Министерство 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 19ВВ2

Пристяжнюк А.М.

Приняли:

старший преподаватель

Митрохин М.А.

### Название

Программа, выполняющая сортировку данных.

### Цель работы

Оценить время работы предложенных алгоритмов сортировки. Научиться вычислять порядок сложности программы

**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.

### 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.

### Пояснительный текст к программе

Программа А представляет из себя предложенный к работе код, вычисляющий произведение двух матриц с выводом времени работы алгоритма в конце.

Программа B выполнена в виде меню, которое включает в себя 4 разных варианта исходных массивов с возможностью выбора одного из трёх алгоритмов сортировки(shell, quicksort, quicksort(vs)):

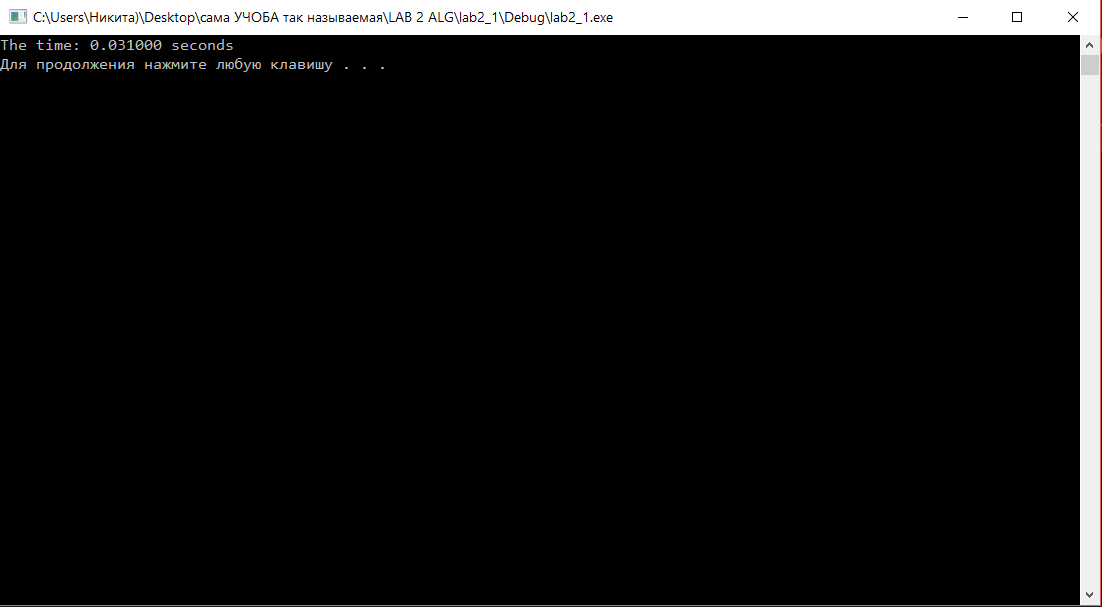
1. Массив со случайными значениями.
2. Массив, представляющий возрастающую последовательность чисел.
3. Массив, представляющий убывающую последовательность чисел.
4. Массив, половина которого возрастает, а вторая убывает.

### Результаты работы программы

|  |  |
| --- | --- |
| Размер массива, эл. | Время выполнения, с. |
| 100\*100 | 0.005 |
| 200\*200 | 0.031 |

**Таблица 1 — зависимость времени выполнения от размера массива в программе A.**

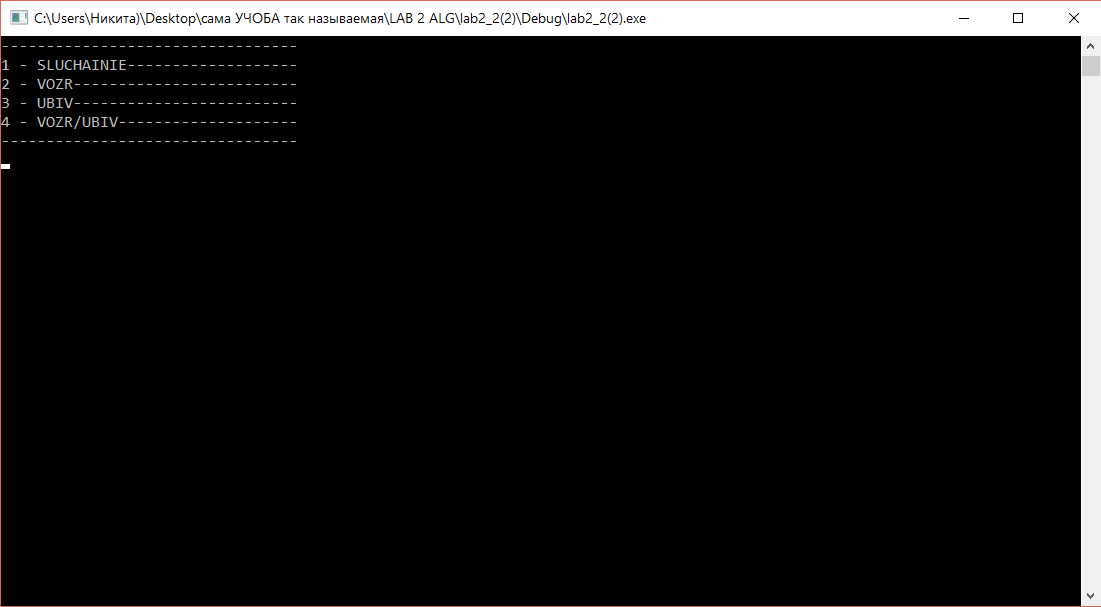
Порядок сложности алгоритма - O(N^3)

****

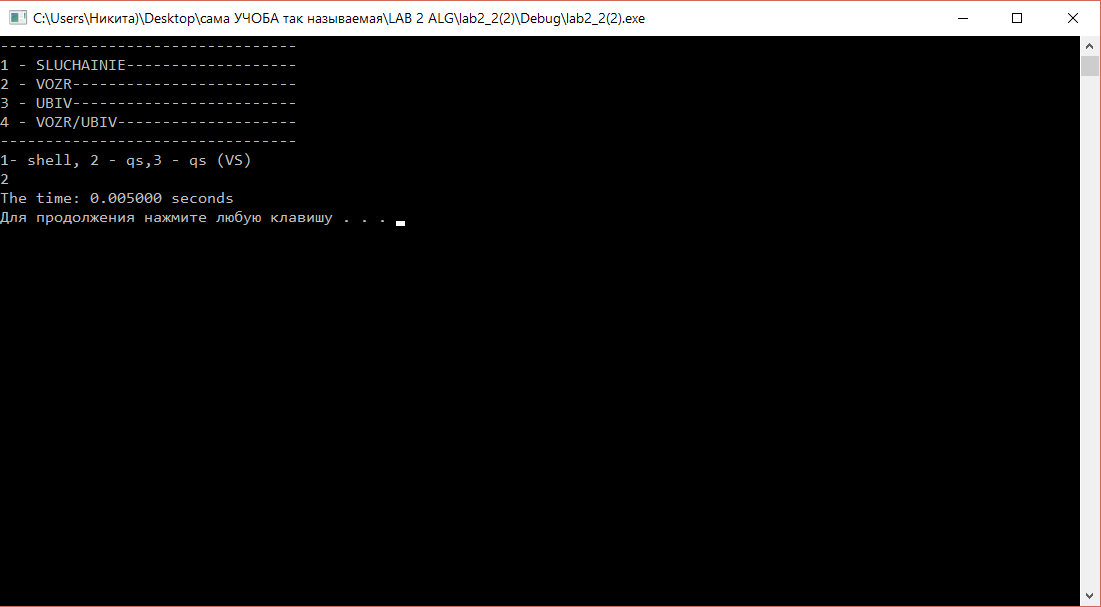
**Рисунок 1 — Результаты работы программы A (Главное меню)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Данные в массиве** | Shell, с. | Quicksort, с. | Quicksort(VS), с. |
| Случайные числа | 0.168 | 0.005 | 0.013 |
| Возрастающий | 0.001 | 0.003 | 0.017 |
| Убывающий | 0.315 | 0.003 | 0.018 |
| Возр./Убыв. | 0.241 | 0.014 | 0.026 |

**Таблица 1 — зависимость времени выполнения разных алгоритмов от типа массива в программе B.**

****

**Рисунок 2 — Результаты работы программы B (главное меню)**

****

**Рисунок 3 — Результаты работы программы B(Пример выполнения quicksort на 1 массиве)**

### Выводы

В ходе выполнения лабораторной работы была разработана программа, выполняющая сортировку данных

Получили опыт в создании проектов в среде Microsoft Visual Studio, научились писать и отлаживать линейные программы на языке Си.

### Листинг программы A

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main(void)

{

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

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

int i = 0, j = 0, r;

const int n = 200;

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

clock\_t start = clock();

srand(time(NULL)); // генератор случайных от времени

while (i<n)

{

while (j<n)

{

a[i][j] = rand() % 100 + 1; // присваивание случ. знач.

j++;

}

i++;

}

srand(time(NULL)); // генератор случайных от времени

i = 0; j = 0;

while (i<n)

{

while (j<n)

{

b[i][j] = rand() % 100 + 1; // присваивание случ. знач.

j++;

}

i++;

}

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;

}

}

}

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

return(0);

}

### Листинг программы B

### Файл «Source.cpp»

#include "header.h"

int main()

{

system("CLS");

while (1)

{

menu();

menu\_input(\_getch());

}

}

### Файл «Header.h»

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include <cstdlib>

#include <malloc.h>

#include <conio.h>

#include <string.h>

const int z = 200;

int compare(const void \* x1, const void \* x2);

void ex1\_s();

void ex2\_s();

void ex3\_s();

void ex4\_s();

void ex5\_s();

void shell(int \*items, int count);

void menu();

void menu\_input(char i);

void qs(int \*items, int left, int right);

Файл «func.cpp»

#include "header.h"

#include <iostream>

void ex4\_s(){

// СОЗДАНИЕ И ЗАПОЛНЕНИЕ МАССИВА

int i = 0, j = 0, r, n = 1, x = z;

int a[z][z];

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

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

{

a[i][j] = n;

a[x / 2 + i][j] = 4999 - n;

n++;

}

if (x % 2) // n - нечетное, осталась одна незаполненная строка в конце

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

{

a[x - 1][j] = 4999 - n;

n++;

}

int \*f = &a[0][0];

// ВЫПОЛНЕНИЕ АЛГОРИТМА

printf("1- shell, 2 - qs,3 - qs (VS)\n");

int symbol = 0;

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

if (symbol == 1)

{

clock\_t start = clock();

shell(f, z\*z);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

if (symbol == 2)

{

clock\_t start = clock();

qs(f, 0, (z\*z) - 1);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

if (symbol == 3)

{

clock\_t start = clock();

qsort(f, z\*z, sizeof(int), compare);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

}

void ex3\_s(){

// СОЗДАНИЕ И ЗАПОЛНЕНИЕ МАССИВА

int i = 0, j = 0, r, n = 1;

int a[z][z];

srand(time(NULL));

while (i < z)

{

while (j < z)

{

a[i][j] = (z\*z) - n; // заполнение псевдослучайными

j++;

n++;

}

j = 0;

i++;

}

int \*f = &a[0][0];

// ВЫПОЛНЕНИЕ АЛГОРИТМА

printf("1- shell, 2 - qs,3 - qs (VS)\n");

int symbol = 0;

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

if (symbol == 1)

{

clock\_t start = clock();

shell(f, z\*z);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

if (symbol == 2)

{

clock\_t start = clock();

qs(f, 0, (z\*z) - 1);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

if (symbol == 3)

{

clock\_t start = clock();

qsort(f, z\*z, sizeof(int), compare);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

}

void ex2\_s()

{

int i = 0, j = 0, r, n = 0;;

int a[z][z];

srand(time(NULL));

while (i < z)

{

while (j < z)

{

a[i][j] = n; // заполнение возрастающими

j++;

n++;

}

j = 0;

i++;

}

int \*f = &a[0][0];

printf("1- shell, 2 - qs,3 - qs (VS)\n");

int symbol = 0;

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

if (symbol == 1)

{

clock\_t start = clock();

shell(f, z\*z);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

if (symbol == 2)

{

clock\_t start = clock();

qs(f, 0, (z\*z) - 1);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

if (symbol == 3)

{

clock\_t start = clock();

qsort(f, z\*z, sizeof(int), compare);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

}

void ex1\_s(){

// СОЗДАНИЕ И ЗАПОЛНЕНИЕ МАССИВА

int i = 0, j = 0, r, n = 15;

int a[z][z];

srand(time(NULL));

while (i < z)

{

while (j < z)

{

a[i][j] = rand() % 100; // заполнение псевдослучайными

j++;

}

j = 0;

i++;

}

int \*f = &a[0][0];

// ВЫПОЛНЕНИЕ АЛГОРИТМА + VREMYA

printf("1- shell, 2 - qs,3 - qs (VS)\n");

int symbol = 0;

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

if (symbol == 1)

{

clock\_t start = clock();

shell(f, z\*z);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

if (symbol == 2)

{

clock\_t start = clock();

qs(f, 0, (z\*z) - 1);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

if (symbol == 3)

{

clock\_t start = clock();

qsort(f, z\*z, sizeof(int), compare);

clock\_t end = clock();

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

printf("The time: %f seconds\n", seconds);

system("pause");

}

}

void qs(int \*items, int left, int right) //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 menu()

{

system("CLS");

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

printf("1 - SLUCHAINIE-------------------\n");

printf("2 - VOZR-------------------------\n");

printf("3 - UBIV-------------------------\n");

printf("4 - VOZR/UBIV--------------------\n");

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

}

void menu\_input(char i)

{

if (i == '1'){ ex1\_s(); }

if (i == '2'){ ex2\_s(); }

if (i == '3'){ ex3\_s(); }

if (i == '4'){ ex4\_s(); }

}

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 compare(const void \* x1, const void \* x2) // функция сравнения элементов массива

{

return (\*(int\*)x1 - \*(int\*)x2); // если результат вычитания равен 0, то числа равны, < 0: x1 < x2; > 0: x1 > x2

}