Министерство науки и высшего образования 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 программ»

Выполнили студенты группы 22ВВП2:

Корнилов В.М.

Самофалова А.В.

Горбунов Д.А.

Принял:

Акифьев И.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,

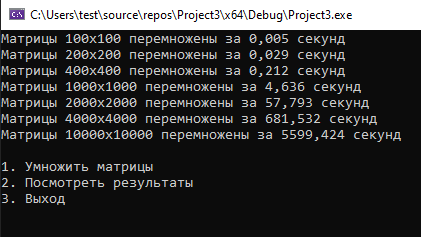
представляющем собой убывающую последовательность чисел.

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

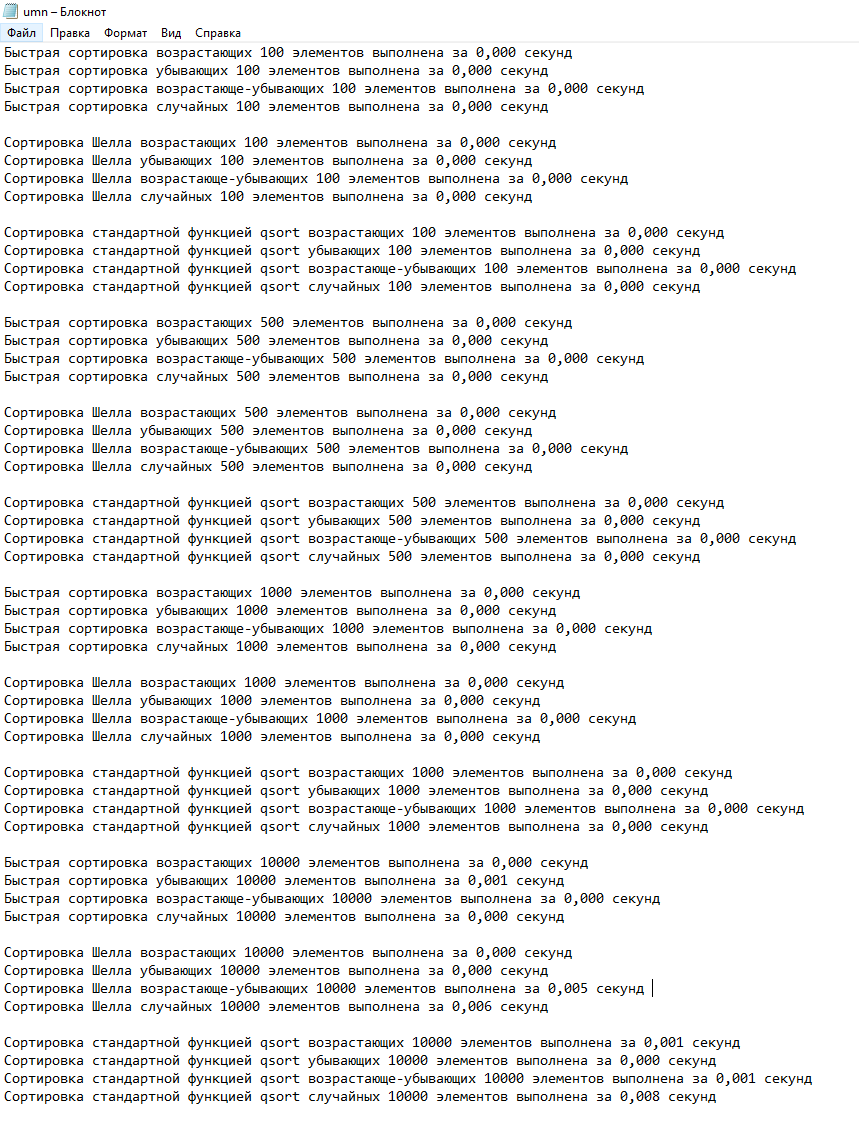
5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

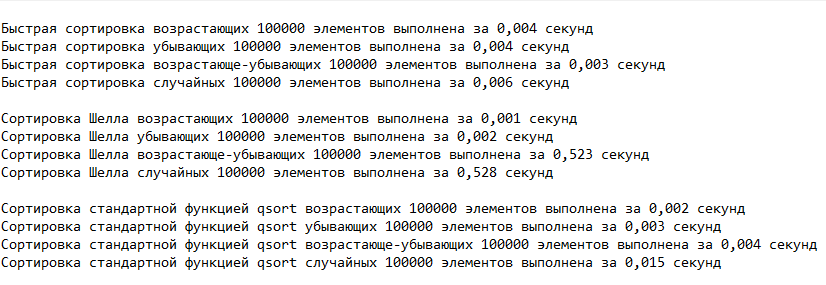
**Результаты работы программы:**

**Задание 1.**



**Задание 2.**





**Вывод:**

**Задание 1.** Уровень сложности данного алгоритма O(n^3)



**Задание 2.** Оценили работу всех алгоритмов на разных наборах данных. Наилучшие алгоритмы для последовательностей:

Возрастающая – Сортировка Шелла (на 100 – 100000+ элементов)

Убывающая – Сортировка Шелла (на 100-100000+ элементов)

Возрастающе-убывающая – Быстрая сортировка (qs) (100-100000+ элементов)

Случайных чисел – Быстрая сортировка (qs) (100-100000+ элементов)

**Приложение А**

**Листинг**

**Файл main1.c**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

void xxx()

{

setlocale(LC\_ALL, "RUSSIAN");

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

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

int i = 0, j = 0, r, elem\_c, m, n;

printf("Введите размер матрицы: ");

scanf("%d", &m);

n = m;

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

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

\*(a + i) = (int\*)malloc(sizeof(int\*) \* n);

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

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

\*(b + i) = (int\*)malloc(sizeof(int\*) \* n);

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

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

\*(c + i) = (int\*)malloc(sizeof(int\*) \* n);

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

while (i < m)

{

while (j < n)

{

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

j++;

}

j = 0;

i++;

}

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

i = 0; j = 0;

while (i < m)

{

while (j < n)

{

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

j++;

}

j = 0;

i++;

}

time\_t start1 = clock();//время до сортировки

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

{

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

{

elem\_c = 0;

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

{

elem\_c = elem\_c + a[i][r] \* b[r][j];

c[i][j] = elem\_c;

}

}

}

time\_t stop = clock();//время до сортировки

double time2 = ((double) stop - (double) start1) / CLOCKS\_PER\_SEC;//время сортировки

printf("Время умножения: %5.3f секунд, результат записан в файл", time2);

FILE\* res = fopen("umn.txt", "a");

fprintf(res, "Матрицы %dx%d перемножены за %5.3f секунд \n", m, m, time2);

fclose(res);

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

free(a[i]);

free(a);

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

free(b[i]);

free(b);

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

free(c[i]);

free(c);

menu();

}

void file() {

FILE\* res;

char str[50];

res = fopen("umn.txt", "r");

while (!feof(res)) {

if (fgets(str, 48, res)) printf("%s", str);

}

fclose(res);

menu();

}

int menu() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int cc;

printf("\n1. Умножить матрицы\n");

printf("2. Посмотреть результаты\n");

printf("3. Выход\n");

scanf("%d", &cc);

switch (cc) {

case 1:

xxx();

break;

case 2:

file();

break;

case 3:

\_Exit(0);

break;

}

return 0;

}

int main(){

menu();

return 0;

}

**Файл main2.c**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

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

return \*i - \*j;

}

int main() {

int n;

setlocale(LC\_ALL, "RUSSIAN");

srand(time(NULL));

printf("Введите размер массива: \n");

scanf("%d", &n);

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

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

for (int l = 0; l < n; l++)

b[i] = l;

}

time\_t start1 = clock();

qs(b, 0, n - 1);

time\_t stop = clock();

double time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

FILE\* res = fopen("umn.txt", "a");

fprintf(res, "Быстрая сортировка возрастающих %d элементов выполнена за %5.3f секунд \n", n, time2);

for (int j = 0; j < n; j++) {

for (int l = n; l > 0; l--)

b[j] = l;

}

start1 = clock();

qs(b, 0, n - 1);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Быстрая сортировка убывающих %d элементов выполнена за %5.3f секунд \n", n, time2);

for (int i = 0; i < n / 2; i++) {

for (int l = 0; l < n / 2; l++)

b[i] = l;

}

for (int j = n / 2; j < n; j++) {

for (int l = n / 2 - 1; l > 0; l--)

b[j] = l;

}

start1 = clock();

qs(b, 0, n - 1);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Быстрая сортировка возрастающе-убывающих %d элементов выполнена за %5.3f секунд \n", n, time2);

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

b[j] = rand() % 100;

start1 = clock();

qs(b, 0, n-1);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Быстрая сортировка случайных %d элементов выполнена за %5.3f секунд \n", n, time2);

fprintf(res, "\n");

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

for (int l = 0; l < n; l++)

b[i] = l;

}

start1 = clock();

shell(b, n);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Сортировка Шелла возрастающих %d элементов выполнена за %5.3f секунд \n", n, time2);

for (int j = 0; j < n; j++) {

for (int l = n; l > 0; l--)

b[j] = l;

}

start1 = clock();

shell(b, n);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Сортировка Шелла убывающих %d элементов выполнена за %5.3f секунд \n", n, time2);

for (int i = 0; i < n / 2; i++) {

for (int l = 0; l < n / 2; l++)

b[i] = l;

}

for (int j = n / 2; j < n; j++) {

for (int l = n / 2 - 1; l > 0; l--)

b[j] = l;

}

start1 = clock();

shell(b, n);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Сортировка Шелла возрастающе-убывающих %d элементов выполнена за %5.3f секунд \n", n, time2);

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

b[j] = rand() % 100;

start1 = clock();

shell(b, n);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Сортировка Шелла случайных %d элементов выполнена за %5.3f секунд \n", n, time2);

fprintf(res, "\n");

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

for (int l = 0; l < n; l++)

b[i] = l;

}

start1 = clock();

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

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Сортировка стандартной функцией qsort возрастающих %d элементов выполнена за %5.3f секунд \n", n, time2);

for (int j = 0; j < n; j++) {

for (int l = n; l > 0; l--)

b[j] = l;

}

start1 = clock();

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

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Сортировка стандартной функцией qsort убывающих %d элементов выполнена за %5.3f секунд \n", n, time2);

for (int i = 0; i < n / 2; i++) {

for (int l = 0; l < n / 2; l++)

b[i] = l;

}

for (int j = n / 2; j < n; j++) {

for (int l = n / 2 - 1; l > 0; l--)

b[j] = l;

}

start1 = clock();

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

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Сортировка стандартной функцией qsort возрастающе-убывающих %d элементов выполнена за %5.3f секунд \n", n, time2);

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

b[j] = rand() % 100;

start1 = clock();

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

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

fprintf(res, "Сортировка стандартной функцией qsort случайных %d элементов выполнена за %5.3f секунд \n", n, time2);

fprintf(res, "\n");

printf("Сортировка успешно завершена)");

fclose(res);

free(b);

}