Пензенский государственный университет

Кафедра "Вычислительной техники"

**Отчёт**

По лабораторной работе №2

по дисциплине "Л и ОА в ИТ"

на тему "Оценка времени выполнения программ"

***Выполнил студент группы 19ВВ1:***

*Кочетов Д.В.*

***Приняли:***

*Митрохин М. А.*

*Юрова О. В.*

Пенза 2020

**Цель работы:** изучить принципы и методы оценки времени выполнения программ в языке Си.

**Лабораторное задание**

1. Вычислить порядок сложности программы (О-символику).
2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.
4. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
5. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
6. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
7. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
8. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

**Теоретическая часть**

Для оценки времени выполнения программ языка Си или их частей могутиспользоваться средства, предоставляемые библиотекой time.h. Данная библиотекасодержит описания типов и прототипы функций для работы с датой и временем.

Типы данных:

1. clock\_t - возвращается функцией clock(). Обычно определён как int или long int.
2. time\_t - возвращается функцией time(). Обычно определён как int или long int.
3. struct tm - нелинейное, дискретное календарное представление времени.

Основные функции:

1. clock\_t clock(void) - возвращает время, измеряемое процессором в тактах от начала выполнения программы, или −1, если оно не известно. Пересчет этого времени в секунды выполняется по формуле:

clock() / CLOCKS\_PER\_SEC

где CLOCKS\_PER\_SEC – константа, определяющая количество тактов системных часов в секунду.

1. time\_t time(time\_t \*tp) Возвращает текущее календарное время или −1, если это время не известно. Если указатель tp не равен NULL, то возвращаемое значение записывается также и в \*tp.
2. double difftime(time\_t time2,time\_t time1) Возвращает разность time2-time1, выраженную в секундах.

**Практическая часть**

**Листинг**

Файл: Lab2.cpp

#include "stdafx.h"

#include "windows.h"

#include "stdio.h"

#include "time.h"

#include "conio.h"

#include "locale.h"

#define SIZE 100

int \_tmain(int argc, \_TCHAR\* argv[])

{

srand(time(NULL));

setlocale(LC\_ALL, "Rus");

DWORD Start = 0, End = 0;

int A[SIZE][SIZE], B[SIZE][SIZE], C[SIZE][SIZE], elem\_c = 0, i = 0, j = 0, r = 0;

while(i < SIZE) {

while(j < SIZE) {

A[i][j] = rand() % 100 + 1;

j++;

}

i++;

j = 0;

}

i = 0; j = 0;

while(i < SIZE) {

while(j < SIZE) {

B[i][j] = rand() % 100 + 1;

j++;

}

i++;

j = 0;

}

Start = GetTickCount();

for(i = 0; i < SIZE; i++) {

for(j = 0; j < SIZE; j++) {

elem\_c = 0;

for(r = 0; r < SIZE; r++) {

elem\_c = elem\_c + A[i][r] \* B[r][j];

C[i][j] = elem\_c;

}

}

}

End = GetTickCount();

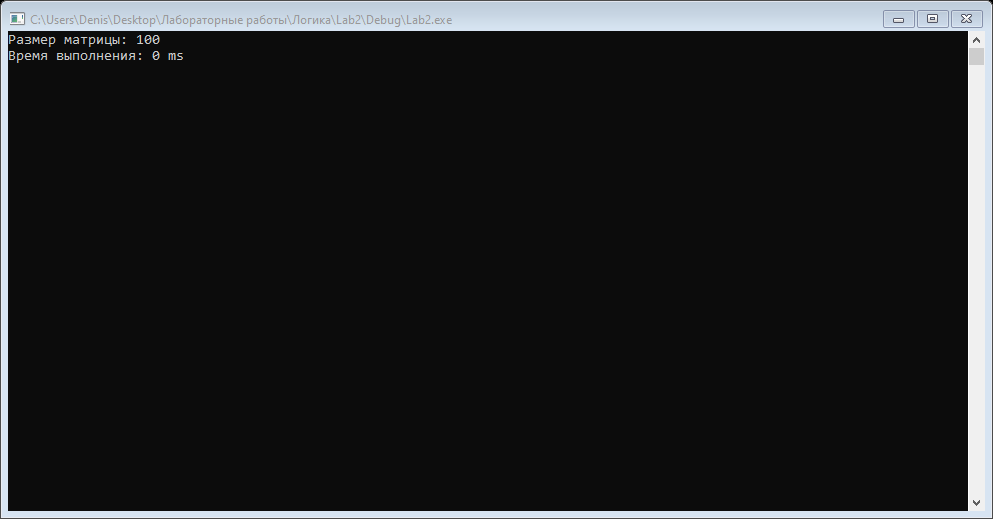
printf("Размер матрицы: %d\nВремя выполнения: %d ms", SIZE, End - Start);

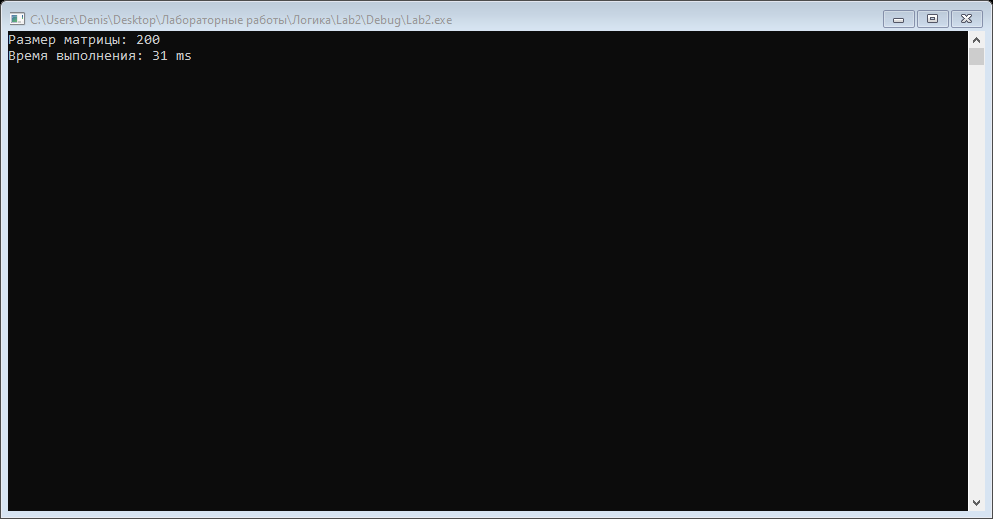
getch();

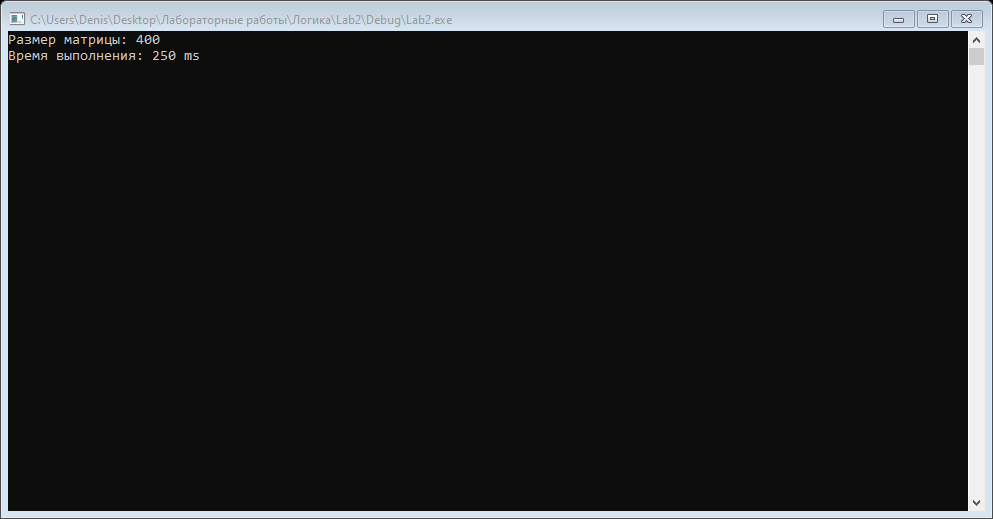
return 0;

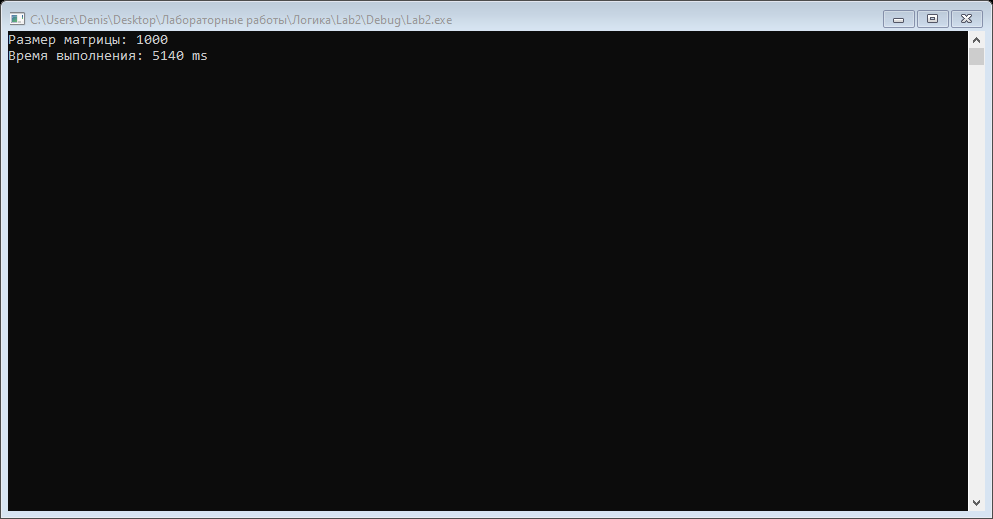
}

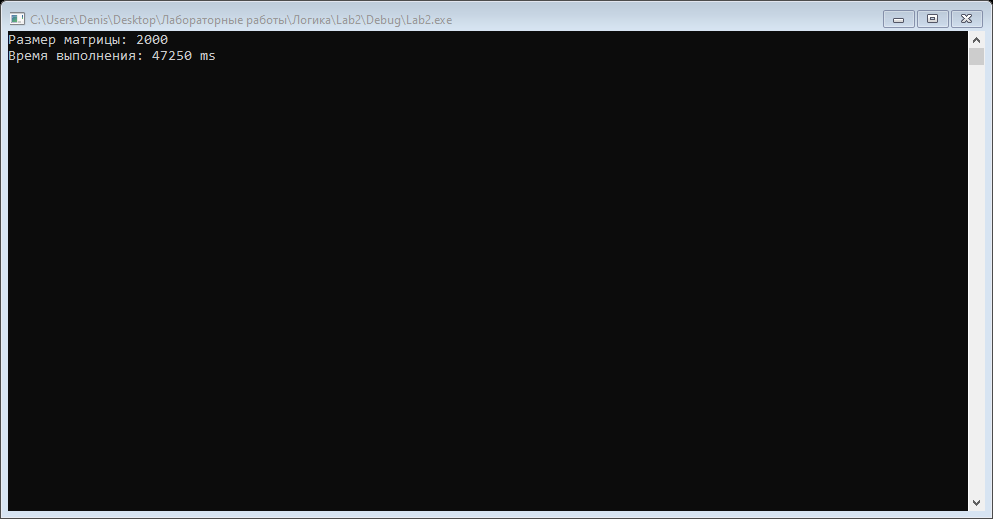
1. Порядок сложности алгоритма – O(n3).
2. Результаты выполнения программы для различных размеров матриц:

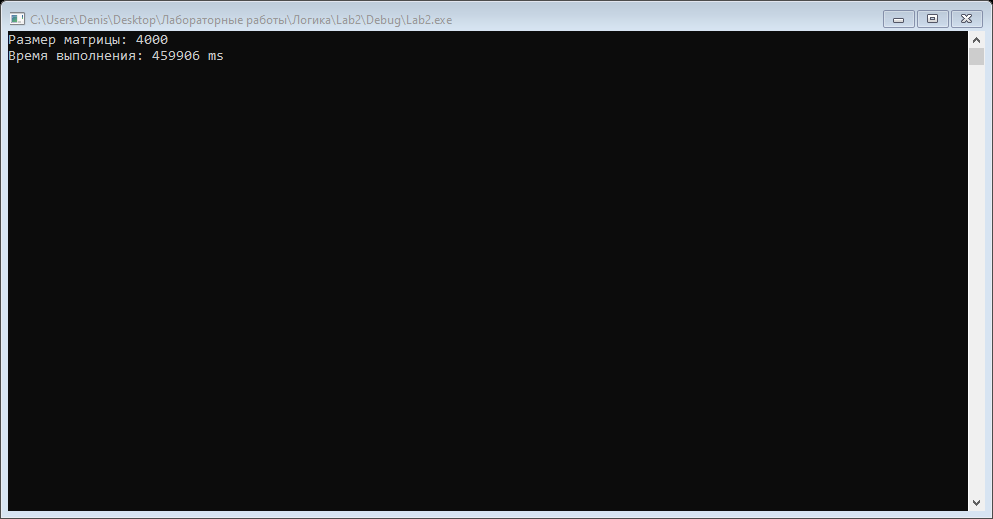


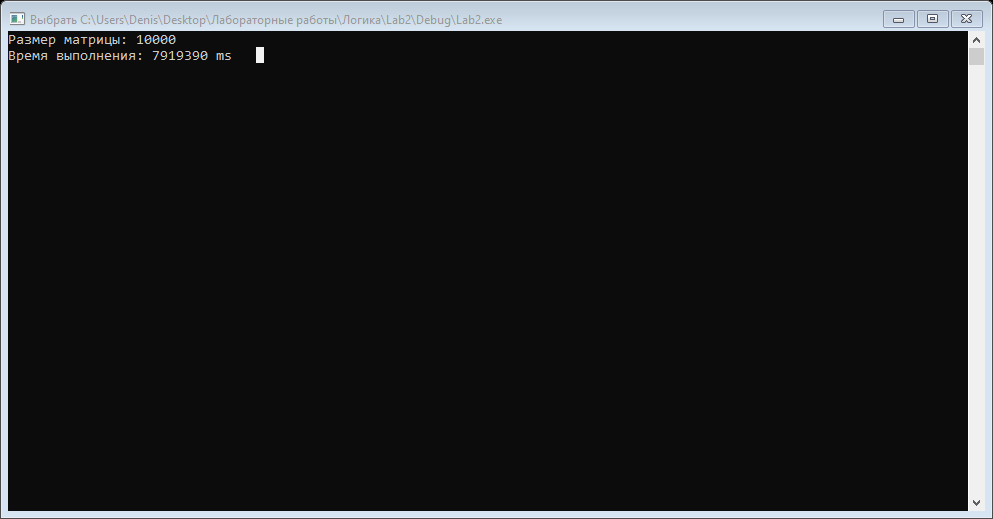








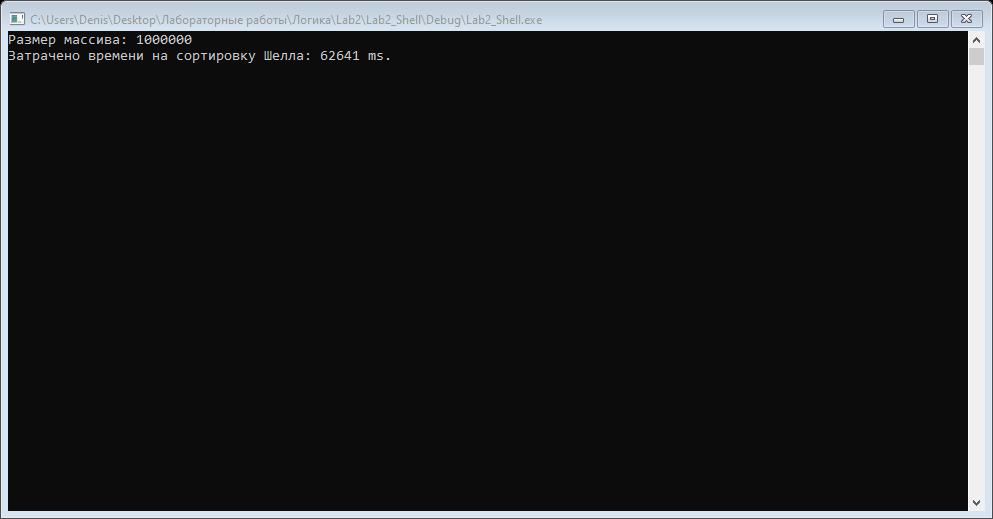


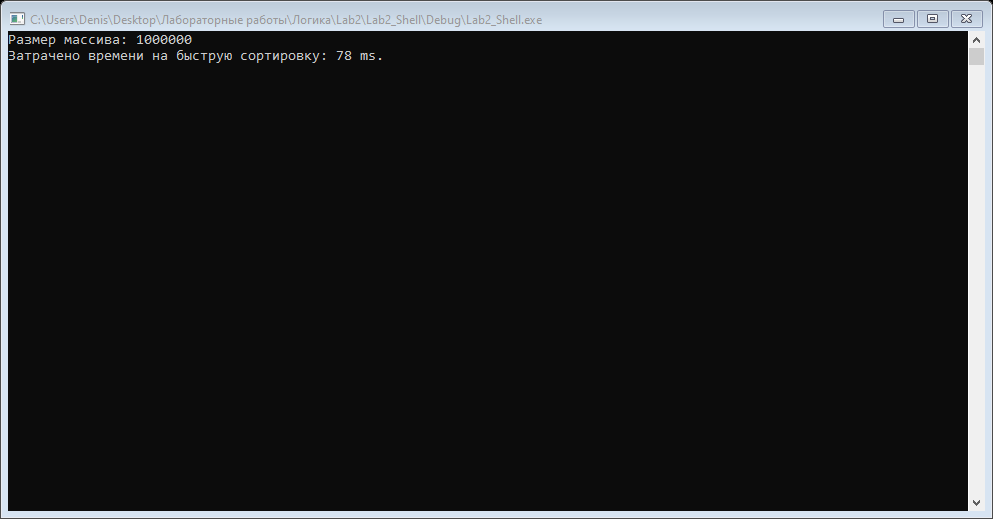


1. Ниже приведены графики времени выполнения алгоритма.

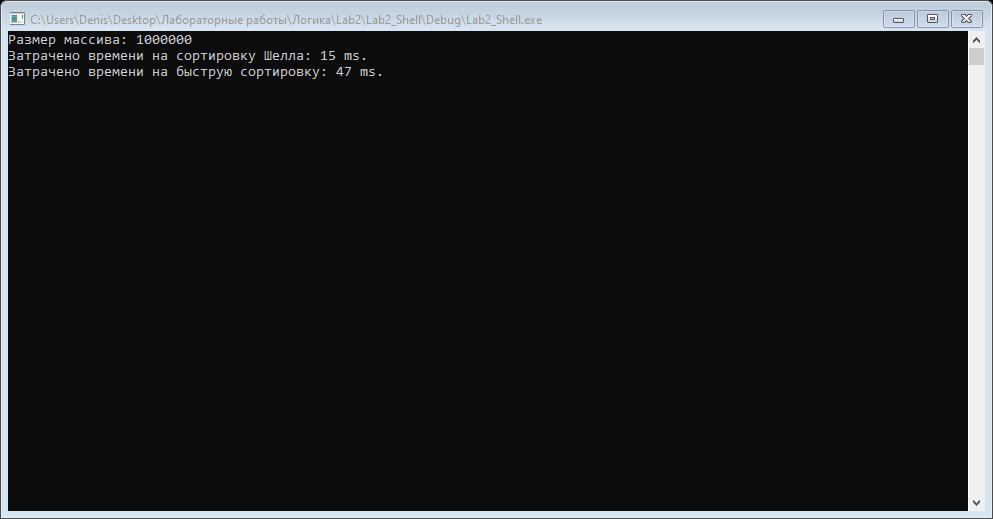
Действительное время выполнения всегда больше расчётного, и с увеличением числа данных растёт и разница во времени.

1. Оценка времени работы каждого из реализованных алгоритмов на случайном наборе значений массива.

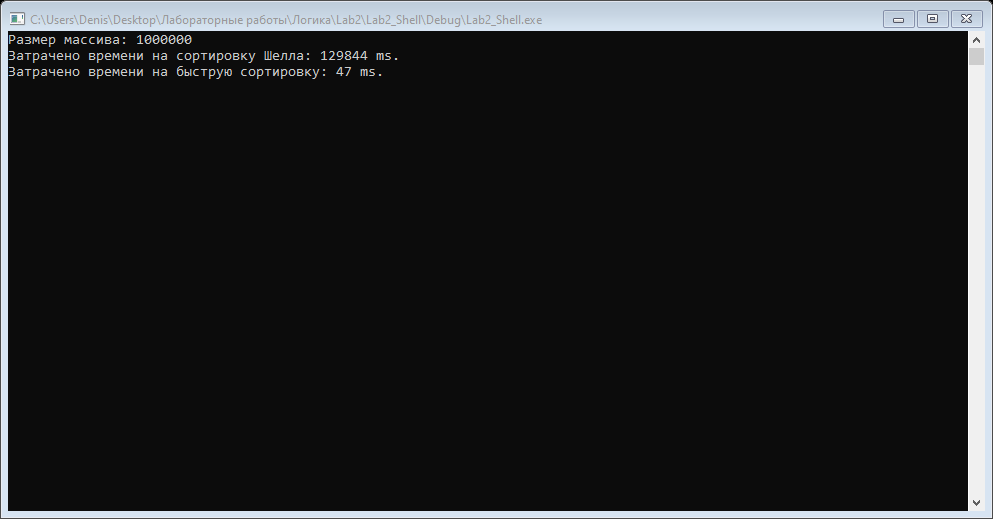




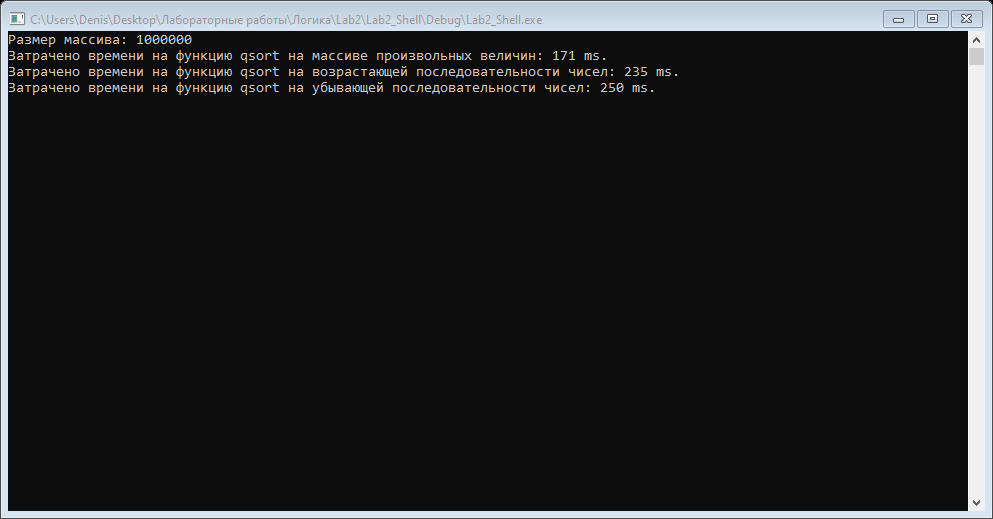
1. Оценка времени работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.



1. Оценка времени работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.

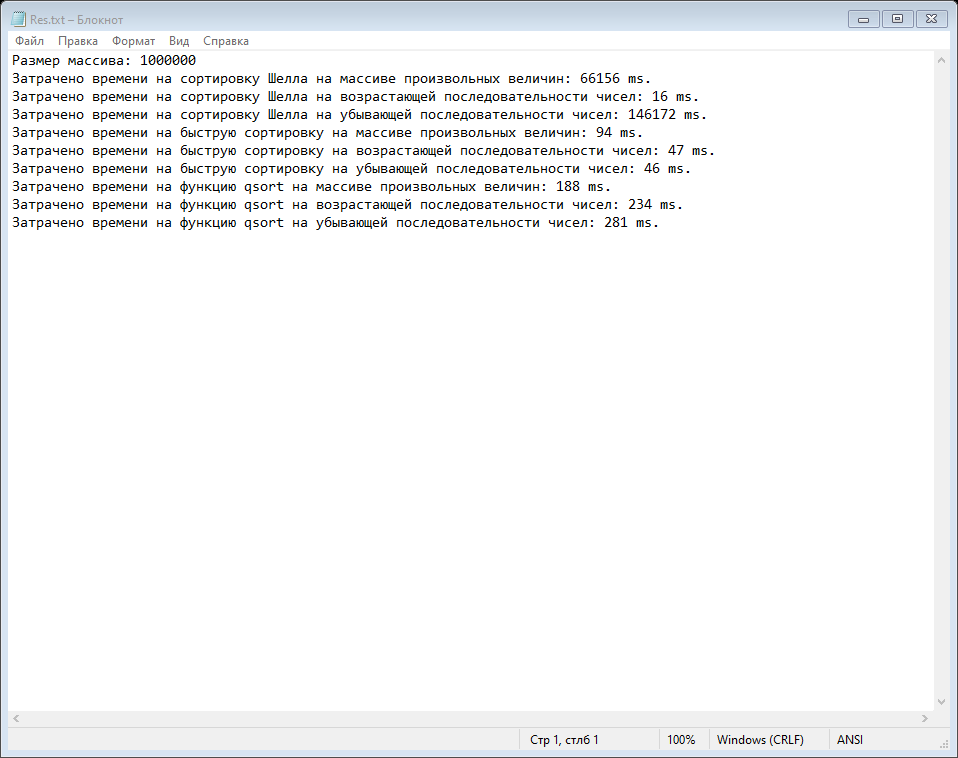


1. Оценка времени работы стандартной функции qsort на выше указанных наборах данных.



**Вывод:** я изучил принципы и методы оценки времени выполнения программ в языке Си.

Результаты работы алгоритмов сортировки:



Из полученных данных следует вывод, что сортировку Шелла выгоднее использовать на уже отсортированном или почти отсортированном наборе данных, в отличие от быстрой сортировки. Однако, стандартная функция qsort библиотеки языка Си более универсальна и работает с почти одинаковой скоростью на любых наборах данных.