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 22ВВВ3:

Байков А. В.

Гераськина Д. А.

Приняли:

Юрова О. В.

Акифьев И. В.

Пенза 2023

**Цель работы**

Повторить и усовершенствовать полученные ранее знания о функциях, связанных с библиотекой time.h, на языке СИ.

**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.

**Листинг**

**Lab2.1**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#define q 8000

#define w 8000

#pragma comment(linker, "/STACK:615246573000")

int main(void)

{

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

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

setlocale(LC\_ALL, "RUS");

clock\_t start, end; // объявляем переменные для определения времени выполнения

int i = 0, j = 0, r;

int a[q][w], b[q][w], c[q][w], elem\_c;

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

while (i < q)

{

while (j < w)

{

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 < q)

{

while (j < w)

{

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

j++;

}

i++;

}

start = clock(); // Начало отсчета времени

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

{

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

{

elem\_c = 0;

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

{

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

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

}

}

}

end = clock() - start; // Конец отсчета времени

printf("Размер матрицы %dx%d\n\n", q, w);

printf("Время работы: %lf", (double)end / CLOCKS\_PER\_SEC);

return(0);

}

**Lab2.2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

void shell(int\* items, int count)

{

clock\_t start\_s, end\_s;

int i, j, gap, k;

int x, a[5];

a[0] = 9; a[1] = 5; a[2] = 3; a[3] = 2; a[4] = 1;

start\_s = clock(); // Начало отсчета времени работы функции shell

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;

}

}

end\_s = clock() - start\_s; // Конец отсчета времени работы функции shell

printf("%lf", (double)end\_s / CLOCKS\_PER\_SEC); // Вывод времени работы функции shell

}

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);

}

// Функция для работы qsort

int comp(const void\* i1, const void\* i2)

{

return (\*(int\*)i1 - \*(int\*)i2);

}

void work\_w\_file(char name[14], int\* items)

{

int count = 0;

clock\_t start\_qs3, end\_qs3; // Время работы qs

// Сохранение длины файла

FILE\* file\_3 = fopen(name, "r");

fseek(file\_3, 0, SEEK\_SET);

while (true)

{

int val;

if (fscanf(file\_3, "%d", &val) == 1)

{

count++;

}

if (feof(file\_3))

{

break;

}

}

fclose(file\_3);

items = (int\*)malloc(count \* sizeof(int)); // Расширение памяти

//

// Запись в массив из файла множества чисел

file\_3 = fopen(name, "r");

fseek(file\_3, 0, SEEK\_SET);

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

fscanf(file\_3, "%d", &(\*(items + i)));

}

fclose(file\_3);

//

clock\_t start3 = clock(), end3; // Начало отсчета времени qsort

qsort(items, count, sizeof(int), comp); // Вызов функции qsort

end3 = clock() - start3; // Время работы qsort

printf(" %lf ", (double)end3 / CLOCKS\_PER\_SEC); // Вывод времени работы qsort

file\_3 = fopen(name, "r");

fseek(file\_3, 0, SEEK\_SET);

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

fscanf(file\_3, "%d", &(\*(items + i)));

}

fclose(file\_3);

shell(items, count); // Вызов функции shell

file\_3 = fopen(name, "r");

fseek(file\_3, 0, SEEK\_SET);

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

fscanf(file\_3, "%d", &(\*(items + i)));

}

fclose(file\_3);

start\_qs3 = clock(); // Начало отсчета времени работы qs

qs(items, 1, count - 1); // Вызов функции qsort

end\_qs3 = clock() - start\_qs3; // Конец отсчета времени работы qs

printf(" %lf\n", (double)end\_qs3 / CLOCKS\_PER\_SEC); // Вывод времени работы qsort

free(items); // Освобождение памяти

}

int main(clock\_t end\_s)

{

setlocale(LC\_ALL, "RUS"); // Подключение русского языка

srand(time(NULL)); // Подключение случайных чисел

char name1[14] = "HeightMas.txt"; // Файл с возрастающей последовательностью чисел

char name2[14] = "DownMas.txt"; // Файл с убывающей последовательностью чисел

char name3[14] = "HeDowMas.txt"; // Файл с возрастающей-убывающей последовательностью чисел

int\* items; // Указатель

int count = 40000; // Счетчик

clock\_t start\_qs, end\_qs; // Время работы qs

int what; // Для определения вида задачи

// 1 Задание

printf("\nРазмер\t qsort shell qs\n");

printf("%d(:::)", count);

items = (int\*)malloc(count \* sizeof(int)); // Расширение памяти

// Заполнение случайными числами

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

\*(items + i) = rand() % 100;

}

clock\_t start = clock(), end; // Начало отсчета времени работы qsort

qsort(items, count, sizeof(int), comp); // Вызов qsort

end = clock() - start; // Конец отсчета времени работы qsort

printf(" %lf ", (double)end / CLOCKS\_PER\_SEC); // Вывод времени работы qsort

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

\*(items + i) = rand() % 100;

}

shell(items, count); // Вызов shell

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

\*(items + i) = rand() % 100;

}

start\_qs = clock(); // Начало отсчета времени работы qs

qs(items, 0, count - 1); // Вызов qs

end\_qs = clock() - start\_qs; // Конец отсчета времени работы qs

printf(" %lf\n", (double)end\_qs / CLOCKS\_PER\_SEC); // Вывод времени работы qs

free(items); // Освобождение памяти

// 2 задание

printf("40000(/)");

work\_w\_file(name1, items);

// 3 задание

printf("40000(\\)");

work\_w\_file(name2, items);

// 4 задание

printf("40000(/\\)");

work\_w\_file(name3, items);

return 0;

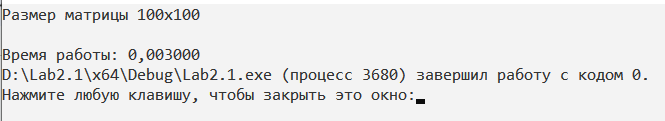
}

**Задание 1**

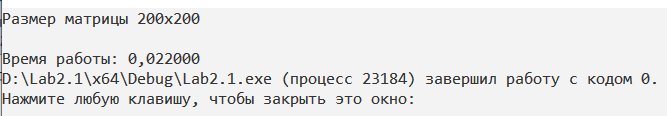
1. Сложность О(N^3)

3.

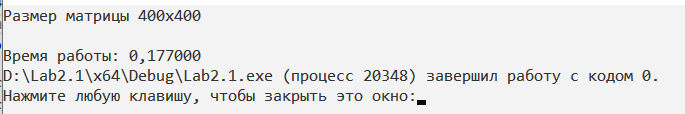
**Результаты работы программ**



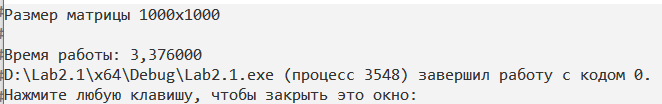
**Рисунок 1 - Результат работы программы Lab2.1**



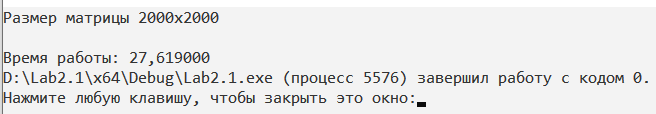
**Рисунок 2 - Результат работы программы Lab2.1**



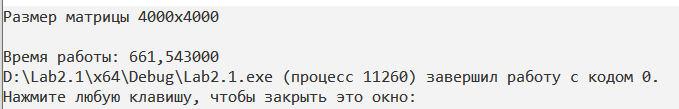
**Рисунок 3 - Результат работы программы Lab2.1**



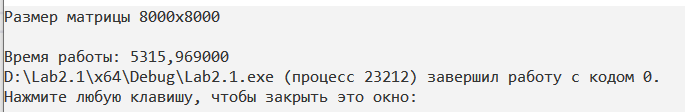
**Рисунок 4 - Результат работы программы Lab2.1**



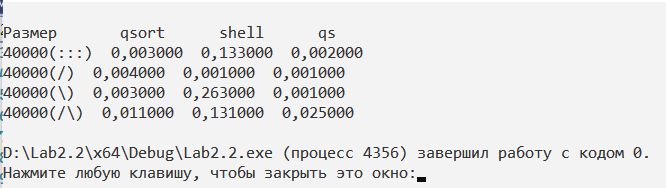
**Рисунок 5 - Результат работы программы Lab2.1**



**Рисунок 6 - Результат работы программы Lab2.1**



**Рисунок 7 - Результат работы программы Lab2.1**



**Рисунок 8 - Результат работы программы Lab2.2**

**Вывод**

В ходе работы, удалось повторить и усовершенствовать полученные ранее знания о функциях, связанных с библиотекой time.h, на языке СИ. Также для оптимизации времени при работе с сортировкой массива разными способами, выяснили: для алгоритмов на случайном наборе значений массива больше подходит быстрая сортировка; для алгоритмов на массиве, представляющем собой возрастающую последовательность чисел – сортировка Шелла и быстрая сортировка; для алгоритмов на массиве, представляющем собой убывающую последовательность чисел – быстрая сортировка; для алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую – встроенный вид сортировки qsort.