МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительной техники»

**ОТЧЁТ**

по лабораторной работе №2

по курсу «Л и ОА в ИЗ»

на тему «Оценка времени выполнения программ»

Выполнили:

Студенты группы 22ВВС1

Костин Максим

Макеева Дарья

Приняли:

Акифьев И.В.

Юрова О.В.

Пенза 2023

***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.

**Листинг:**

**Задание 1:**

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

int n = 0;

this->chart1->Series[0]->Points->Clear();

while (n <= 6) {

//100, 200, 400, 1000, 2000, 4000, 10000

int ci[7] = { 1,2,4,10,20,40 , 100 };

int d = 0;

d = ci[n];

double t = multi(d \* 100);

printf("matrica n\*n %d , time %lf \n", d \* 100, t);

this->chart1->Series[0]->Points->AddXY(d \* 100, t);

n += 1;

}

return System::Void();

}

double Laba2::MyForm::multi(int size)

{

srand(time(NULL));

clock\_t start, end;

int r = 0;

int i = 0;

int j = 0;

int s = 0;

double elem\_c;

int\*\* a = (int\*\*)malloc(sizeof(int\*) \* size);

int\*\* b = (int\*\*)malloc(sizeof(int\*) \* size);

int\*\* c = (int\*\*)malloc(sizeof(int\*) \* size);

while (size > i) {

j = 0;

b[i] = (int\*)malloc(sizeof(int) \* size);

a[i] = (int\*)malloc(sizeof(int) \* size);

c[i] = (int\*)malloc(sizeof(int) \* size);

while (size > j) {

a[i][j] = rand() % 1000;

b[i][j] = rand() % 1000;

c[i][j] = rand() % 1000;

j++;

}

i++;

}

start = clock();

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 = clock();

free(a);

free(b);

free(c);

while (size > i) {

free(a[i]);

free(b[i]);

free(c[i]);

i++;

}

double diff = (end - start);

double t = diff / CLOCKS\_PER\_SEC;

return(t);

}

**Задание 2:**

# #include <stdlib.h>

# #include <stdio.h>

# #include <time.h>

# #include <conio.h>

# #include <locale.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) {

# int i, j, 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);

# }

# int compare(const void\* x1, const void\* x2) {

# return(\*(int\*)x1 - \*(int\*)x2);

# }

# int main() {

# time\_t end, start;

# srand(time(NULL));

# setlocale(LC\_ALL, "Rus");

# int n = 0;

# printf("Введите количество эл массива:");

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

# int\* c = (int\*)calloc(sizeof(int), n);

# int\* d = (int\*)calloc(sizeof(int), n);

# int\* k = (int\*)calloc(sizeof(int), n);

# printf("\nВремя сортировки : \n");

# printf("Массив случайных значений:\n");

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

# c[i] = rand() % 100 - 50;

# d[i] = rand() % 100 - 50;

# k[i] = rand() % 100 - 50;

# }

# start = clock();

# qsort(k, n, sizeof(int), compare);

# end = clock();

# double def = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм стандартной сортировки:%lf\n", def);

# start = clock();

# shell(c, n);

# end = clock();

# double dif = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм Шелла:%lf\n", dif);

# start = clock();

# qs(d, 0, (n - 1));

# end = clock();

# double dif1 = (double)(end - start) / (CLOCKS\_PER\_SEC);

# printf("Алгоритм быстрой сортировки:%lf\n", dif1);

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

# printf("Массив возрастающий:\n");

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

# k[i] = i + 21;

# }

# start = clock();

# qsort(k, n, sizeof(int), compare);

# end = clock();

# double def1 = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм стандартной сортировки:%lf\n", def1);

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

# c[i] = i + 21;

# }

# start = clock();

# shell(c, n);

# end = clock();

# double dif2 = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм Шелла:%lf\n", dif2);

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

# d[i] = i + 21;

# }

# start = clock();

# qs(d, 0, (n - 1));

# end = clock();

# double dif3 = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм быстрой сортировки:%lf\n", dif3);

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

# printf("Массив убывающий:\n");

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

# k[i] = n - i;

# }

# start = clock();

# qsort(k, n, sizeof(int), compare);

# end = clock();

# double def2 = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм стандартной сортировки:%lf\n", def2);

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

# c[i] = n - i;

# }

# start = clock();

# shell(c, n);

# end = clock();

# double dift = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм Шелла:%lf\n", dift);

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

# d[i] = n - i;

# }

# start = clock();

# qs(d, 0, (n - 1));

# end = clock();

# double dift1 = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм быстрой сортировки:%lf\n", dift1);

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

# printf("Массив на половину возрастающий на половину убывающий:\n");

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

# if (i < (n / 2))

# k[i] = i + 21;

# else

# k[i] = i - 12;

# }

# start = clock();

# qsort(k, n, sizeof(int), compare);

# end = clock();

# double def3 = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм стандартной сортировки:%lf\n", def3);

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

# if (i < (n / 2))

# c[i] = i + 21;

# else

# c[i] = i - 12;

# }

# start = clock();

# shell(c, n);

# end = clock();

# double dift2 = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм Шелла:%lf\n", dift2);

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

# if (i < (n / 2))

# d[i] = i + 21;

# else

# d[i] = i - 12;

# }

# start = clock();

# qs(d, 0, (n - 1));

# end = clock();

# double dift3 = (double)(end - start) / CLOCKS\_PER\_SEC;

# printf("Алгоритм быстрой сортировки:%lf\n", dift3);

# free(c);

# free(d);

# free(k);

# \_getch();

# }Пояснительный текст к программе:

**Задание 1 и 2:**

Кнопка при нажатии которой генерируются матрицы различного размера и перемножаются.

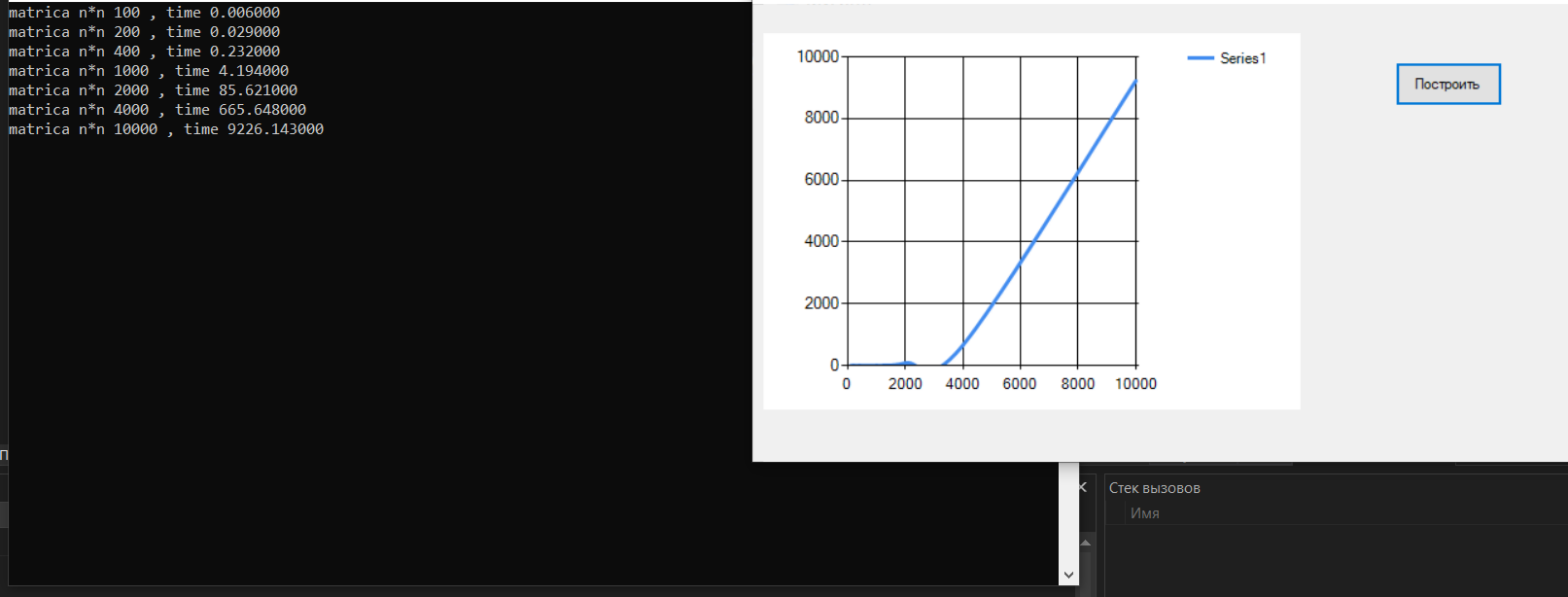
Clock() – возвращает время, измеряемое процессором в тактах от начала выполнения программы, или −1, если оно не известно. Пересчет этого времени в секунды выполняется по формуле:

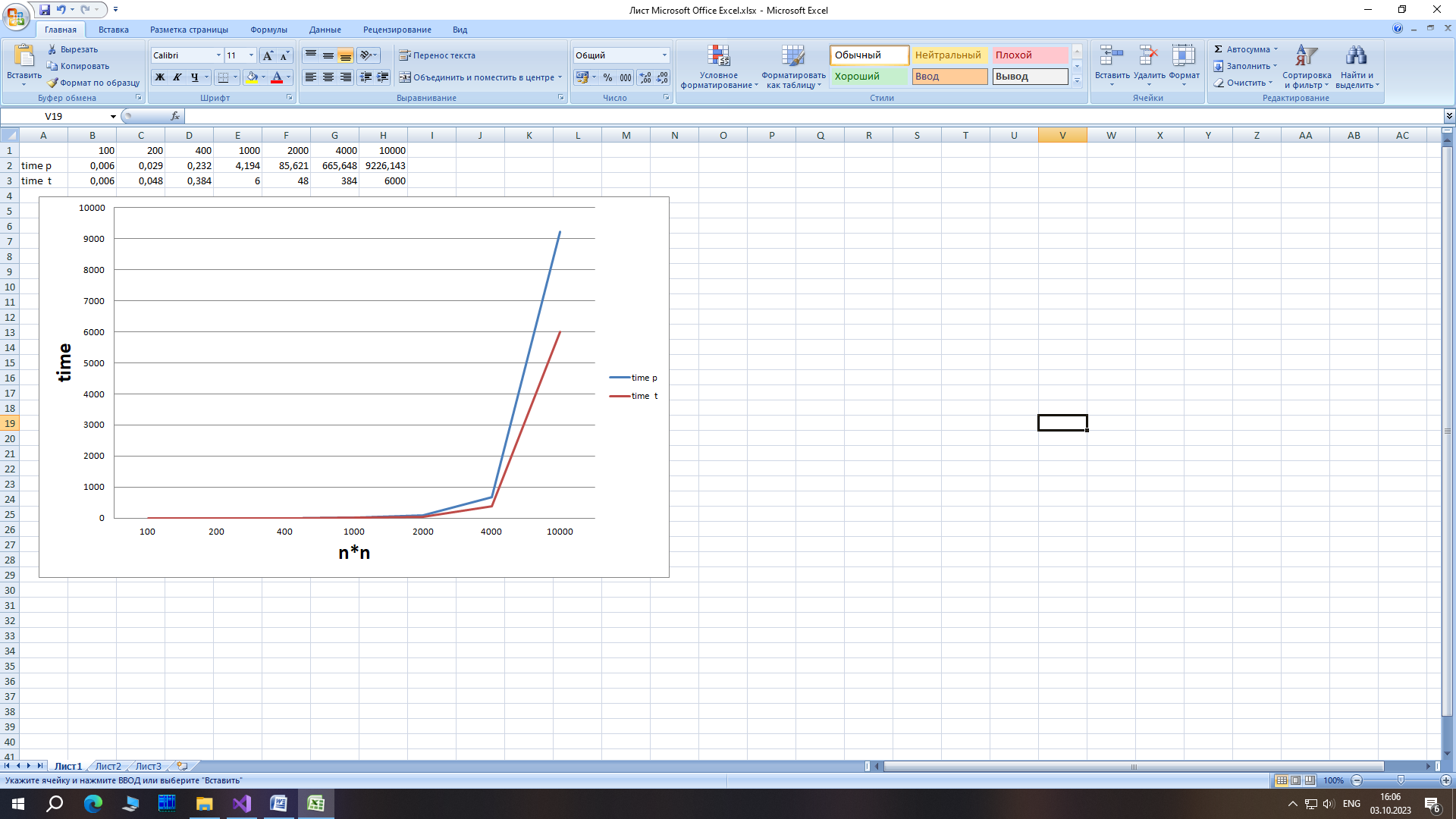
clock() / CLOCKS\_PER\_SEC

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

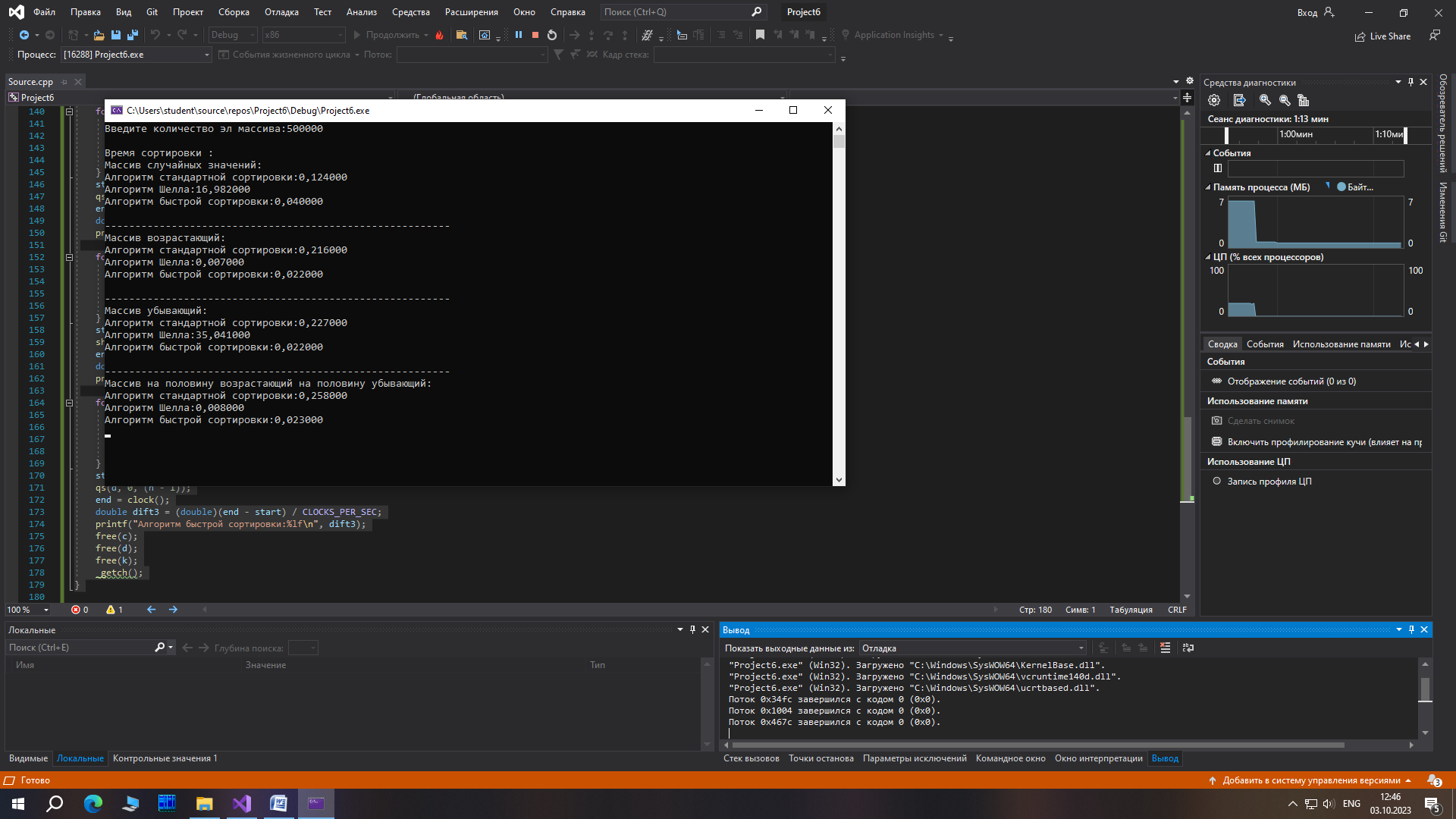
**Результат работы программы**

**Задание 1:**

****

****

**Задание 2:**

****

**Вывод**

В ходе выполнения лабораторной работы были разработаны программы для измерения времени и мы пришли к следующим выводам:

1. После того как определили сложность программы – О(n3), произвели расчет времени теоретически и практически и получили примерно одинаковый результат с определенной погрешностью.
2. Далее после рассмотрения нескольких алгоритмов сортировки на различных массивах, мы определили, что для массива случайных чисел более резонно использовать алгоритм стандартной сортировки языка си. Алгоритм Шелла больше подходит для сортировки возрастающего массива. Алгоритм быстрой сортировки практически во всех случаях показывает относительно одинаковое время.