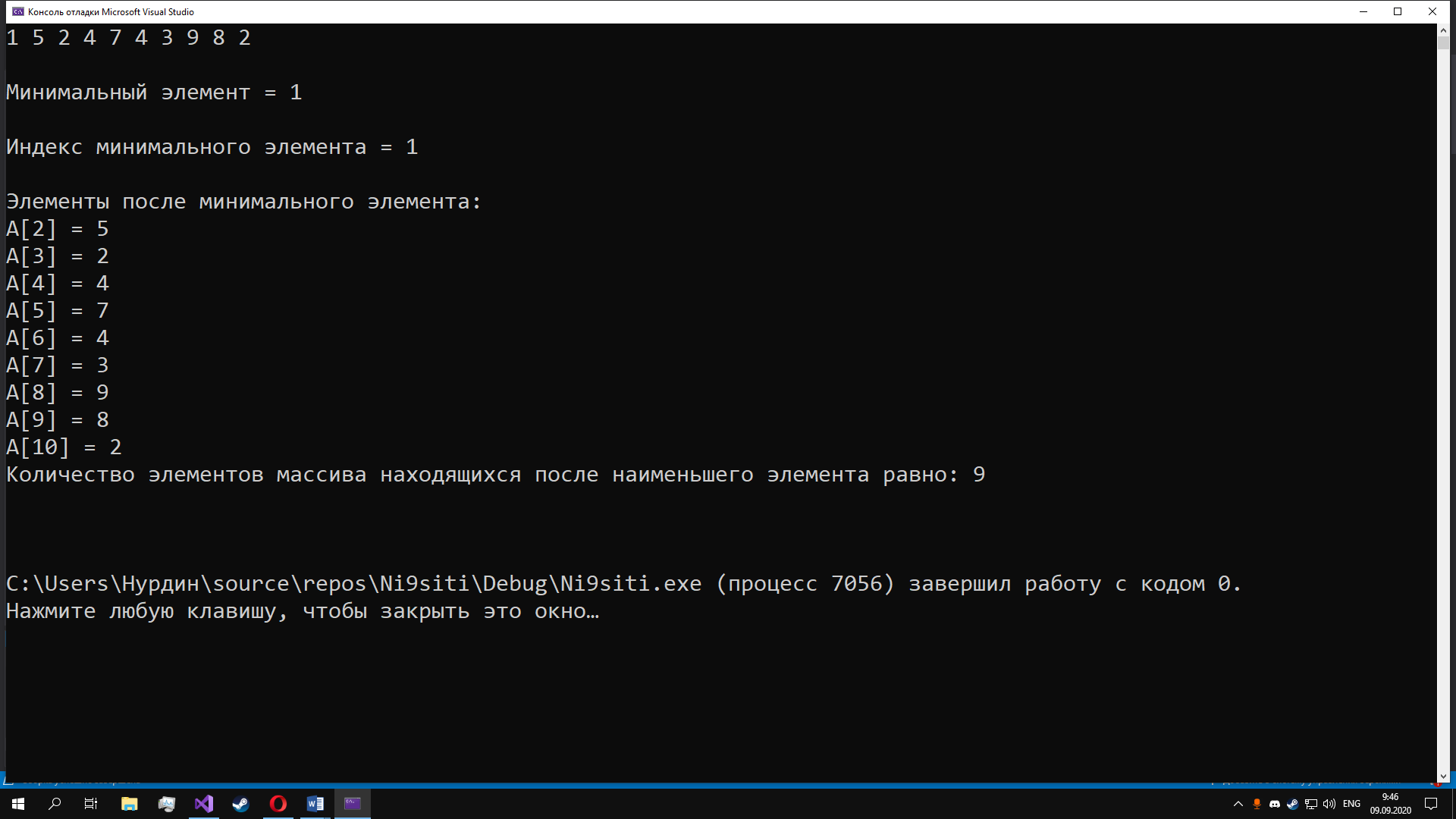


Рис 4.

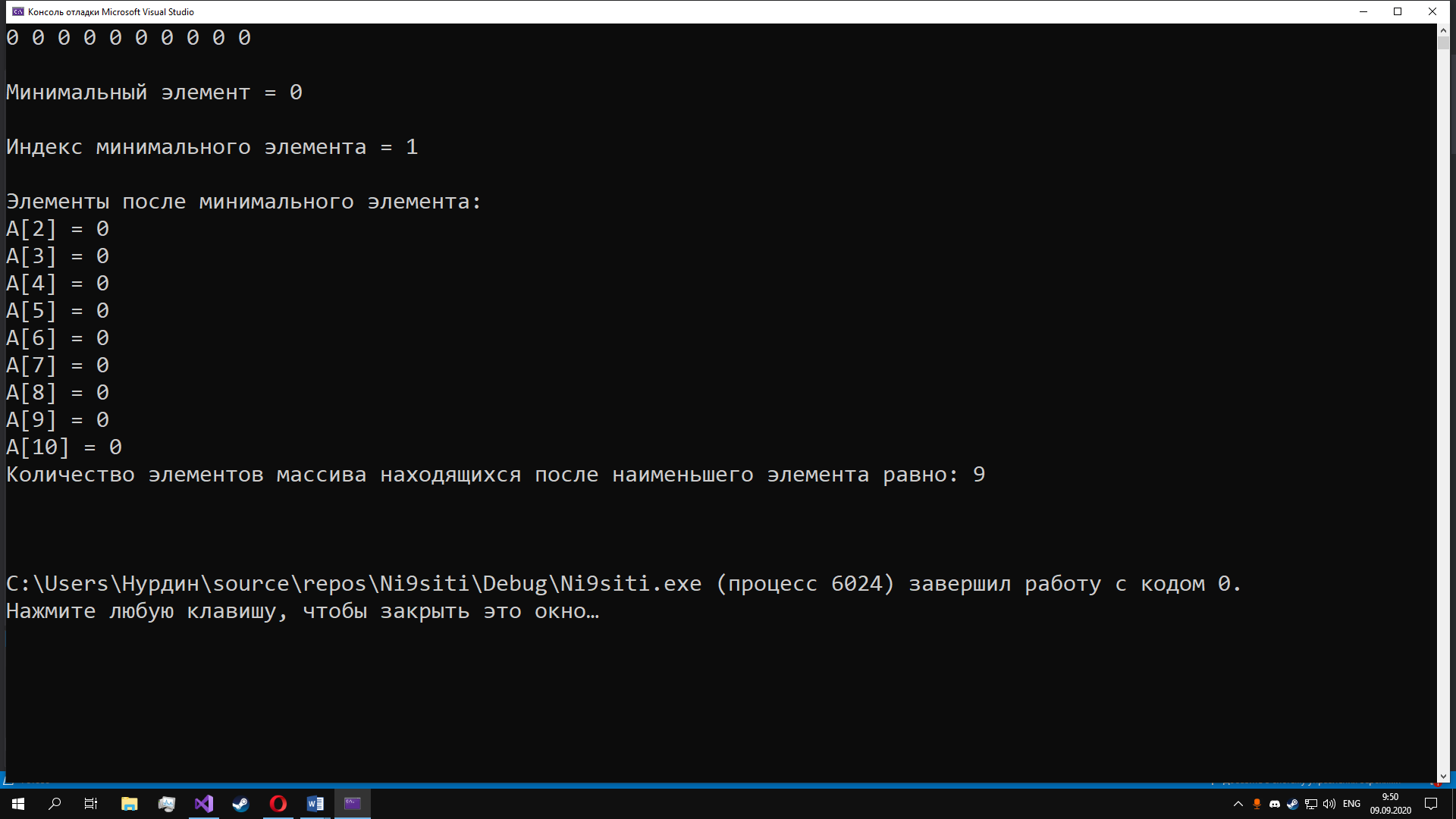


Рис 5.

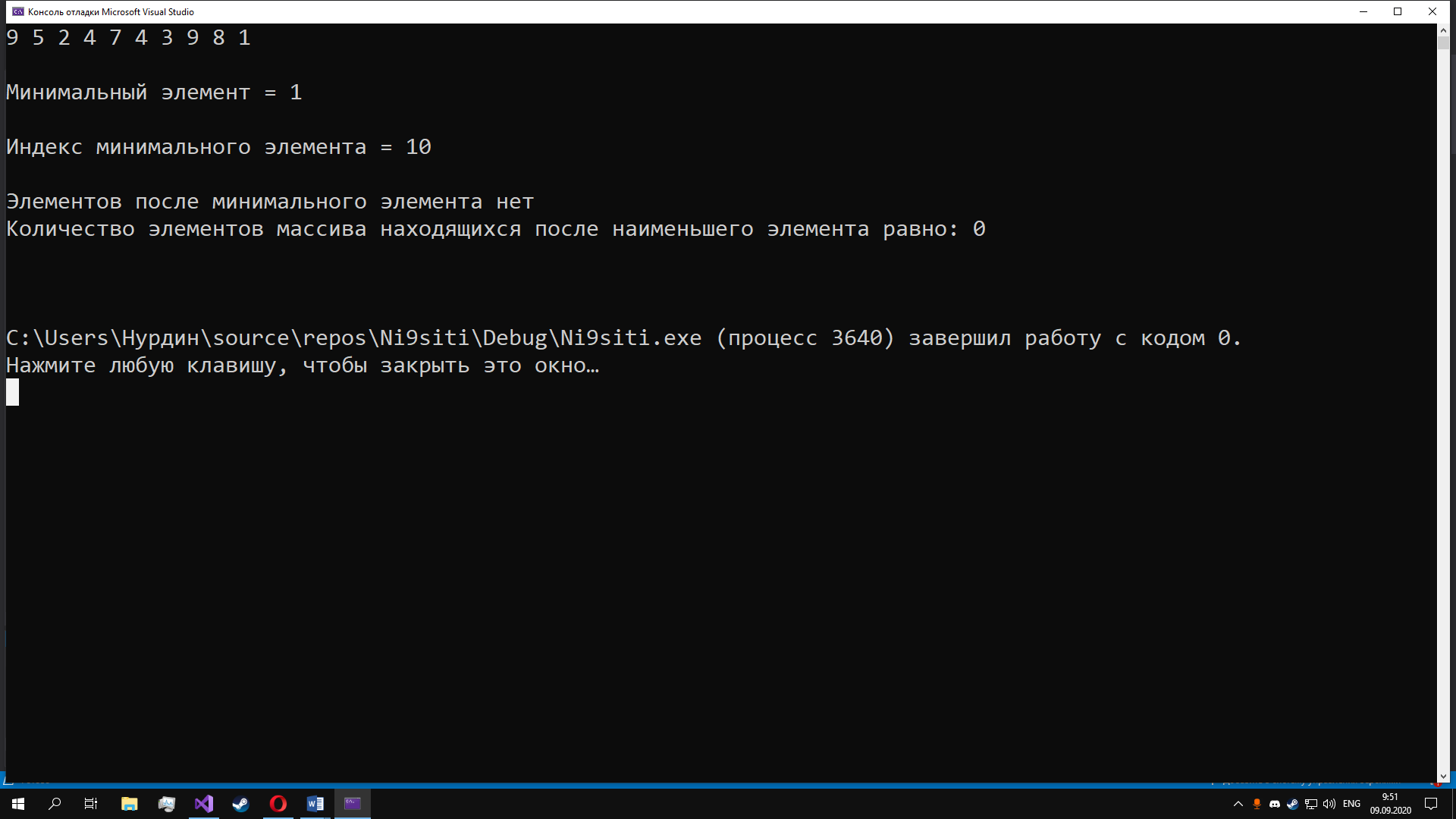


Рис 6.

**1. Постановка задачи**

В заданном 2-мерном массиве (А=3 х 3) поменять местами строки содержащие минимальный элемент и максимальный элемент массива.

**2. Описание входных и выходных данных**

Исходные данные:

arr **- массив, подлежащий обработке**;

n - количество реально введенных элементов массива A.

Выходные данные:

min – переменная для хранения минимального элемента массива

imin – переменная для хранения индекса минимального элемента массива по строке

jmin – переменная для хранения индекса минимального элемента массива по столбцу

max – переменная для хранения максимального элемента массива

imax – переменная для хранения индекса максимального элемента массива по строке

jmax – переменная для хранения индекса максимального элемента массива по столбцу

**3. Описание алгоритма**

Сначала инициализируем все элементы массива. После в переменные **min** и **max** присваиваем **arr[0][0]**, то - есть первый элемент массива. После мы ищем минимальный и максимальный элемент массива и их индексы. Далее мы, меняем местами по индексу строки содержащие минимальный элемент и максимальный элемент массива.

Начало

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

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

cin >> A[i][j]

system("cls");

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

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

cout >> A[i][j]

min = arr[0][0], max = arr[0][0];

imin = 0, imax = 0;

jmin = 0, jmax = 0;

min, imin, jmin;

max, imax, jmax;

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

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

False

True

arr[i][j] > max

max = arr[i][j];

imax = i; jmax = j;

False

True

arr[i][j] > min

min = arr[i][j]; imin = i; jmin = j;

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

swap(arr[imin][i], arr[imax][i]);

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

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

cout >> A[i][j]

Конец

**4. Набор тестов**

Исходные данные:

**arr = 1 2 3**

**4 5 6**

**7 8 9**

**n=3**

Выходные данные:

**min = 1 imin = 1 jmin = 1**

**max = 9 imax = 3 jmax = 3**

**arr = 7 8 9**

**4 5 6**

**1 2 3**

Исходные данные:

**arr = 9 5 2**

**4 7 4**

**3 8 1**

**n=3**

Выходные данные:

**min = 1 imin = 3 jmin = 3**

**max = 9 imax = 1 jmax = 1**

**arr = 3 8 1**

**4 7 4**

**9 5 2**

**5. Текст программы**

#include <iostream>

using namespace std;

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int n = 3;

int arr[n][n];

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

for (int j = 0; j < n; j++){

cin >> arr[i][j];

}

}

system("cls");

cout << "Начальный массив:" << endl;

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

for (int j = 0; j < n; j++){

cout << "A[" << i + 1 << "][" << j + 1 << "]=" << arr[i][j] << "\t";

}

cout << endl;

}

int min = arr[0][0], max = arr[0][0];

int imin = 0, imax = 0;

int jmin = 0, jmax = 0;

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

for (int j = 0; j < n; j++){

if (arr[i][j] > max){

max = arr[i][j];

imax = i;

jmax = j;

}

if (arr[i][j] < min){

min = arr[i][j];

imin = i;

jmin = j;

}}}

cout << "\n\n";

cout << "Минимальный элемент стоящий в столбце № " << (jmin + 1) << " строке № " << (imin + 1) << " = " << min << endl;

cout << "Максимальный элемент стоящий в столбце № " << (jmax + 1) << " строке № " << (imax + 1) << " = " << max << endl;

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

swap(arr[imin][i], arr[imax][i]);

}

cout << "\n\nИзмененный массив:" << endl;

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

for (int j = 0; j < n; j++){

cout << "A[" << i + 1 << "][" << j + 1 << "]=" << arr[i][j] << "\t";

}

cout << endl;

}

return 0;}

**6. Анализ результатов и выводы**

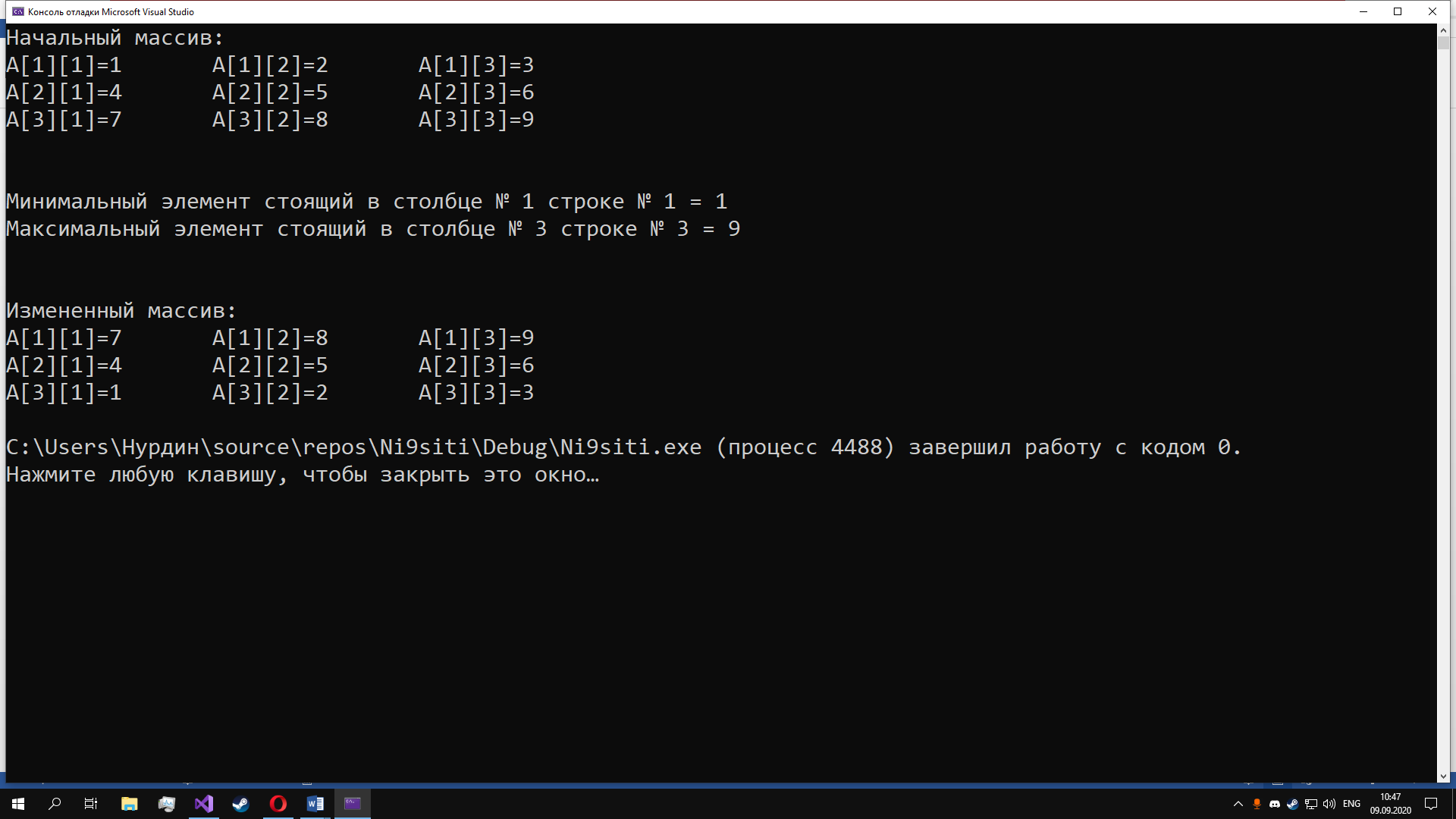


Рис 7.

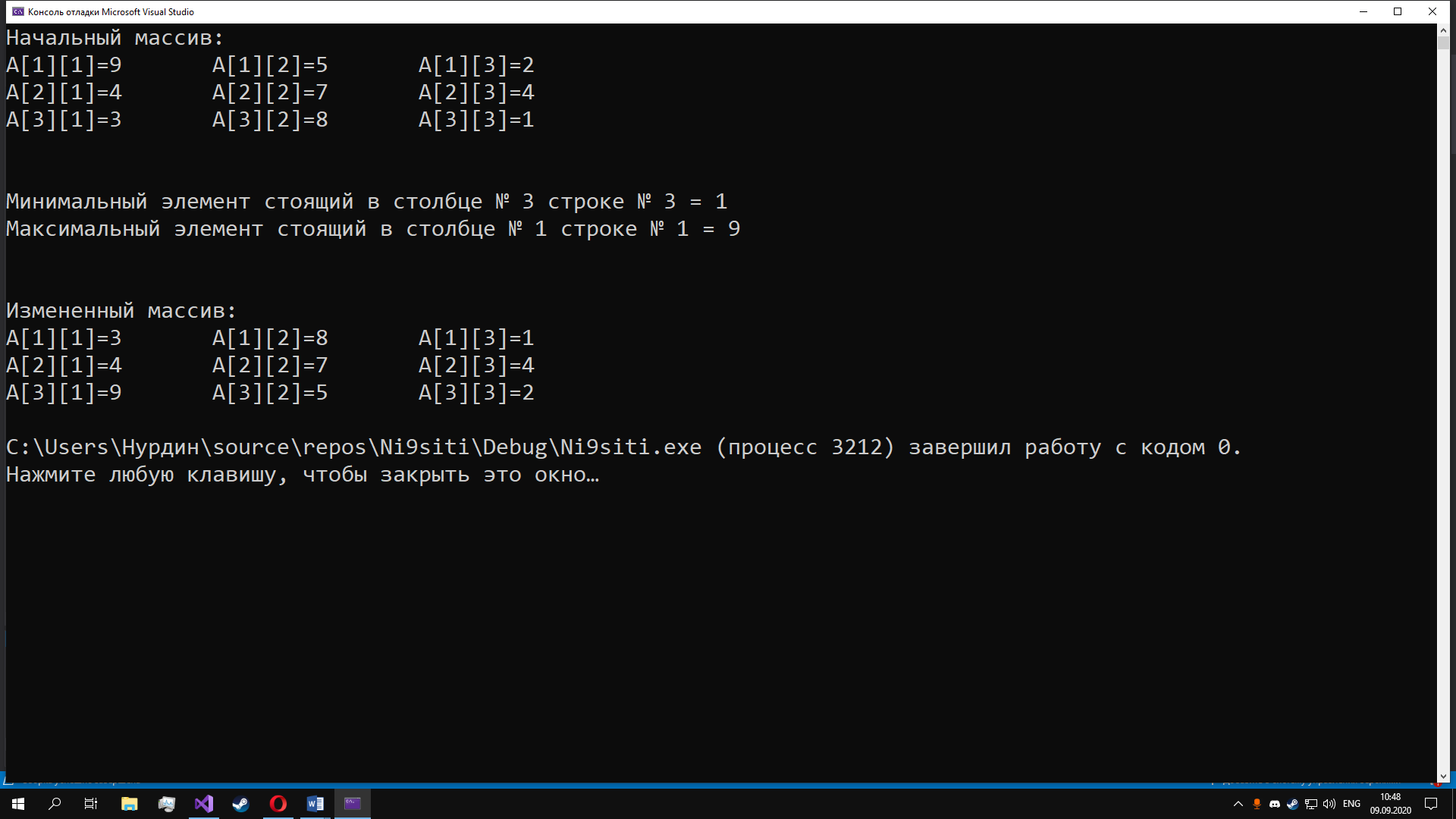


Рис 8.

**1. Постановка задачи**

Отсортировать массив методом «пузырька»

**2. Описание входных и выходных данных**

Исходные данные:

arr **- массив, подлежащий обработке**;

n - количество реально введенных элементов массива A.

Выходные данные:

arr – отсортированный массив по возрастанию.

**3. Описание алгоритма**

Сначала инициализируем все элементы массива. После мы попарно сравниваем соседние элементы. Сортировка в худшем случае составляет (((n\*n)-n) / 2) операций, в лучшем случае - 0 операций, а в количестве операций сравнения двух соседних элементов массива в худшем случае – (((n\*n)+n) / 2-1) операций, в лучшем - n-1 операций.

Начало

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

cin >> A[i]

system("cls");

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

cout >> A[i]

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

for (i = 0; i < (n-1); i++)

arr[i] > arr[i + 1]

False

True

swap(arr[i], arr[i + 1]);

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

Конец

cout >> A[i]

**4. Набор тестов**

Исходные данные:

**arr = 9 8 7 6 5 4 3 2 1 0**

**n=10**

Выходные данные:

**Arr = 0 1 2 3 4 5 6 7 8 9**

Исходные данные:

**arr = 1 5 2 4 7 4 3 9 8 2**

**n=10**

Выходные данные:

**Arr = 1 2 2 3 4 4 5 7 8 9**

Исходные данные:

**arr = 9 5 2 4 7 4 3 9 8 1**

**n=10**

Выходные данные:

**arr = 1 2 3 4 4 5 7 8 9 9**

**5. Текст программы**

#include <iostream>

#include <ctime>

#include <string>

#include <fstream>

#include <iomanip>

#include "Windows.h"

#include "Arithmetic.h"

using namespace std;

int main()

{

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const int n = 10;

int arr[n];

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

{

cin >> arr[i];

}

system("cls");

cout << "Начальный массив: " << endl;

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

{

cout << "A[" << i << "] = " << arr[i] << "\n";

}

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

{

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

{

if (arr[i] > arr[i + 1])

{

swap(arr[i], arr[i + 1]);

}

}

}

cout << "\n\nОтсортированный массив\n\n" << endl;

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

{

cout << "A[" << i << "] = " << arr[i] << "\n";

}

return 0;

}

**6. Анализ результатов и выводы**

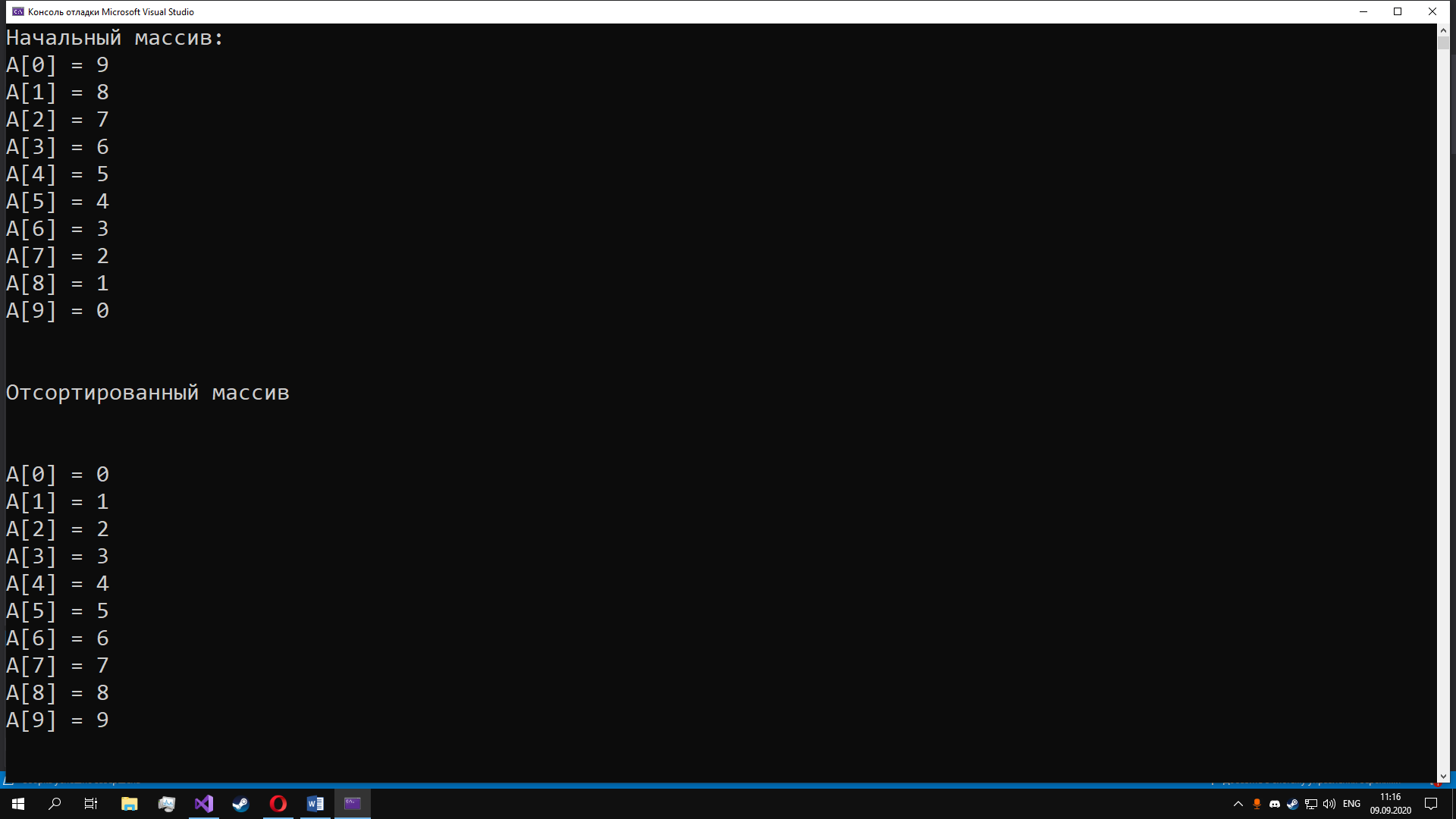


Рис 9.

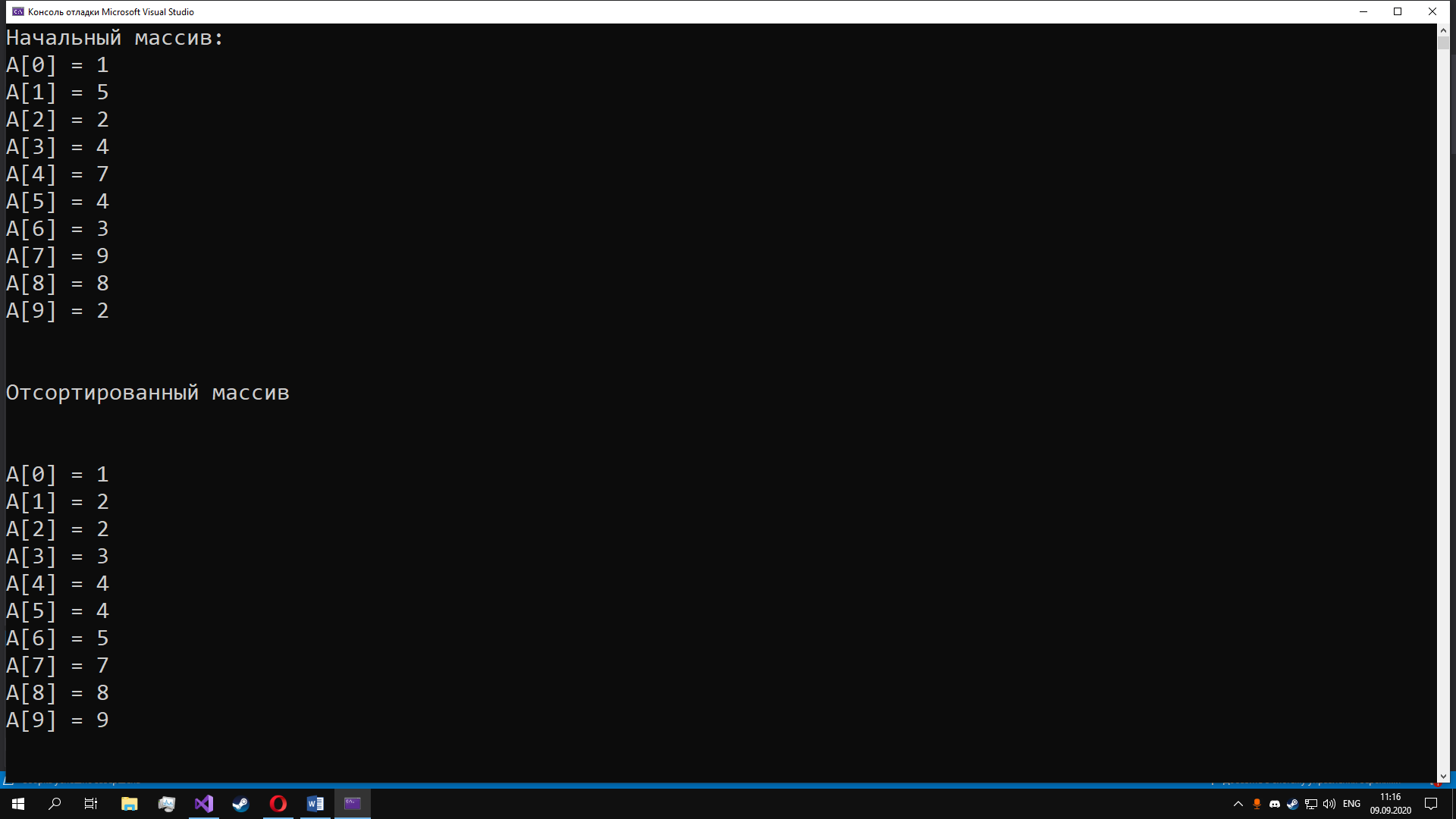


Рис 10.

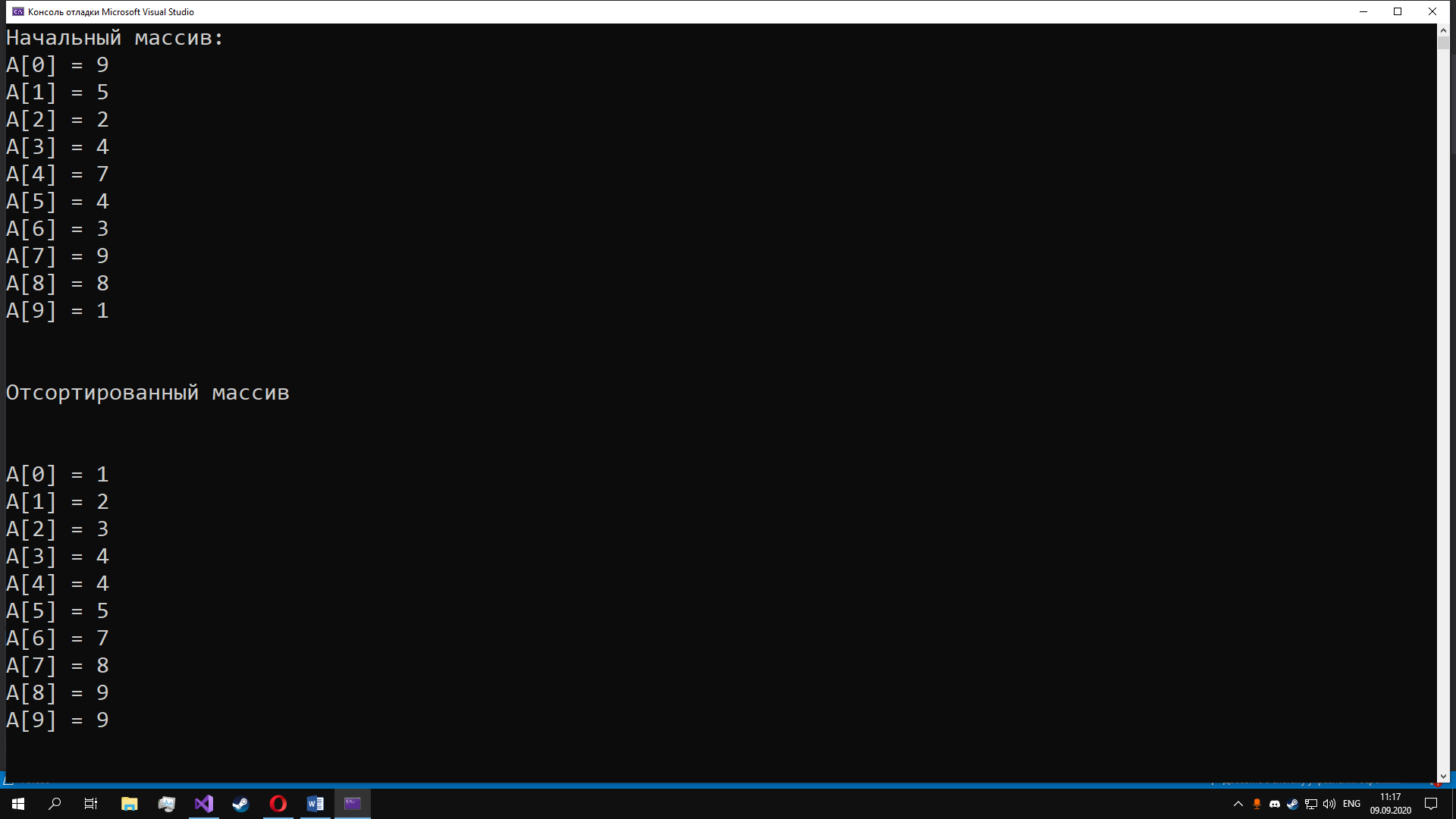


Рис 11.