Пензенский государственный университет  
Кафедра «Вычислительной техники»

**Отчет**по лабораторной работе №2  
по дисциплине:«Логика и основы алгоритмизации в инженерных задачах»  
на тему: «**Оценка времени выполнения программ**»

**Выполнил студент группы 19ВВ1:**

Яцков В.В

**Приняли:**

д.т.н. профессор Митрохин М. А.

Пенза 2020.

# Цель работы: Оценить время работы программ.

**Лабораторное задание:**

**Задание 1:**

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.

**Задание 2**:

1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
5. 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 могут использоваться средства, предоставляемые библиотекой **time.h**. Данная библиотека содержит описания типов и прототипы функций для работы с датой и временем.

Типы данных:

1. clock\_t - возвращается функцией clock(). Обычно определён как int или longint.

2. time\_t - возвращается функцией time(). Обычно определён как int или longint.

3. structtm - нелинейное, дискретное календарное представление времени.

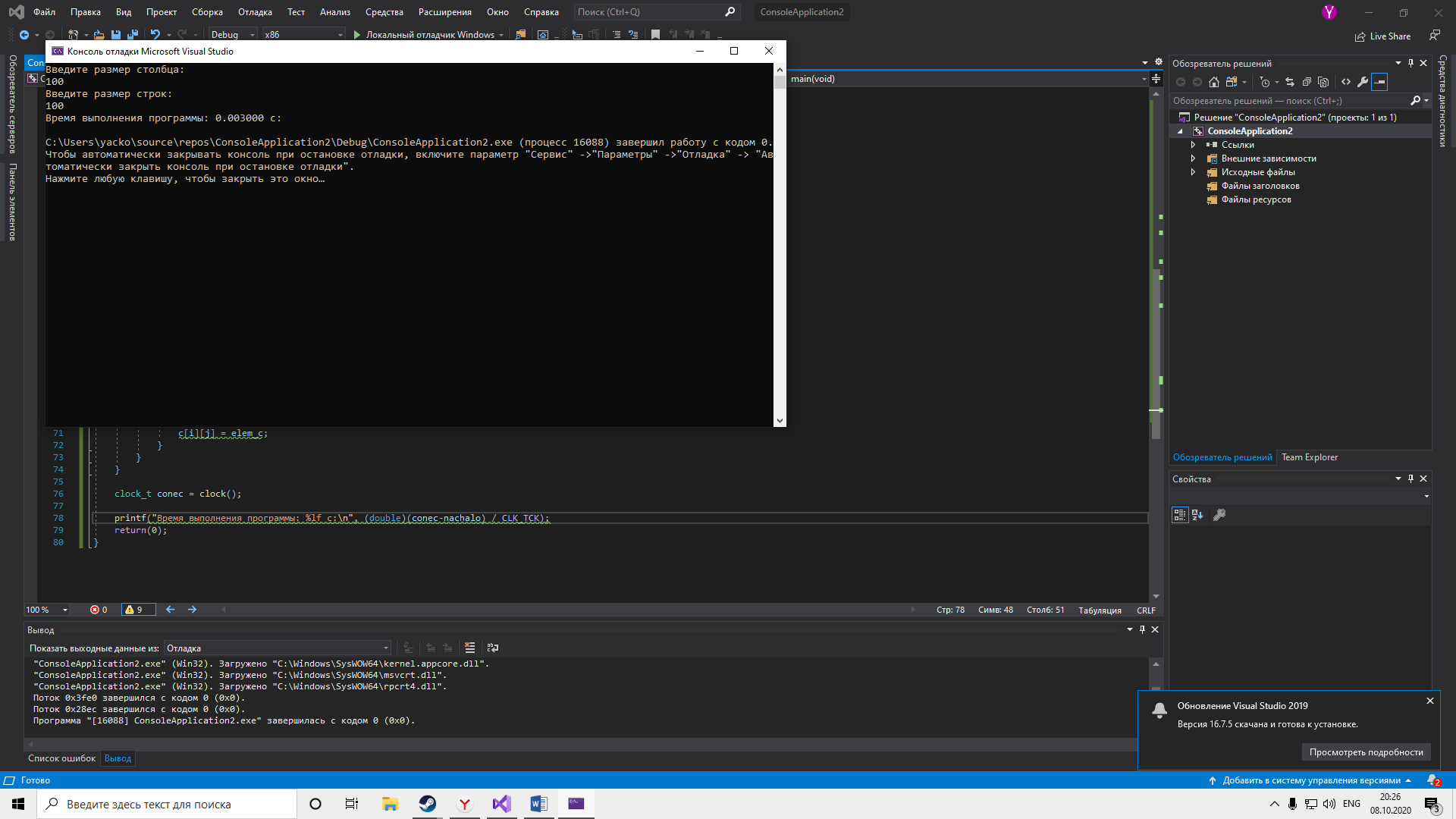
**Практическаячасть:**

**Задание 1.**

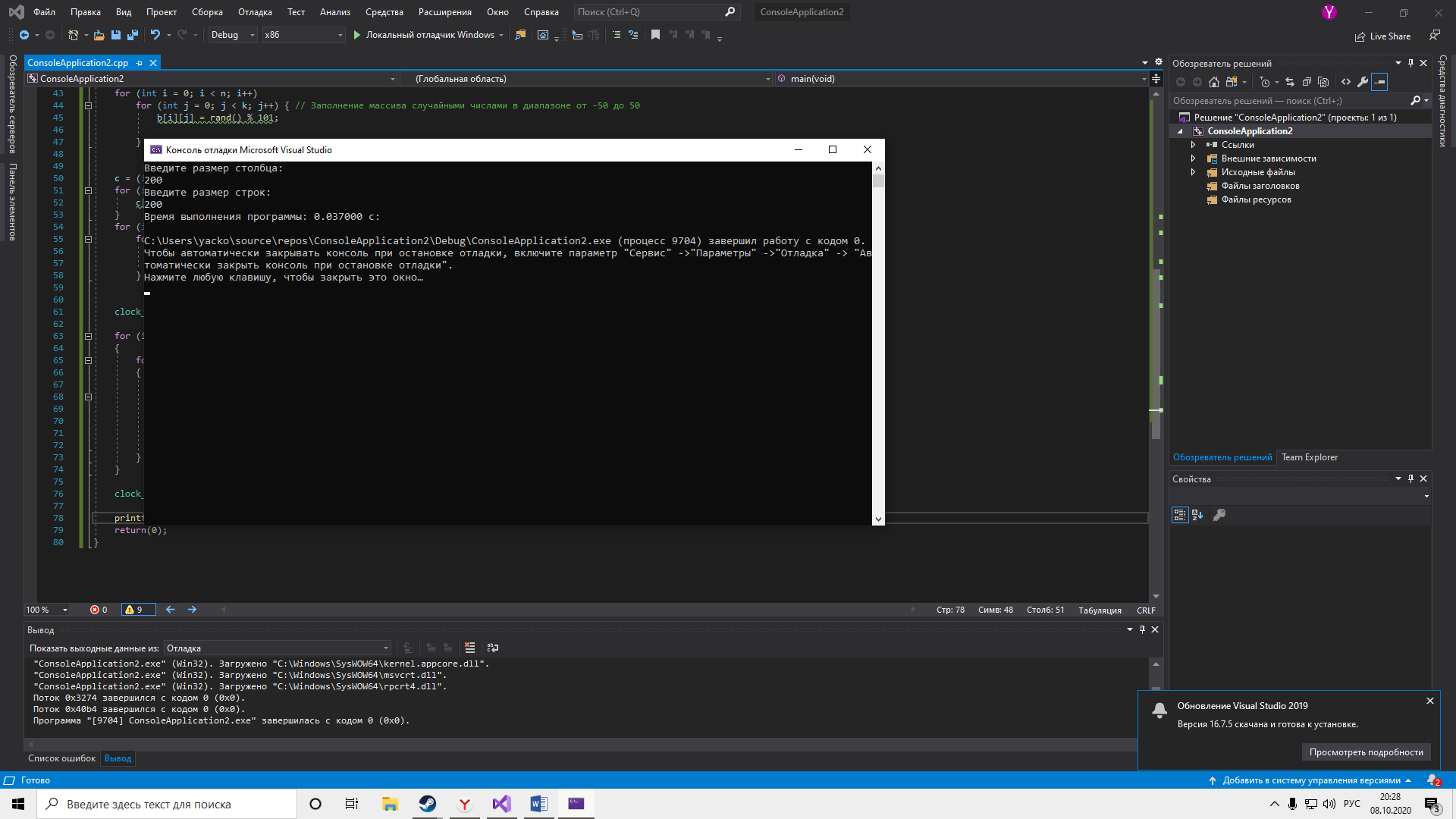
**Листинг:**

1. [#define](https://vk.com/im?sel=161635097&st=%23define) \_CRT\_SECURE\_NO\_WARNINGS  
   [#include](https://vk.com/im?sel=161635097&st=%23include) <windows.h>  
   [#include](https://vk.com/im?sel=161635097&st=%23include) <stdio.h>  
   [#include](https://vk.com/im?sel=161635097&st=%23include) <conio.h>  
   [#include](https://vk.com/im?sel=161635097&st=%23include) <math.h>  
   [#include](https://vk.com/im?sel=161635097&st=%23include) <time.h>  
     
   intmain(void)  
   {  
     
   SetConsoleCP(1251);  
   SetConsoleOutputCP(1251);  
   setvbuf(stdin, NULL, \_IONBF, 0);  
   setvbuf(stdout, NULL, \_IONBF, 0);  
   // объявляем переменные для определения времени выполнения  
   int i = 0, j = 0, r;  
   int \*\*a, \*\*b, \*\*c, elem\_c;  
     
   int n, k;  
     
     
     
   printf("Введите значение n\n");  
   scanf\_s("%d", &n);  
   printf("Введите значение k\n");  
   scanf\_s("%d", &k);  
     
   a = (int\*\*)malloc(n \* sizeof(int\*));  
   for (int i = 0; i < n; i++) {  
   a[i] = (int\*)malloc(k \* sizeof(int));  
   }  
   for (int i = 0; i < n; i++)  
   for (int j = 0; j < k; j++) { // Заполнение массива случайными числами в диапазоне от -50 до 50  
   a[i][j] = rand() % 101;  
     
   }  
     
   i = 0; j = 0;  
   b = (int\*\*)malloc(n \* sizeof(int\*));  
   for (int i = 0; i < n; i++) {  
   b[i] = (int\*)malloc(k \* sizeof(int));  
   }  
   for (int i = 0; i < n; i++)  
   for (int j = 0; j < k; j++) { // Заполнение массива случайными числами в диапазоне от 0 до 100  
   b[i][j] = rand() % 101;  
     
   }  
     
     
   c = (int\*\*)malloc(n \* sizeof(int\*));  
   for (int i = 0; i < n; i++) {  
   c[i] = (int\*)malloc(k \* sizeof(int));  
   }  
   for (int i = 0; i < n; i++)  
   for (int j = 0; j < k; j++) { // Заполнение массива случайными числами в диапазоне от -50 до 50  
   elem\_c = 0;  
     
   }  
     
     
   clock\_t t0 = clock();  
     
   for (i = 0; i < n; i++)  
   {  
   for (j = 0; j < k; j++)  
   {  
   elem\_c = 0;  
   for (r = 0; r < n; r++)  
   {  
   elem\_c = elem\_c + a[i][r] \* b[r][j];  
   c[i][j] = elem\_c;  
   }  
   }  
   }  
     
   clock\_t t1 = clock();  
     
   printf("Произведение двух матриц размерами %d x %d заняло по времени %lf c:\n", n, k, (double)(t1 - t0) / CLK\_TCK);  
   return(0);  
   }
2. Сложность программы: О(n3)
3. Время выполнения:

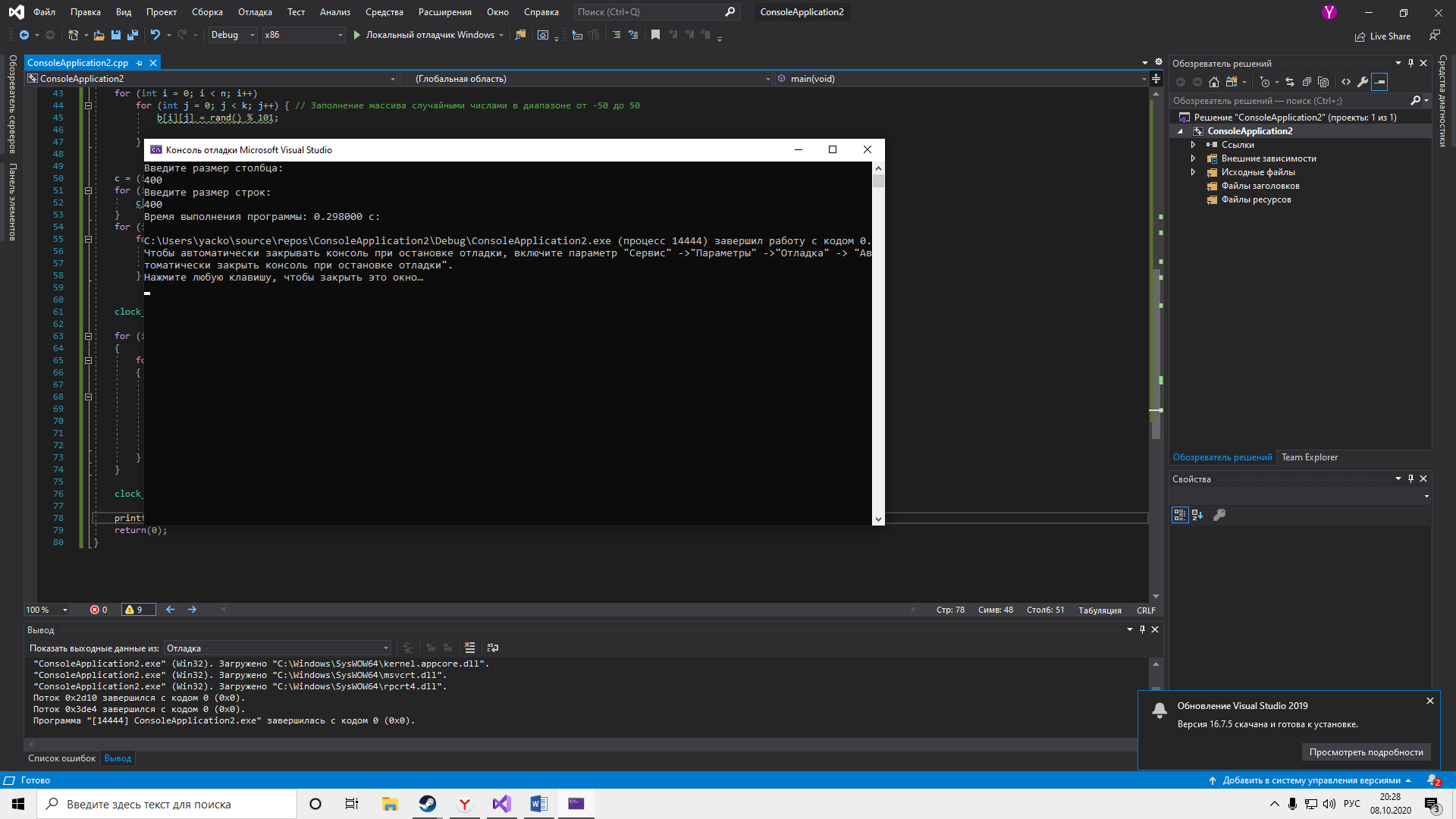
100x100:



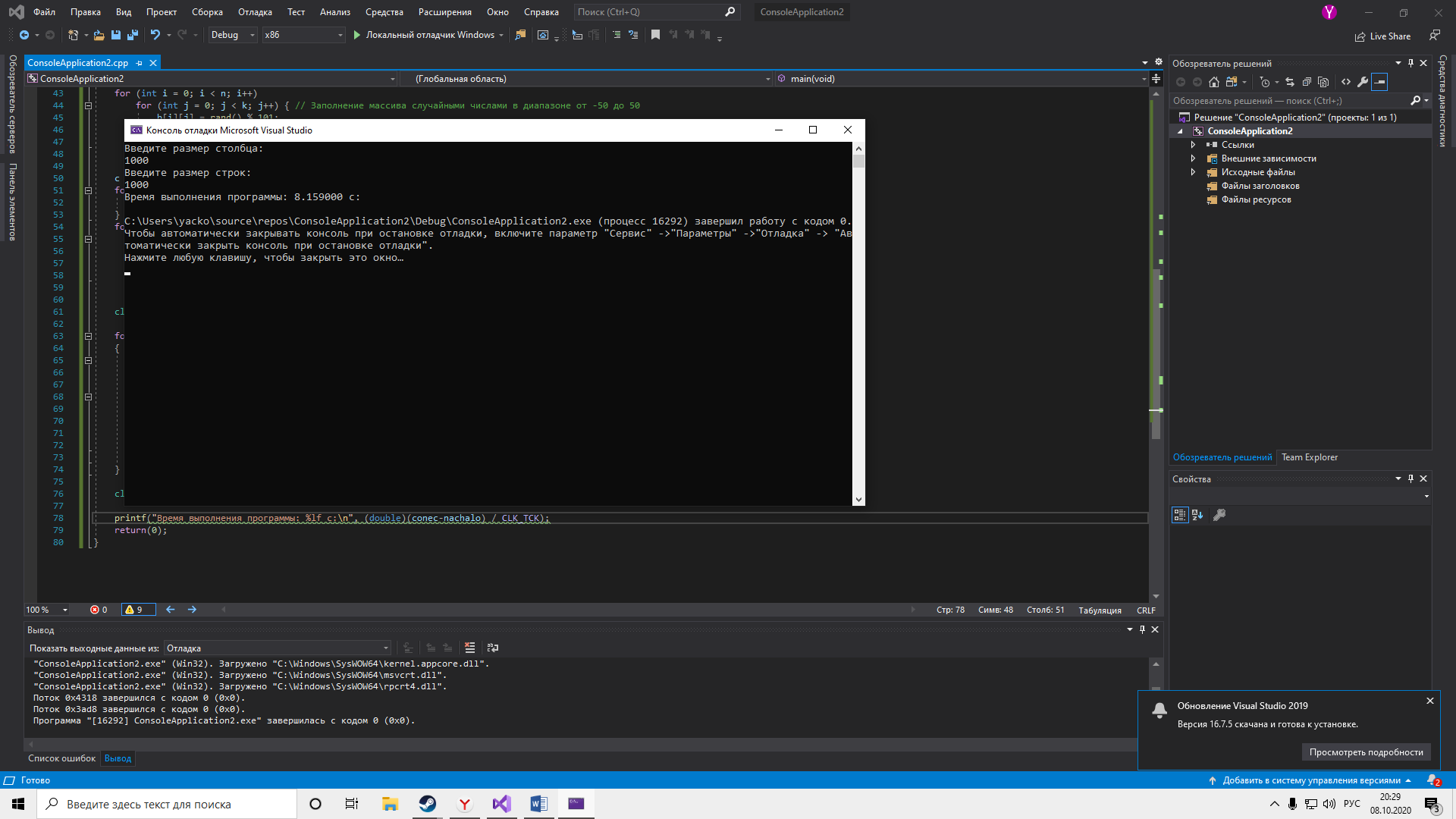
200х200



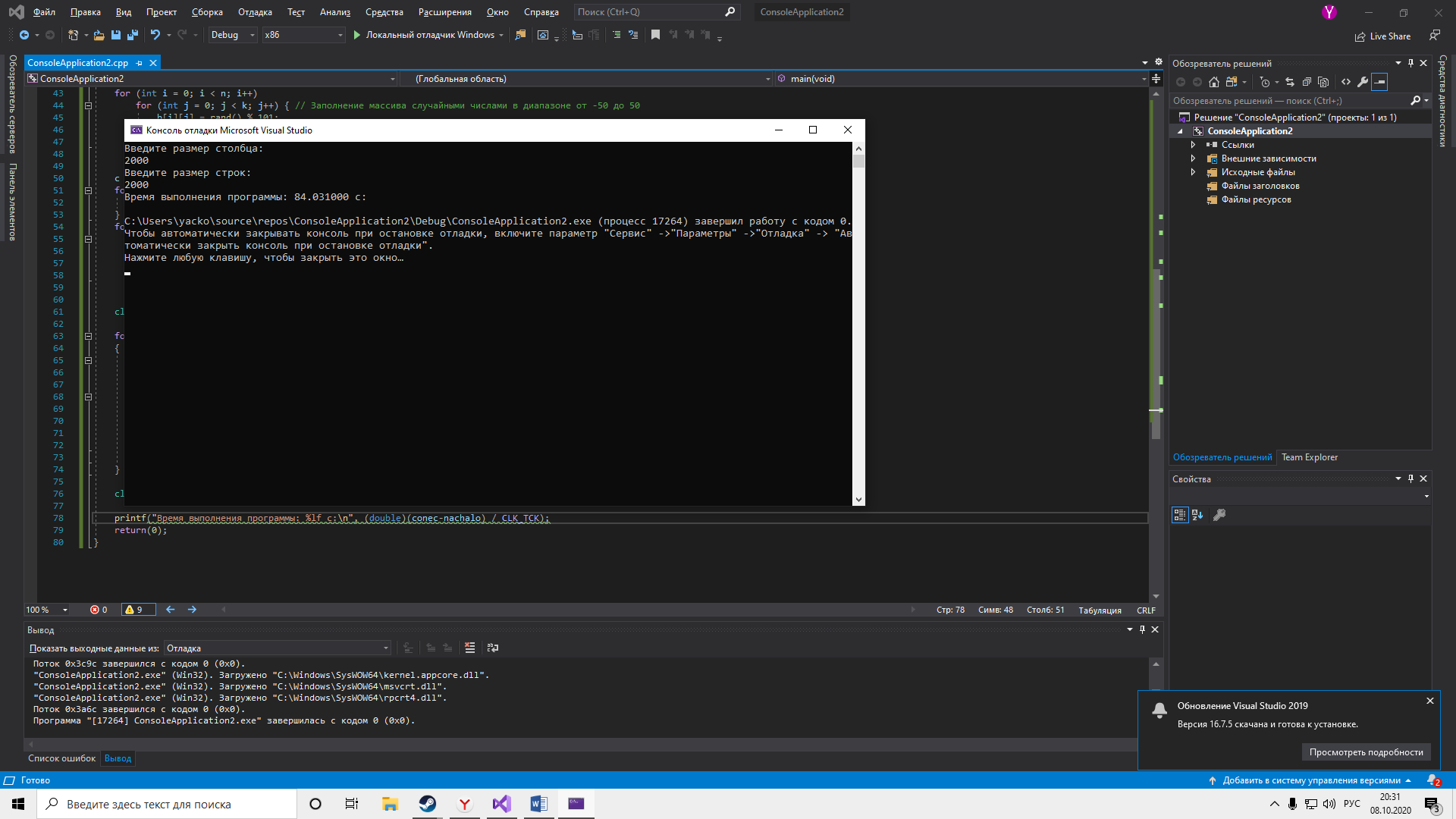
400х400



1000х1000

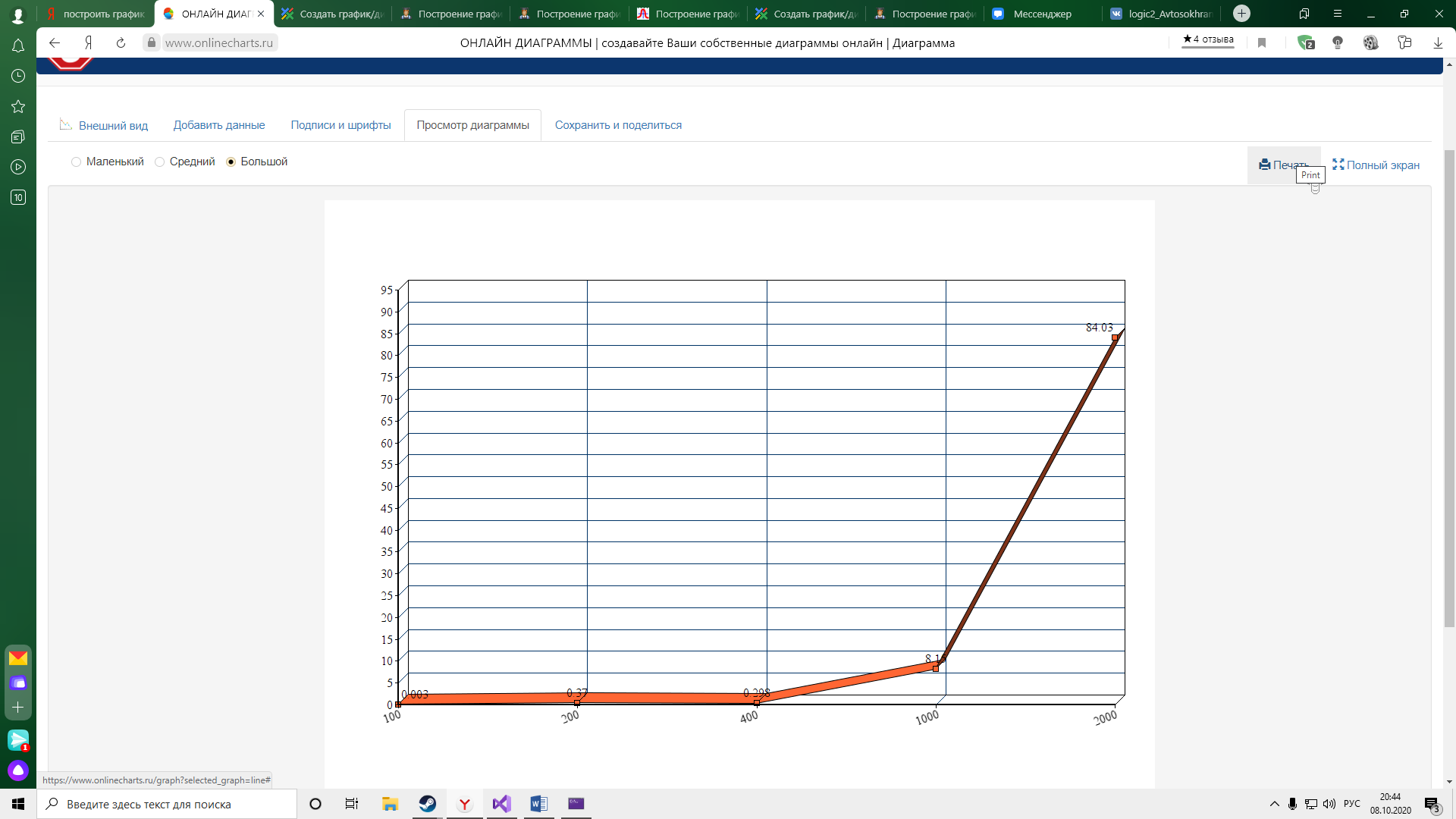


2000х2000



4000х4000

1. Построил график зависимости времени выполнения программы от размера матриц.



**Задание 2:**

**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

#include<locale.h>

#include<time.h>

#include<string.h>

void shell(int\* items, int count)

{

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

for (k = 0; k < 5; k++) {

gap = a[k];

for (i = gap; i<count; ++i) {

x = items[i];

for (j = i - gap; (x <items[j]) && (j >= 0); j = j - gap)

items[j + gap] = items[j];

items[j + gap] = x;

}

}

}

void qs(int\* items, int left, int right) //вызовфункции: qs(items, 0, count-1);

{

int i, j;

int x, y;

i = left; j = right;

/\* выбор компаранда \*/

x = items[(left + right) / 2];

do {

while ((items[i] < x) && (i<right)) i++;

while ((x <items[j]) && (j >left)) j--;

if (i<= j) {

y = items[i];

items[i] = items[j];

items[j] = y;

i++; j--;

}

} while (i<= j);

if (left< j) qs(items, left, j);

if (i<right) qs(items, i, right);

}

/\* сравнениедвухцелых \*/

int comp(const void\* i, const void\* j) {

return(int)i - (int)j;

}

void mas(int n, double\*\* ar, int b) {

//srand(time(NULL));

int\* a1, \* a2, \* a3, \* a4, \* a5;

int m = 0;

double t;

a1 = (int\*)malloc(n \* sizeof(int));

a2 = (int\*)malloc(n \* sizeof(int));

a3 = (int\*)malloc(n \* sizeof(int));

a4 = (int\*)malloc(n \* sizeof(int));

a5 = (int\*)malloc(n \* sizeof(int));

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

a1[i] = rand() % 100;

}

for (int i = 0; i <n; i++) { //возрастающая последовательность чисел

a2[i] = i;

}

for (int i = n; i> 0; i--) {//убывающаяпоследовательностьчисел

a3[i] = i;

}

for (int i = 0; i <n; i++) { //первая половина которого возроставет, а вторая убывает

if (i<n / 2) {

a4[i] = i;

//printf("%d\n", a4[i]);

}

else {

a4[i] = i - m;

m = m + 2;

//printf("%d\n", a4[i]);

}

}

//СортировкаШелла

memcpy(a5, a1, n);

time\_t start = clock();

shell(a5, n);

time\_t stop = clock();

t = (stop - start) / 1000.0;

ar[b - 3][0] = t;

printf("Время выполнения сортировки Шелла со случайным набором значений массива %fl\n", t);

//fprintf(f1, "Сортировка Шелла:\n");

//fprintf(f1, "1) %fl\n", t);

memcpy(a5, a2, n);

start = clock();

shell(a5, n);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 3][1] = t;

printf("Время выполнения сортировки Шелла c возрастающей последовательностью чисел %fl\n", t);

//fprintf(f1, "2) %fl\n", t);

memcpy(a5, a3, n);

start = clock();

shell(a5, n);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 3][2] = t;

printf("Время выполнения сортировки Шелла c убывающей последовательностью чисел %fl\n", t);

//fprintf(f1, "3) %fl\n", t);

memcpy(a5, a4, n);

start = clock();

shell(a5, n);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 3][3] = t;

printf("Время выполнения сортировки Шелла первая половина которого возроставет, а вторая убывает %fl\n", t);

//fprintf(f1, "4) %fl\n", t);

printf("\*\*\*\*\*\*\*\*\*\*\*\n");

// Быстрая сортировка

//fprintf(f1, "Быстрая сортировка:\n");

memcpy(a5, a1, n);

start = clock();

qs(a5, 0, n - 1);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 2][0] = t;

printf("Время выполнения быстрой сортировки со случайным набором значений массива %fl\n", t);

//fprintf(f1, "1) %fl\n", t);

memcpy(a5, a2, n);

start = clock();

qs(a5, 0, n - 1);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 2][1] = t;

printf("Время выполнения быстрой сортировки c возрастающей последовательностью чисел %fl\n", t);

//fprintf(f1, "2) %fl\n", t);

memcpy(a5, a3, n);

start = clock();

qs(a5, 0, n - 1);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 2][2] = t;

printf("Время выполнения быстрой сортировки c убывающей последовательностью чисел %fl\n", t);

//fprintf(f1, "3) %fl\n", t);

memcpy(a5, a4, n);

start = clock();

qs(a5, 0, n - 1);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 2][3] = t;

printf("Время выполнения быстрой сортировки первая половина которого возроставет, а вторая убывает %fl\n", t);

//fprintf(f1, "4) %fl\n", t);

//стандартной функции qsort

printf("\*\*\*\*\*\*\*\*\*\*\*\n");

//fprintf(f1, "Стандартной функции qsort\n");

memcpy(a5, a1, n);

start = clock();

qsort(a5, n, sizeof(int), comp);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 1][0] = t;

printf("Время выполнения стандартной функции qsort со случайным набором значений массива %fl\n", t);

//fprintf(f1, "1) %fl\n", t);

memcpy(a5, a2, n);

start = clock();

qsort(a5, n, sizeof(int), comp);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 1][1] = t;

printf("Время выполнения стандартной функции qsort c возрастающей последовательностью чисел %fl\n", t);

//fprintf(f1, "2) %fl\n", t);

memcpy(a5, a3, n);

start = clock();

qsort(a5, n, sizeof(int), comp);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 1][2] = t;

printf("Время выполнения стандартной функции qsort c убывающей последовательностью чисел %fl\n", t);

//fprintf(f1, "3) %fl\n", t);

memcpy(a5, a4, n);

start = clock();

qsort(a5, n, sizeof(int), comp);

stop = clock();

t = (stop - start) / 1000.0;

ar[b - 1][3] = t;

printf("Время выполнения стандартной функции qsort первая половина которого возроставет, а вторая убывает %fl\n", t);

//fprintf(f1, "4) %fl\n", t);

}

int main() {

system("chcp 1251");

system("cls");

FILE\* f1;

int n = 0, arr, m = 4, c, b = 0, z = 0, j = 0, g = 0;

double\*\* a;

double\*\* bb;

int\* aa;

double v;

char f[] = "REZ.txt";

f1 = fopen(f, "w");

printf("Кол-во:");

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

c = arr \* 3;

g = 12;

a = (double\*\*)malloc(c \* sizeof(double\*));

bb = (double\*\*)malloc(g \* sizeof(double\*));

aa = (int\*)malloc(arr \* sizeof(int\*));

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

a[i] = (double\*)malloc(m \* sizeof(double));

}

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

bb[i] = (double\*)malloc(arr \* sizeof(double));

}

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

b = b + 3;

printf("Введите размер массива\n");

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

aa[i] = n;

mas(n, a, b);

}

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

fprintf(f1, " %d", aa[i]);

}

fprintf(f1, "%\n");

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

for (int k = 0; k < g; k++) {

bb[k][i] = a[z][j];

if (j < m) {

j++;

}

else {

j = 0;

if (z != c - 1) {

z++;

}

}

if (j == 4) {

j = 0;

if (z != c - 1) {

z++;

}

}

}

}

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

for (int k = 0; k <arr; k++) {

fprintf(f1, " %lf", bb[i][k]);

}

fprintf(f1, "%\n");

}

}

