Пензенский государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №2

По дисциплине: «[Логика и основы алгоритмизации в инженерных задачах](http://moodle.pnzgu.ru/course/view.php?id=48560)»

на тему «Оценка времени выполнения программы»

*Выполнил студенты группы 21ВВ2:*

*Зайкин С.А*

*Приняли:*

*Митрохин М.А.*

*Юрова О.А.*

Пенза 2022

**Цель работы:**

Оценить времени выполнения программы.

**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.

1) Порядок сложности (O-*С*имволика):

#include"stdafx.h"

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<locale>

int main(void)

{

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

setlocale (LC\_ALL,"Rus");

system("cls");

clock\_t start, end; // объявляем переменные для определения времени выполнения

int i=0, j=0, r;

int m,u;

int\*\* A;

int\*\* B;

printf("Введите размер: ");

scanf("%d",&m);

A = (int\*\*)malloc(m \* sizeof(int));

B = (int\*\*)malloc(m \* sizeof(int));

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

{

A[i]=(int\*)malloc(m \* sizeof(int));

B[i]=(int\*)malloc(m \* sizeof(int));

//Сложность О (n^2):

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

{

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

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

}

}

int\*\* C;

C = (int\*\*)malloc(m \* sizeof(int));

unsigned int start\_time = clock();

//Сложность О (n^3):

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

{

C[i]=(int\*)malloc(m \* sizeof(int));

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

{

u=0;

for(r=0;r<m;r++)

{

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

C[i][j]=u;

}

}

}

unsigned int end\_time = clock();

double search\_time = (end\_time - start\_time)/1000.0;

printf("%lf\n\n",search\_time);

system("pause");

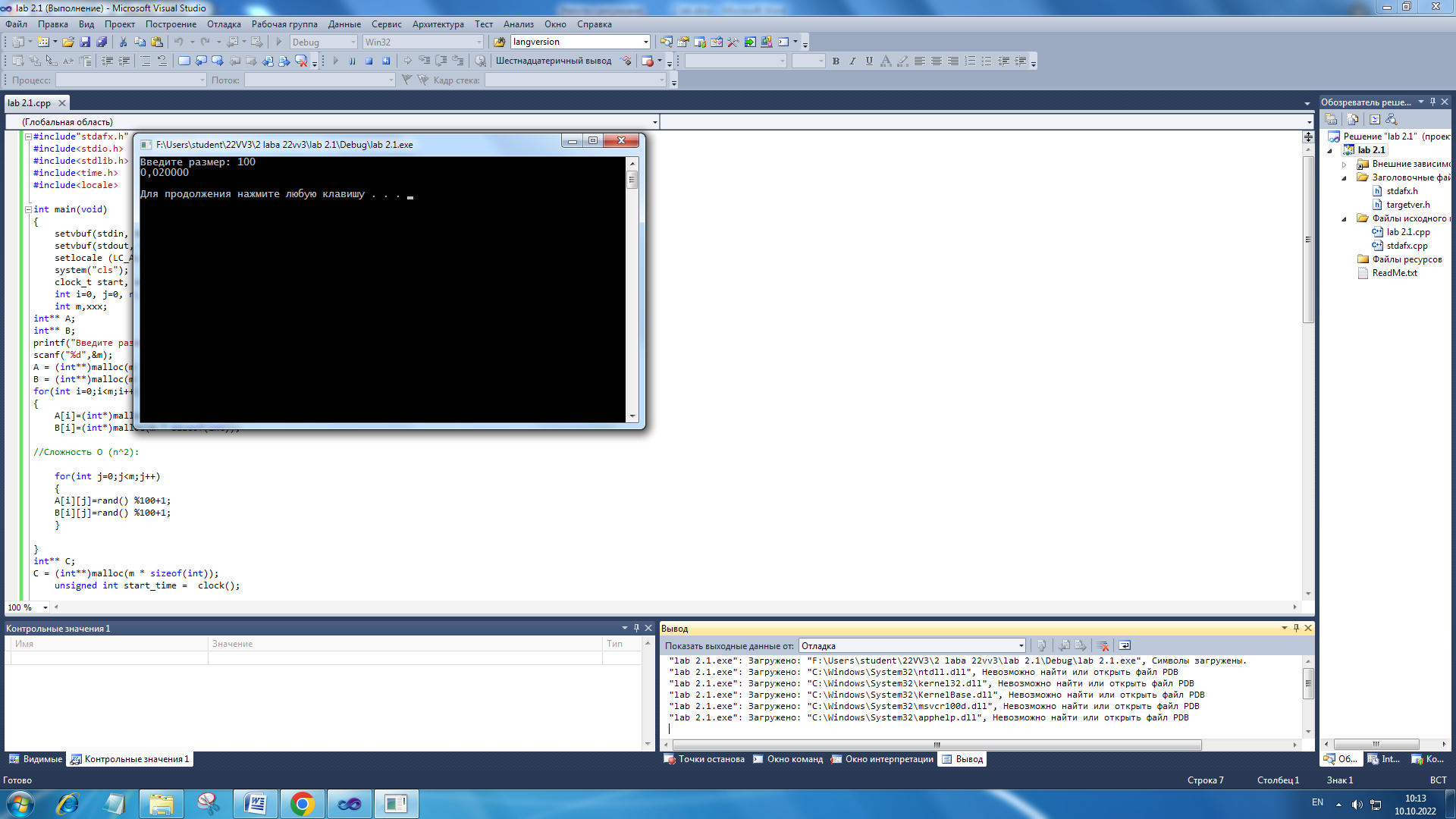
}

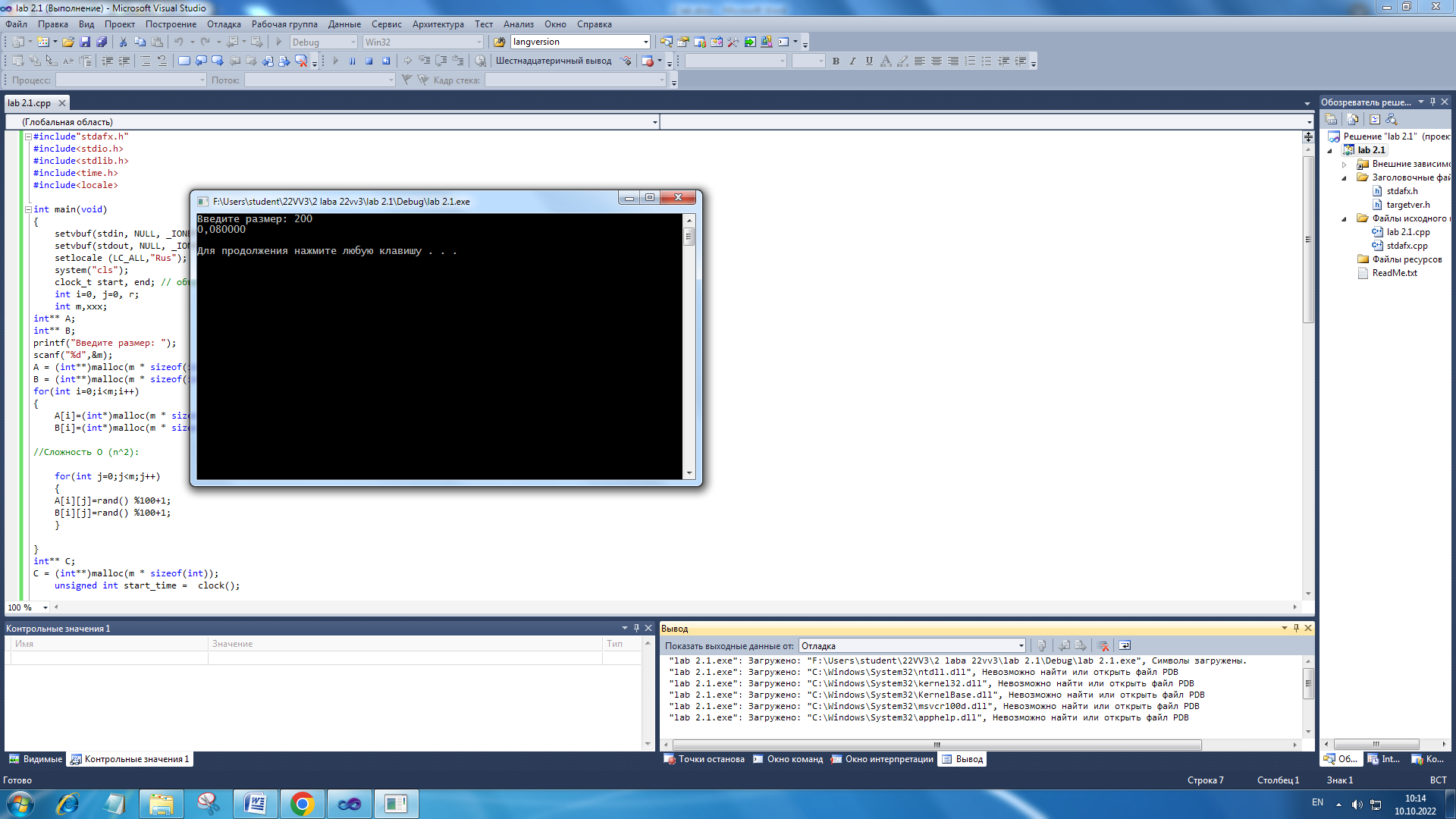
Порядок сложности программы n3

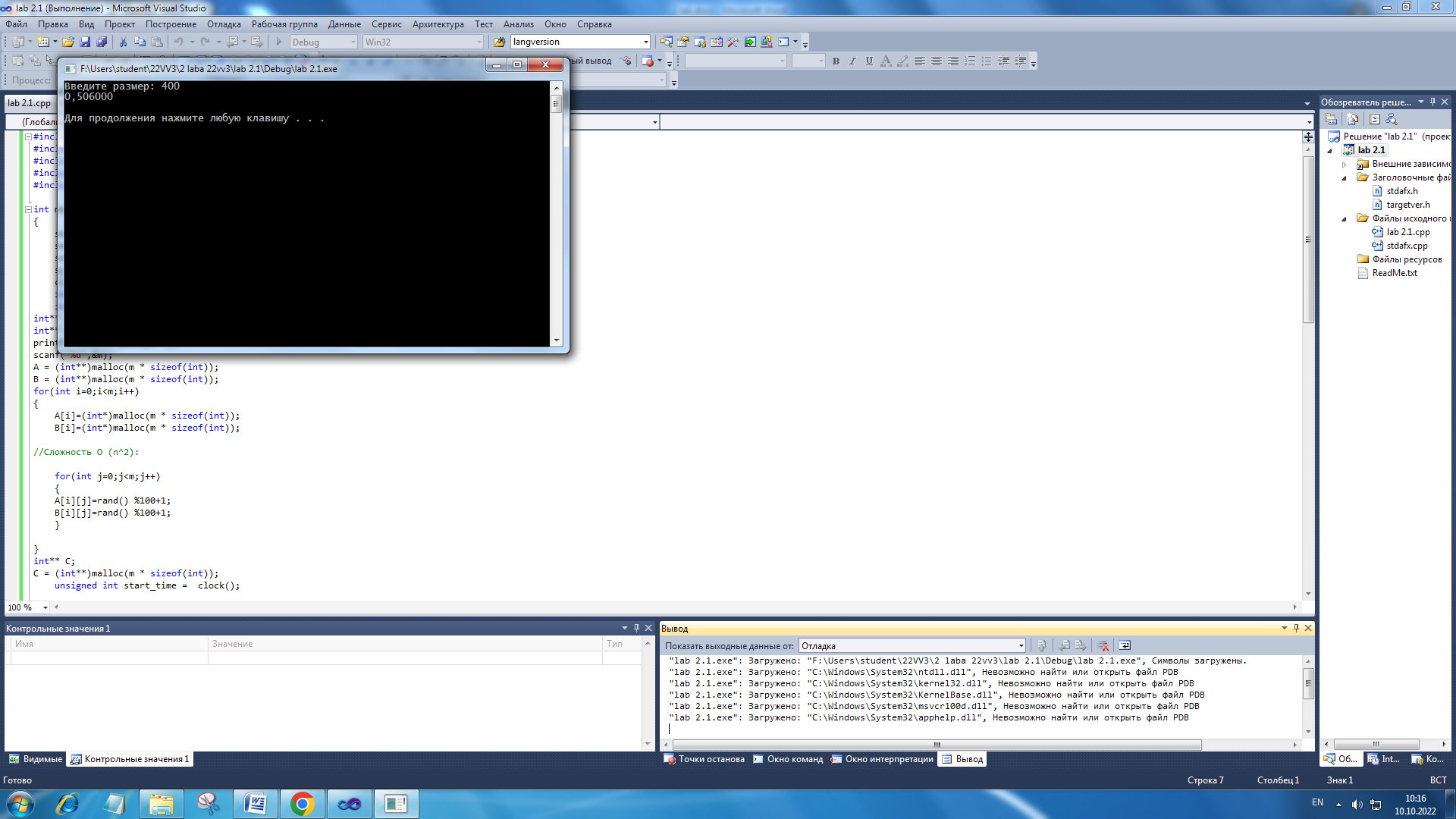
2) Оценить время выполнения программы и кода, выполняющего перемножение

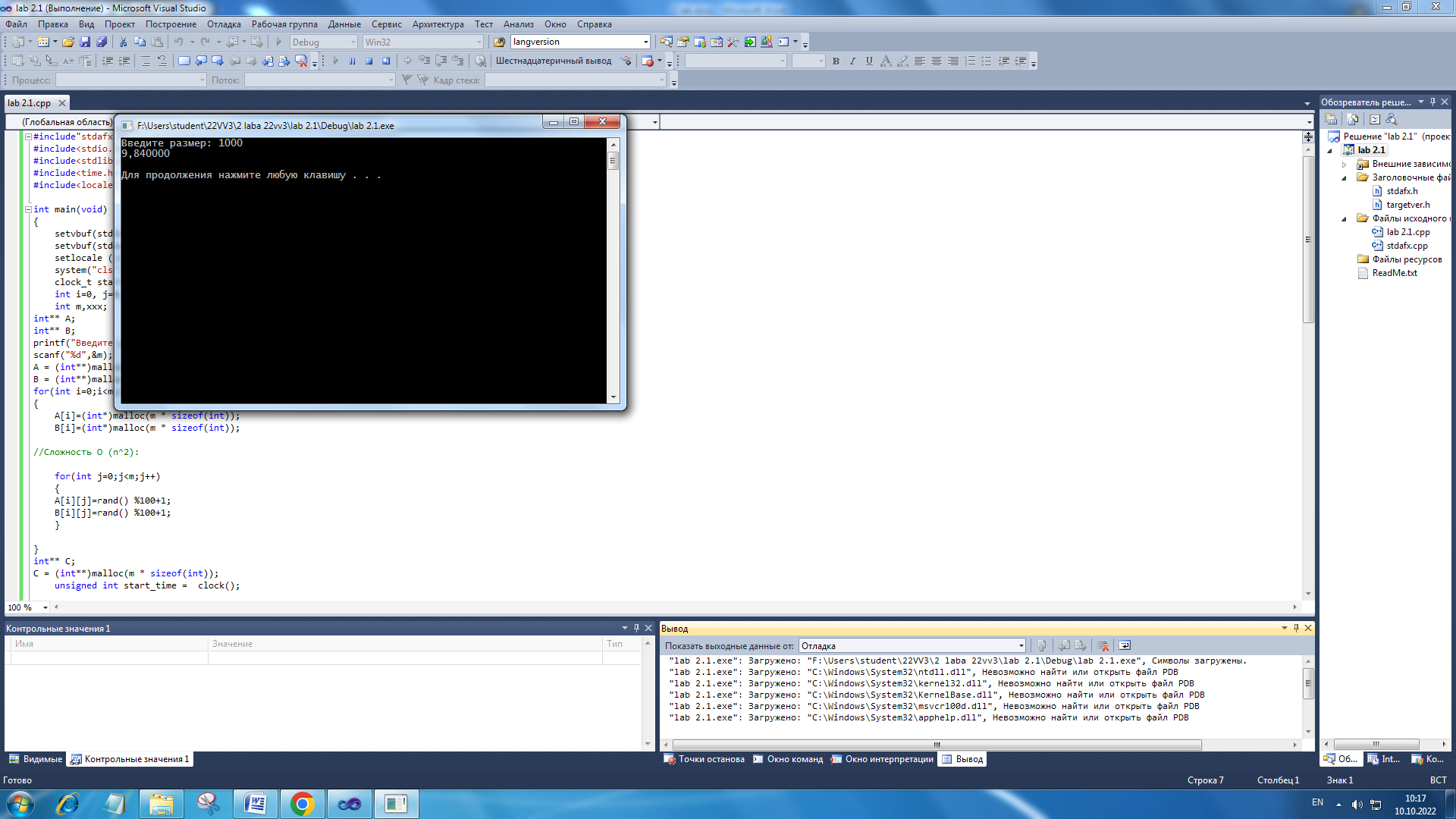
матриц, используя функции библиотеки time.h для матриц размерами от 100 , 200, 400,

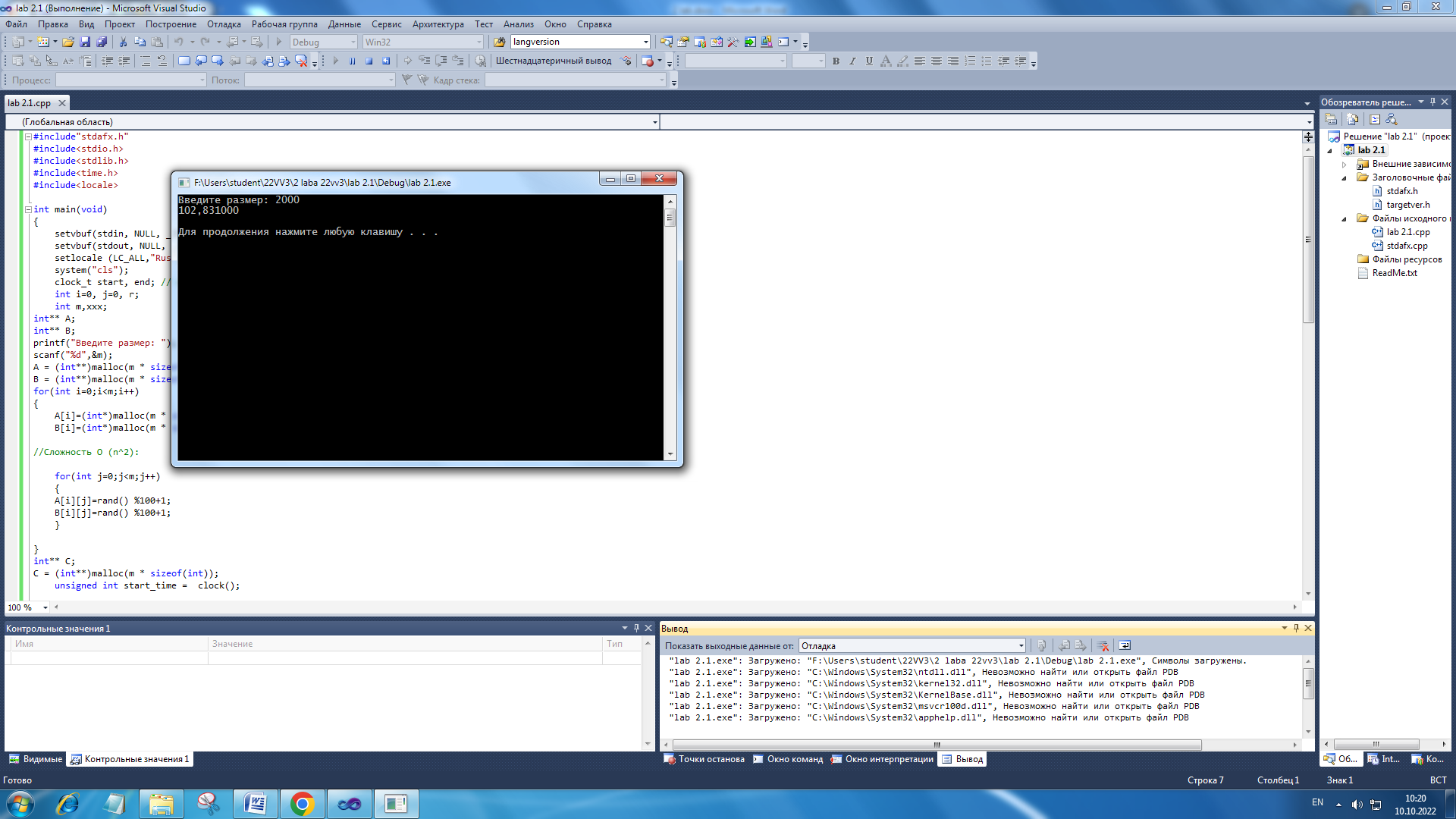
1000, 2000, 4000.

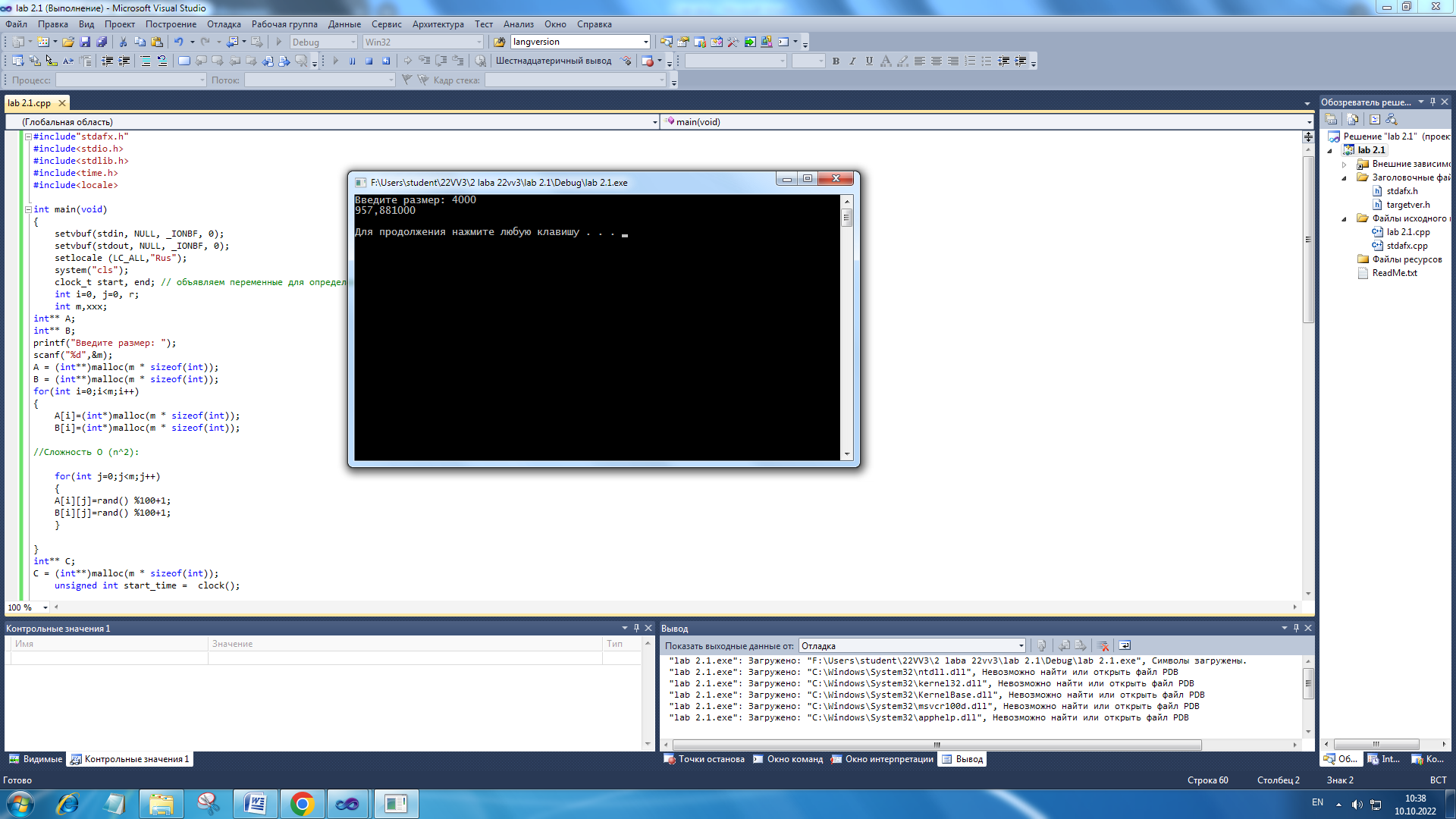




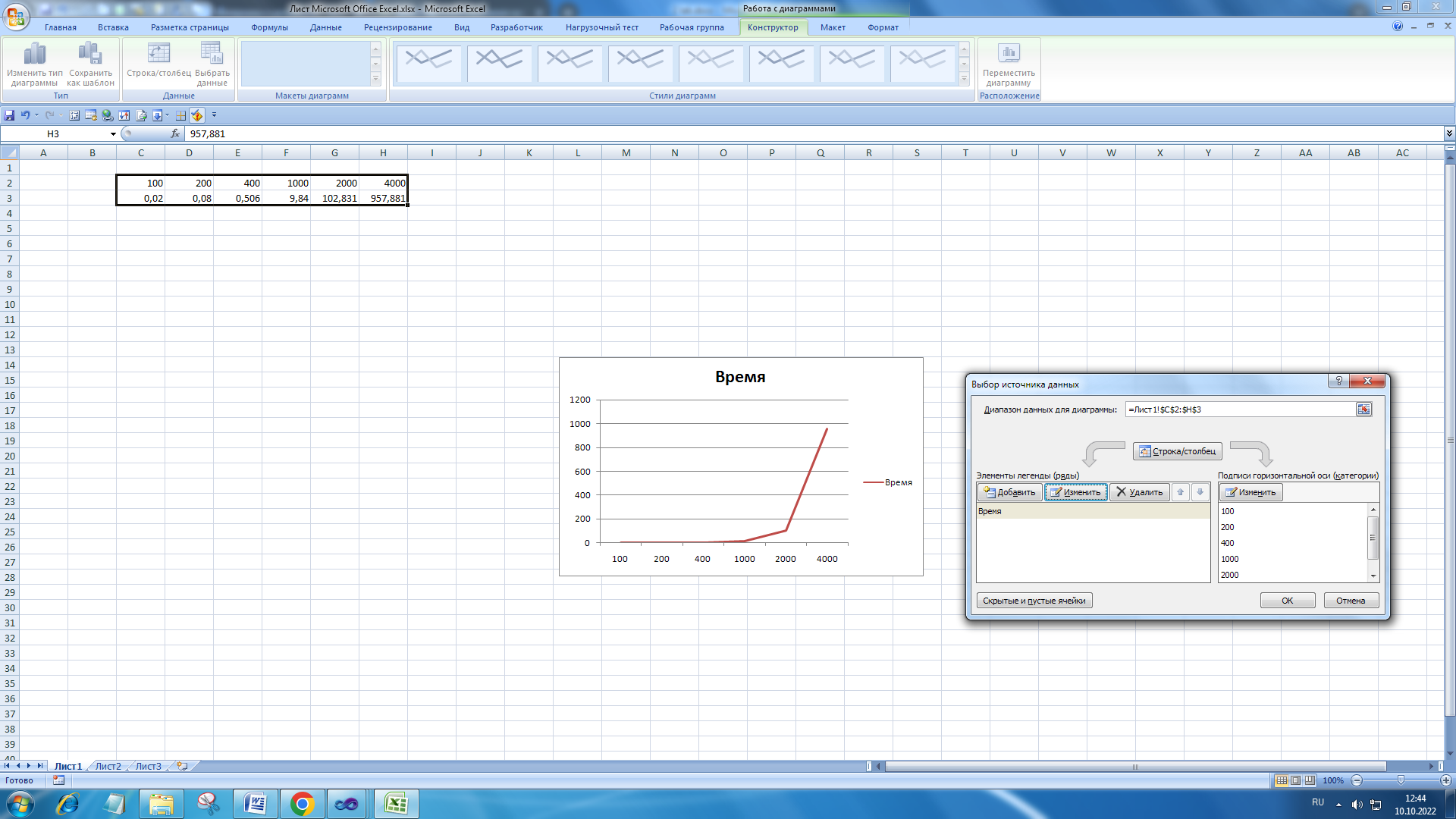






**

3) Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

**

*Задание 2:*

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void qs(int\* items, int left, int right)

{

int i, j;

int x, y;

i = left; j = right;

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);

}

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;

}

}

}

int compare(const void \* x1, const void \* x2) // функция сравнения элементов массива

{

return ( \*(int\*)x1 - \*(int\*)x2 ); // если результат вычитания равен 0, то числа равны, < 0: x1 < x2; > 0: x1 > x2

}

int main(void)

{

srand(time(NULL));

setvbuf(stdin, NULL, \_IONBF, 0);

setvbuf(stdout, NULL, \_IONBF, 0);

system("chcp 1251");

system("cls");

clock\_t start, end; // объявляем переменные для определения времени выполнения

clock\_t start1, end1;

clock\_t start2, end2;

clock\_t start3, end3;

clock\_t start4, end4;

clock\_t start5, end5;

clock\_t start6, end6;

clock\_t start7, end7;

clock\_t start8, end8;

clock\_t start9, end9;

clock\_t start10, end10;

clock\_t start11, end11;

clock\_t start12, end12;

int a[42000], b[42000], c[42000], n = 42000, i;

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

{

a[i] = rand() % 800;

b[i] = a[i];

c[i] = a[i];

}

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

unsigned int start\_time = clock();

shell(a, n);

unsigned int end\_time = clock();

double search\_time = (end\_time - start\_time) / 1000.0;

unsigned int start4\_time = clock();

qs(b, 0, n - 1);

unsigned int end4\_time = clock();

double search4\_time = (end4\_time - start4\_time) / 1000.0;

unsigned int start9\_time = clock();

qsort(c, n, sizeof(int), compare);

unsigned int end9\_time = clock();

double search9\_time = (end9\_time - start9\_time) / 1000.0;

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

{

a[i] = i + 1;

b[i] = a[i];

c[i] = a[i];

}

unsigned int start1\_time = clock();

shell(a, n);

unsigned int end1\_time = clock();

double search1\_time = (end1\_time - start1\_time) / 1000.0;

unsigned int start5\_time = clock();

qs(b, 0, n - 1);

unsigned int end5\_time = clock();

double search5\_time = (end5\_time - start5\_time) / 1000.0;

unsigned int start10\_time = clock();

qsort(c, n, sizeof(int), compare);

unsigned int end10\_time = clock();

double search10\_time = (end10\_time - start10\_time) / 1000.0;

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

{

a[i] = 42000 - i;

b[i] = a[i];

c[i] = a[i];

}

unsigned int start2\_time = clock();

shell(a, n);

unsigned int end2\_time = clock();

double search2\_time = (end2\_time - start2\_time) / 1000.0;

unsigned int start6\_time = clock();

qs(b, 0, n - 1);

unsigned int end6\_time = clock();

double search6\_time = (end6\_time - start6\_time) / 1000.0;

unsigned int start11\_time = clock();

qsort(c, n, sizeof(int), compare);

unsigned int end11\_time = clock();

double search11\_time = (end11\_time - start11\_time) / 1000.0;

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

{

if (i < 21000) { a[i] = i; }

if (i >= 21000) { a[i] = a[i - 1] - 1; }

b[i] = a[i];

c[i] = a[i];

}

unsigned int start8\_time = clock();

shell(a, n);

unsigned int end8\_time = clock();

double search8\_time = (end8\_time - start8\_time) / 1000.0;

unsigned int start7\_time = clock();

qs(b, 0, n - 1);

unsigned int end7\_time = clock();

double search7\_time = (end7\_time - start7\_time) / 1000.0;

unsigned int start12\_time = clock();

qsort(c, n, sizeof(int), compare);

unsigned int end12\_time = clock();

double search12\_time = (end12\_time - start12\_time) / 1000.0;

printf("|-----------------|-----------------|--------------|--------------|\n");

printf("| ранд | возраст. | убыв. | труег. |\n");

printf("|-----------------|-----------------|--------------|--------------|\n");

printf("|shel | %f | %f | %f | %f |\n", search\_time, search1\_time, search2\_time, search8\_time);

printf("| qs | %f | %f | %f | %f |\n", search4\_time, search5\_time, search6\_time, search7\_time);

printf("|qsort| %f | %f | %f | %f |\n", search9\_time, search10\_time, search11\_time, search12\_time);

printf("|-----------------|-----------------|--------------|--------------|\n");

system("pause");

}

1. Оценил время работы каждого из реализованных алгоритмов на случайном наборе значений массива.

Время выполнения программы Быстрой сортировкой - 0,011s

Время выполнения программы сортировкой Шелли - 1,002s

Время выполнения программы Быстрой сортировкой из библиотеки - 0,039s

2. Оценил время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.

Время выполнения программы Быстрой сортировкой - 0,007s

Время выполнения программы сортировкой Шелли - 0,003s

Время выполнения программы Быстрой сортировкой из библиотеки - 0,026s

3. Оценил время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.

Время выполнения программы Быстрой сортировкой - 0,06s

Время выполнения программы сортировкой Шелли - 1,968s

Время выполнения программы Быстрой сортировкой из библиотеки - 0,026s

4. Оценил время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

Время выполнения программы Быстрой сортировкой - 2,373000s

Время выполнения программы сортировкой Шелли - 0,977s

Время выполнения программы Быстрой сортировкой из библиотеки - 0,025s

**Вывод:** в ходе данной работы научился оценивать время работы программы.