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:**

Тебнев Р. В.

**Приняли:**

д.т.н. профессор Митрохин М. А.

Юрова О.В.

Пенза 2020

**Цель работы:** оценить время работы программ.

**Задание 1**

**Листинг**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

#include <locale.h>

int i=0, j=0, r;

const int N = 4000;

int a[N][N], b[N][N], c[N][N], elem\_c;

int main(void)

{

FILE\* fp;

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

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

setlocale(LC\_ALL, "rus");

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

while(i<N)

{

while(j<N)

{

a[i][j]=rand()% 100 + 1; // заполняем массив случайными числами

j++;

}

i++;

}

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

i=0; j=0;

while(i<N)

{

while(j<N)

{

b[i][j]=rand()% 100 + 1; // заполняем массив случайными числами

j++;

}

i++;

}

double start = clock();

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

{

for(j=0;j<N;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;

}

}

}

double end = clock();

fp = fopen("lab2.txt", "a");

printf("Количество элементов: %d\n", N);

fprintf(fp, "Количество элементов: %d\n", N);

printf("Время выполнения программы: %f\n", (end-start) / CLOCKS\_PER\_SEC);

fprintf(fp, "Время выполнения программы: %f\n", (end-start) / CLOCKS\_PER\_SEC);

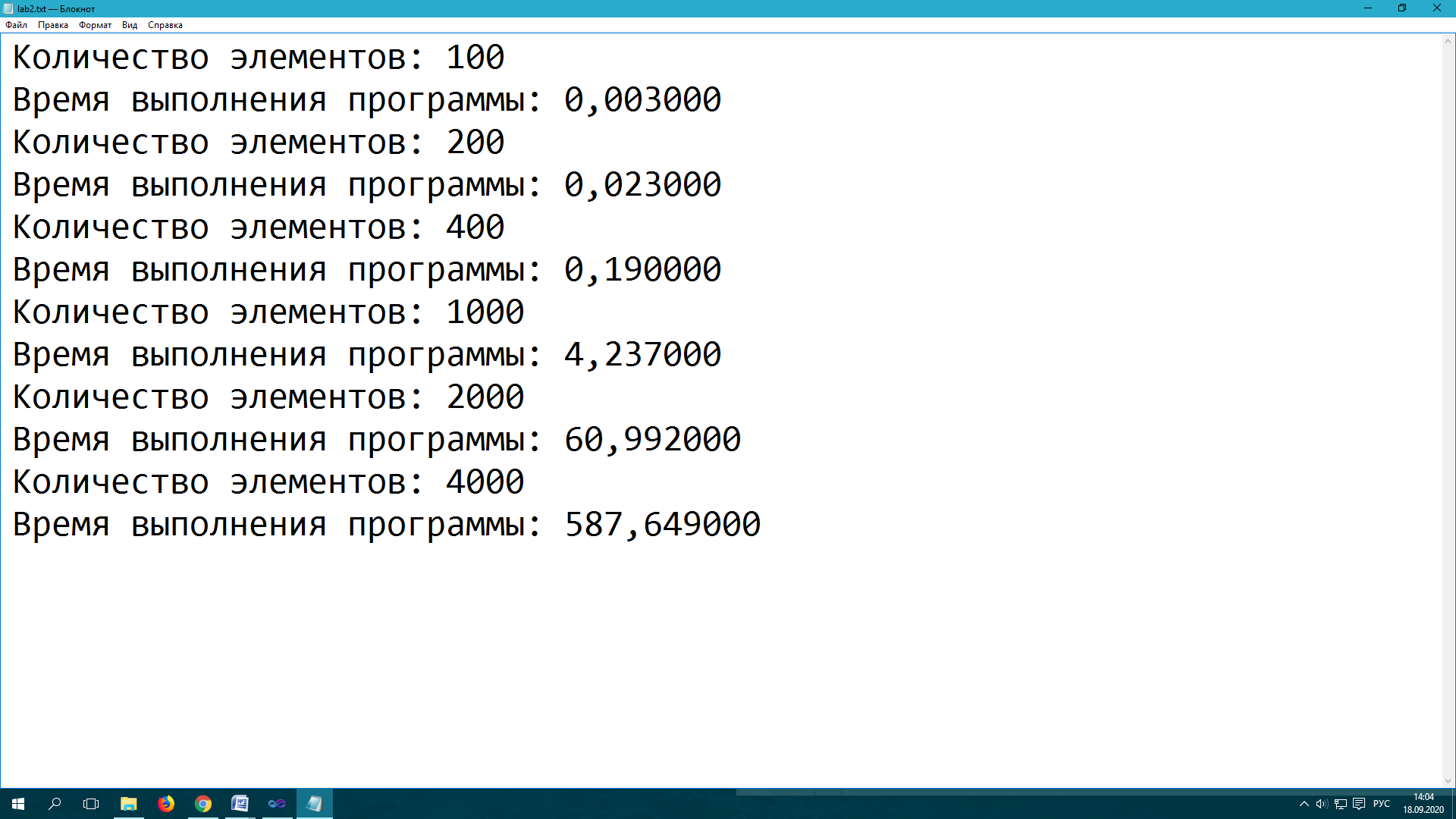
fclose(fp);

getch();

return(0);

}

1. Вычислить порядок сложности программы  
   Сложность программы – О(n3).
2. Оценить время выполнения (в секундах) программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.



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

**Задание 2**

**Листинг**#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <time.h>

#include <locale.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

int mas[15000];

FILE\* fp;

double start, end;

void qs(int\* items, int left, int right) //вызов функции: 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);

}

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 cmp(const void\* a, const void\* b) { //функция сравнения элементов массива

{

return \*(int\*)a - \*(int\*)b; // Если результат вычитания равен 0, то числа равны

}

}

void sort(int\* items) {

printf("\nСортировка Шелла:\t");

fprintf(fp, "\nСортировка Шелла:\t");

start = clock();

shell(items, 15000);

end = clock();

printf("%20lf", (end - start) / CLOCKS\_PER\_SEC);

fprintf(fp, "%20lf", (end - start) / CLOCKS\_PER\_SEC);

printf("\nБыстрая сортировка:\t");

fprintf(fp, "\nБыстрая сортировка:\t");

start = clock();

qs(items, 0, 15000 - 1);

end = clock();

printf("%20lf", (end - start) / CLOCKS\_PER\_SEC);

fprintf(fp, "%20lf", (end - start) / CLOCKS\_PER\_SEC);

printf("\nФункция быстрой сортировки:\t");

fprintf(fp, "\nФункция быстрой сортировки:\t");

start = clock();

qsort(items, 15000, sizeof(int), cmp);

end = clock();

printf("%12lf", (end - start) / CLOCKS\_PER\_SEC);

fprintf(fp, "%12lf", (end - start) / CLOCKS\_PER\_SEC);

}

int main() {

setlocale(LC\_ALL, "rus");

srand(time(0));

int i, j;

fp = fopen("lab2.2.txt", "w");

printf("Случайные числа\n ");

fprintf(fp, "Случайные числа\n ");

for (i = 0; i < 15000; i++) {

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

}

sort(mas);

printf("\n\nВозрастающая последовательность\n ");

fprintf(fp, "\n\nВозрастающая последовательность\n ");

for (i = 0; i < 15000; i++) {

mas[i] = i;

}

sort(mas);

printf("\n\nУбывающая последовательность\n ");

fprintf(fp, "\n\nУбывающая последовательность\n ");

for (i = 0; i < 15000; i++) {

mas[i] = 15000 - i;

}

sort(mas);

printf("\n\nВозрастающе-убывающая последовательность\n ");

fprintf(fp, "\n\nВозрастающе-убывающая последовательность\n ");

for (i = 0; i < 7500; i++) {

mas[i] = mas [i-1] + 1;

}

for (i = 7500; i < 15000; i++) {

mas[i] = 15000 - i;

}

sort(mas);

fclose(fp);

getch();

return(0);

}

Количество чисел в массиве – 15000. Время работы измеряется в секундах.

**Результат работы программы**

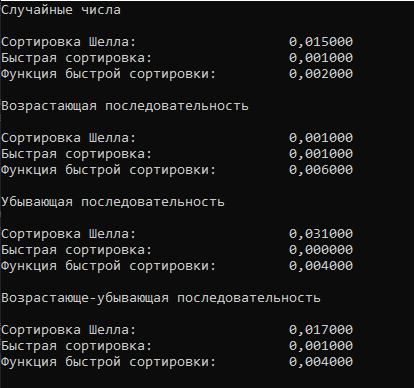


Рисунок 1 - Результат работы программы

**Вывод:** научились вычислять порядок сложности программы, рассчитывать и оценивать время работы программы; сортировка шелла выполняется дольше, чем быстрая сортировка и стандартная функция qsort; самая лучшая сортировка по времени выполнения – быстрая сортировка.