Министерство высшего образования и науки

Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Оценка времени выполнения программы»

Выполнили:

студенты группы 21ВВ2:

Хабибулин А.М.

Щеглов Д.А.

Приняли:

Митрохин М.А.

Юрова О.В.

**Задание 1:**

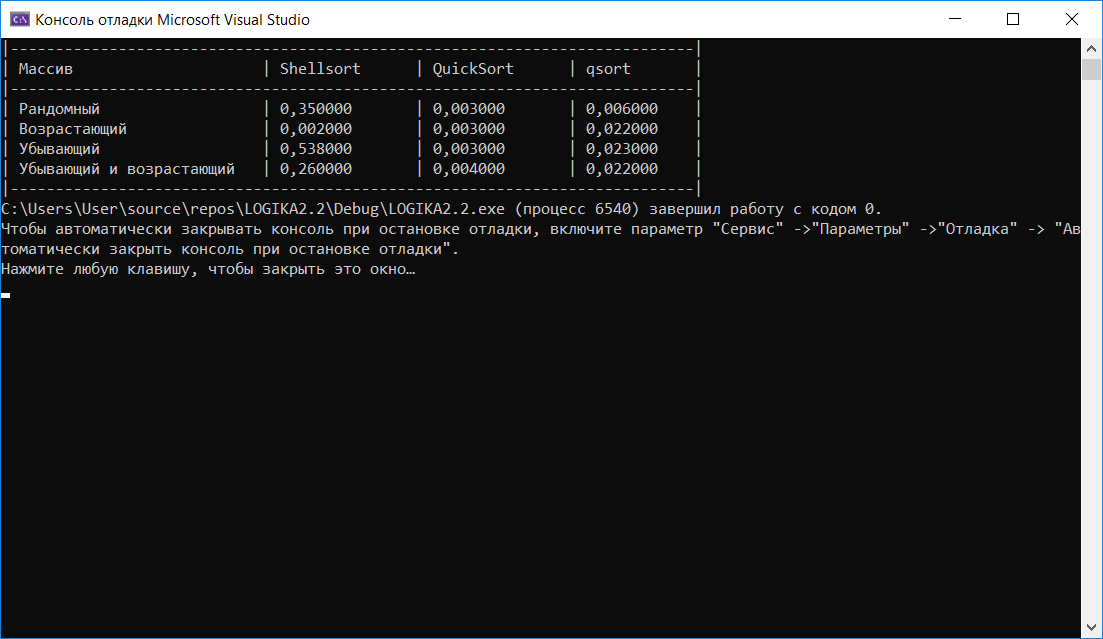
1. Вычислить порядок сложности программы (*О*-символику).
2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

|  |  |  |
| --- | --- | --- |
| № теста | Размер массива | Результат выполнения алгоритма в секундах |
| 1 | 100 | 0,00600 |
| 2 | 200 | 0,03600 |
| 3 | 400 | 0,29100 |
| 4 | 1000 | 5,87500 |
| 5 | 2000 | 64.3400 |
| 6 | 4000 | 640.100 |
| 7 | 10000 | 6200.72 |

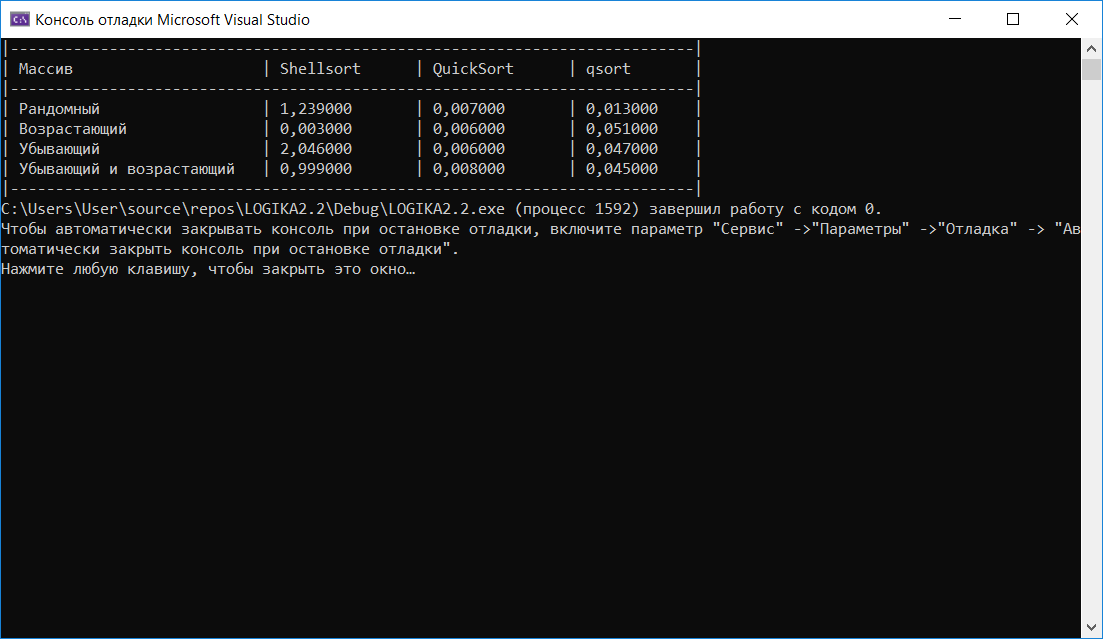
**Задание 2**:

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.

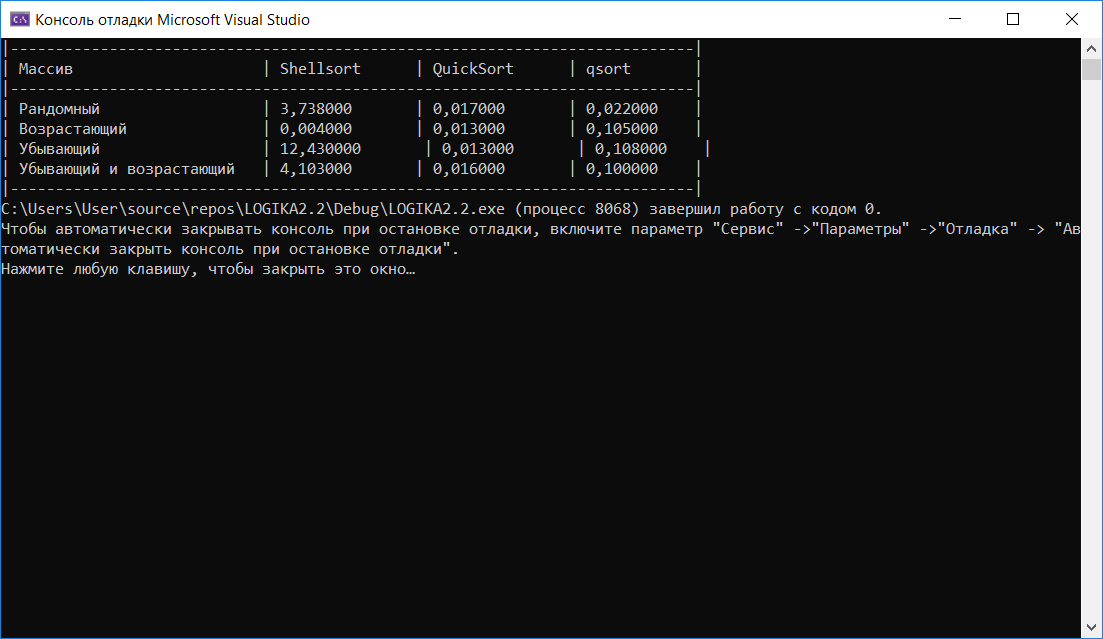
Размерность массива =50 000



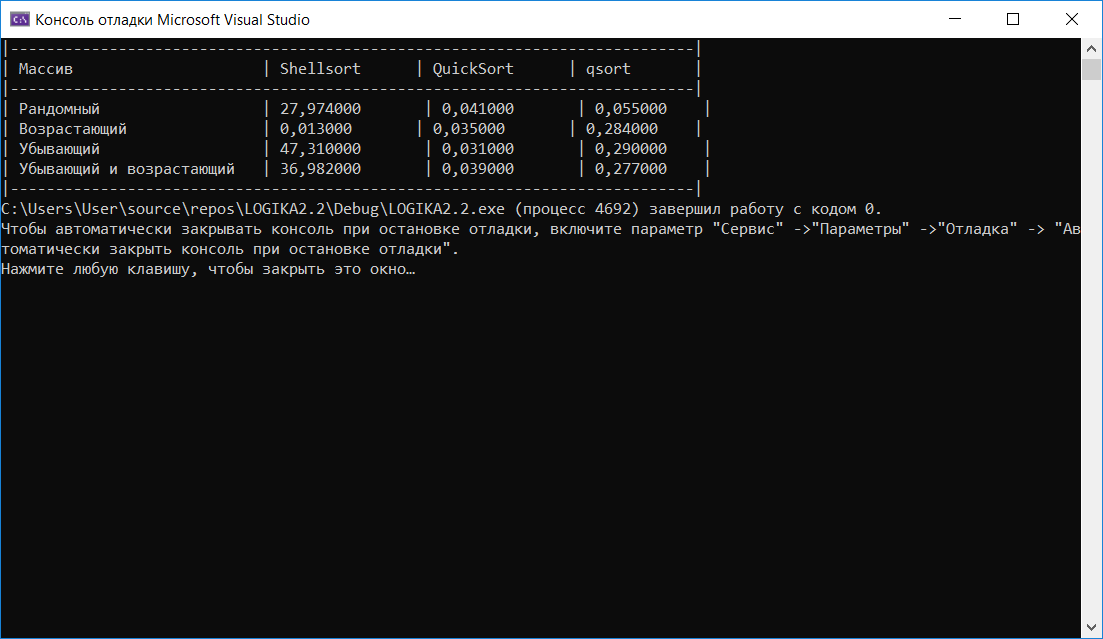
Размерность массива =100 000



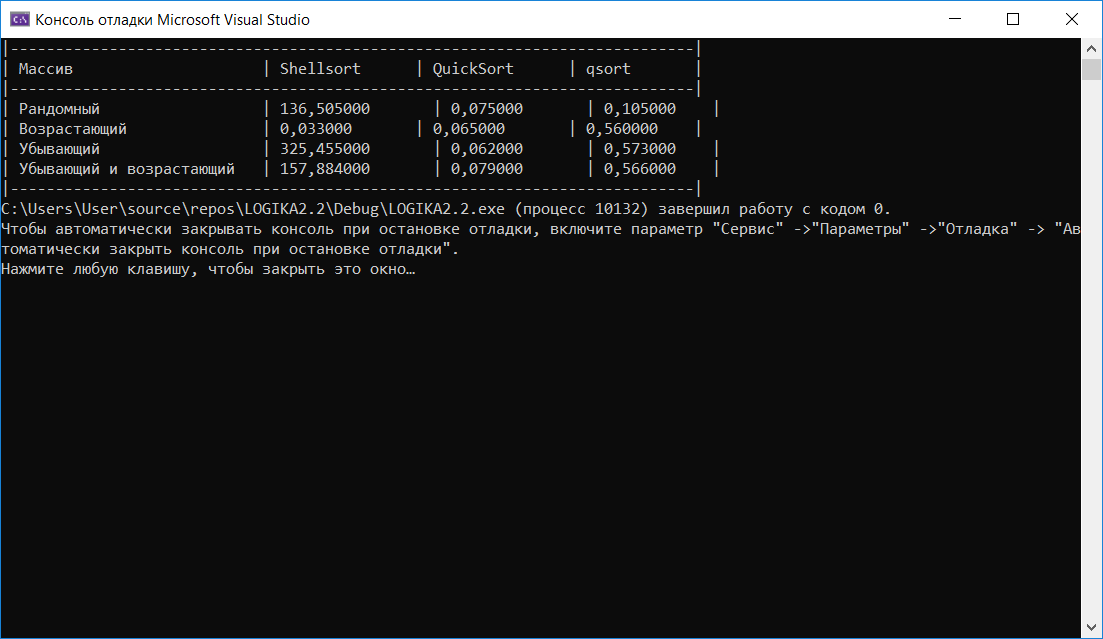
Размерность массива =200 000



Размерность массива = 500 000



Размерность массива = 1 000 000



**Листинг программы**

**2.1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#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, k;

int N, n;

int\*\* A;

int\*\* B;

int\*\* C;

printf("Введите размерность массива: ");

scanf("%d", &N);

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

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

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

{

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

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

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

{

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

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

}

}

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

unsigned int start\_time = clock();

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

{

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

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

{

n = 0;

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

{

n = n + A[i][k] \* B[k][j];

C[i][j] = n;

}

}

}

unsigned int end\_time = clock();

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

printf("Время выполнения программы %f секунд\n\n", search\_time);

system("pause");

}

**2.2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<locale>

#include <iostream>

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) //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);

}

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

{ return \*i - \*j;

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(NULL));

clock\_t startTime, stopTime;

double timeA;

double timeB;

double timeC;

double timeD;

double timeE;

double timeF;

double timeG;

double timeK;

double timeL;

double timeM;

double timeX;

double timeZ;

const int size = 10000;

int i = 0, N;

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

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

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

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

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

a[i] = rand() % 10;

}

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

b[i] = i \* 2;

}

c[0] = 0;

for (i = 1; i < size; i++) {

c[i] = c[0] - i \* 2;

}

if (size % 2 == 0) {

N = size / 2;

}

else {

N = size / 2 + 1;

}

d[0] = 0;

for (i = 1; i < N; i++) {

d[i] = d[i - 1] + 3;

}

for (i = N; i < size; i++) {

d[i] = d[i - 1] - 3;

}

startTime = clock();

shell(a, size);

stopTime = clock();

timeA = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

shell(b, size);

stopTime = clock();

timeB = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

shell(c, size);

stopTime = clock();

timeC = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

shell(d, size);

stopTime = clock();

timeD = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

qs(a, 0, size - 1);

stopTime = clock();

timeE = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

qs(b, 0, size - 1);

stopTime = clock();

timeF = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

qs(c, 0, size - 1);

stopTime = clock();

timeG = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

qs(d, 0, size - 1);

stopTime = clock();

timeK = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

qsort(a, size, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

stopTime = clock();

timeL = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

qsort(b, size, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

stopTime = clock();

timeM = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

qsort(c, size, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

stopTime = clock();

timeX = double(stopTime - startTime) / CLK\_TCK;

startTime = clock();

qsort(d, size, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

stopTime = clock();

timeZ = double(stopTime - startTime) / CLK\_TCK;

printf("|----------------------------------------------------------------------------|");

printf("\n");

printf("| Массив | Shellsort | QuickSort | qsort |");

printf("\n");

printf("|----------------------------------------------------------------------------|");

printf("\n");

printf("| Рандомный | %f", timeA);

printf(" | %f", timeE);

printf(" | %f", timeL);

printf(" |");

printf("\n");

printf("| Возрастающий | %f", timeB);

printf(" | %f", timeF);

printf(" | %f", timeM);

printf(" |");

printf("\n");

printf("| Убывающий | %f", timeC);

printf(" | %f", timeG);

printf(" | %f", timeX);

printf(" |");

printf("\n");

printf("| Убывающий и возрастающий | %f", timeD);

printf(" | %f", timeK);

printf(" | %f", timeZ);

printf(" |");

printf("\n");

printf("|----------------------------------------------------------------------------|");

free(a);

free(b);

free(c);

free(d);

}

**Вывод:** В первой части лабораторной работы определили сложность алгоритма, равна O(n3). А во второй части были разработаны программы с использованием массивов и реализованы алгоритмы сортировки Шелла, быстрой сортировки и qsort. Сортировка Шелла оказалась эффективна на возрастающем массиве, быстрая сортировка и qsort оказались эффективны на всех видах массивов.