Пензенский государственный университет

Кафедра “Вычислительная техника”

Отчёт  
о выполнении лабораторной работы №2  
по дисциплине "Логика и основы алгоритмизации в инженерных задачах"  
на тему  
**"**Оценка времени выполнения программ**"**

Выполнили студенты гр.21ВВ3:  
Борисова А. В.  
Саломатин В. А.

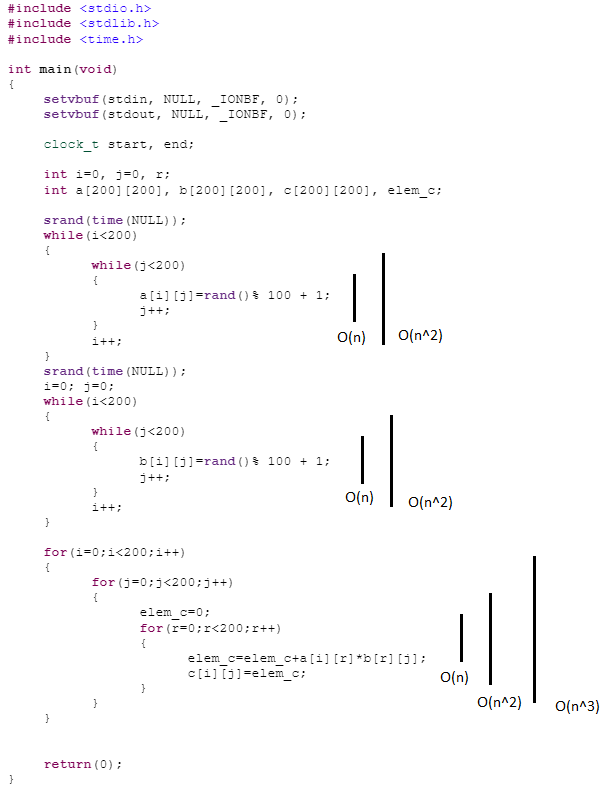
Принял:  
д.т.н. профессор Митрохин М.А. к.т.н. доцент Юрова О.В.

Пенза 2022

Лабораторное задание:

**Задание 1:**

1. Вычислить порядок сложности программы (*О*-символику).



O(max(n, n^2, n^3)) = O(n^3)

1. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 20, 50, 80, 100, 150, 200, 250.

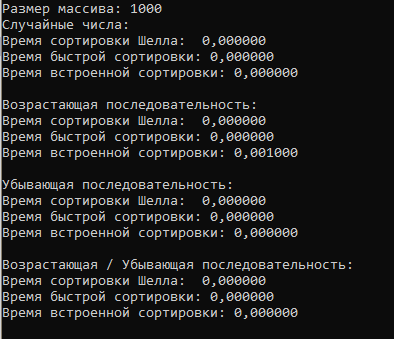
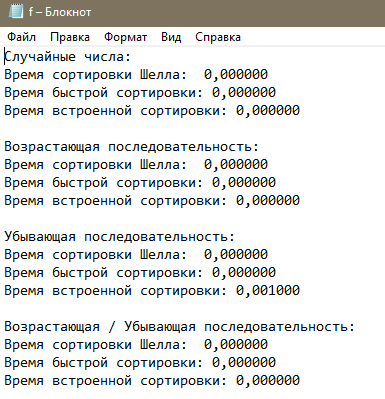
|  |  |  |
| --- | --- | --- |
| № Теста | Размер матриц | Время выполнения программы |
| 1. | 20 | 0,010 |
| 2. | 50 | 0,010 |
| 3. | 80 | 0,010 |
| 4. | 100 | 0,010 |
| 5. | 150 | 0,020 |
| 6. | 200 | 0,040 |
| 7. | 250 | 0,095 |

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

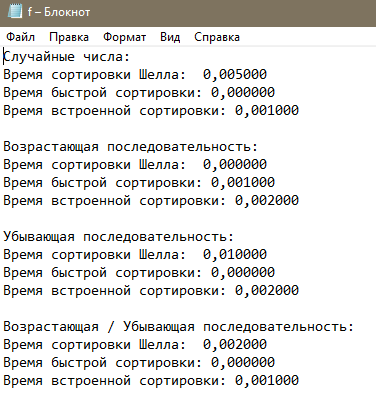
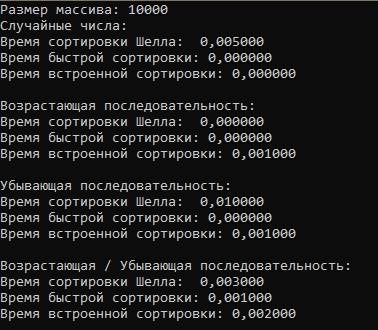
**Задание 2**:

Оценили время работы каждого из реализованных алгоритмов на случайном наборе значений, c возрастающей последовательностью чисел, c убывающей последовательностью чисел, сначала с возрастающей последовательностью чисел, а затем – убывающей. Оценили время работы стандартной функции qsort,

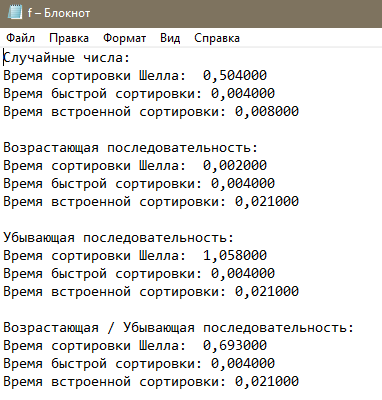
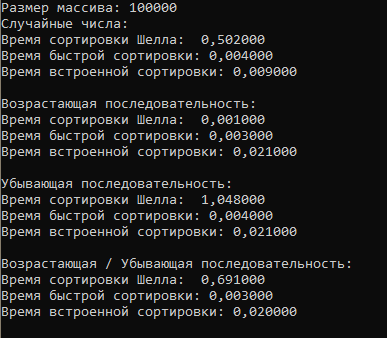
Количество элементов – **1000**:

Количество элементов – **10000**:



Количество элементов – **100000**:



**Листинг:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_CTR\_NONSTDC\_NO\_WARNINGS

#include <stdio.h>

#include <conio.h>

#include <time.h>

#include <stdlib.h>

#include <random>

#include <locale>

void shell(int \*mass, int n)

{

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 < n; ++i) {

x = mass[i];

for (j = i - gap; (x < mass[j]) && (j >= 0); j = j - gap)

mass[j + gap] = mass[j];

mass[j + gap] = x;

}

}

}

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

{

int i, j;

int x, y;

i = left;

j = right;

x = mass[(left + right) / 2];

do {

while ((mass[i] < x) && (i < right))i++;

while ((x < mass[j]) && (j > left))j--;

if (i <= j)

{

y = mass[i];

mass[i] = mass[j];

mass[j] = y;

i++; j--;

}

} while (i <= j);

if (left < j) qs(mass, left, j);

if (i < right) qs(mass, i, right);

}

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

return \*(int \*)i - \*(int \*)j;

}

void sort(FILE \*f, int \*mass, int n) {

time\_t start1 = clock();

shell(mass, n);

time\_t stop1 = clock();

double time1 = (stop1 - start1) / 1000.0;

printf("\n");

printf("Время сортировки Шелла: %lf", time1);

fprintf(f, "Время сортировки Шелла: %lf\n", time1);

printf("\n");

time\_t start2 = clock();

qs(mass, 0, n - 1);

time\_t stop2 = clock();

double time2 = (stop2 - start2) / 1000.0;

printf("Время быстрой сортировки: %lf", time2);

fprintf(f, "Время быстрой сортировки: %lf\n", time2);

printf("\n");

time\_t start3 = clock();

qsort(mass, n, sizeof(int), comp);

time\_t stop3 = clock();

double time3 = (stop3 - start3) / 1000.0;

printf("Время встроенной сортировки: %lf", time3);

fprintf(f, "Время встроенной сортировки: %lf\n\n", time3);

}

int main()

{

FILE \*f;

int \*mass;

int n;

f = fopen("f.txt", "w");

setlocale(LC\_ALL, "RUS");

printf("Размер массива: ");

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

srand(time(NULL));

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

printf("Случайные числа:");

fprintf(f, "Случайные числа:\n");

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

mass[i] = rand() % 100;

}

sort(f, mass, n);

printf("\n");

printf("\n");

printf("Возрастающая последовательность:");

fprintf(f, "Возрастающая последовательность:\n");

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

mass[i] = i;

}

sort(f, mass, n);

printf("\n");

printf("\n");

printf("Убывающая последовательность:");

fprintf(f, "Убывающая последовательность:\n");

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

mass[i] = n - i;

}

sort(f, mass, n);

printf("\n");

printf("\n");

printf("Возрастающая / Убывающая последовательность:");

fprintf(f, "Возрастающая / Убывающая последовательность:\n");

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

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

}

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

mass[i] = n - i;

}

sort(f, mass, n);

fclose(f);

free(mass);

getchar();

getchar();

return(0);

}

**Вывод:** в ходе лабораторной работы мы оценили время, затраченное программой, на осуществление некоторых сортировок. Сравнив показатели, выяснили, что быстрая сортировка затрачивает меньше всего времени.