**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 23ВВВ1 :

Жиганов Н.Д.

Видяев А.А.

Приняли:

д.т.н. доцент Митрохин М.А.

к.т.н. доцент Юрова О.В.

Пенза 2024

**Цель работы.**

Изучить работу с библиотекой time.h в языке Си и закрепить на практике.

**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.

400х400 = 0.256000

1000х1000 = 4.947000

2000х2000 = 56.853001

4000х4000 = 1046.333984

10000x10000 = 7653.038086

**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.

**Ход работы:**

**1)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main(void)

{

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

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

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

int i = 0, j = 0, r, min = -100, max = 100;

int elem\_c, m = 10000, n = 10000;

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

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

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

{

mas[i] = (int\*)malloc(sizeof(int) \* m);

}

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

{

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

{

mas[i][j] = min + rand() % (max - min + 1);

//printf("%d ",mas[i][j]);

}

}

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

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

{

mass[i] = (int\*)malloc(sizeof(int) \* m);

}

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

{

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

{

mass[i][j] = min + rand() % (max - min + 1);

//printf("%d ",mas[i][j]);

}

}

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

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

{

c[i] = (int\*)malloc(sizeof(int) \* m);

}

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

{

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

{

c[i][j] = min + rand() % (max - min + 1);

//printf("%d ",mas[i][j]);

}

}

clock\_t starttime;

starttime = clock();

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 + mas[i][r] \* mass[r][j];

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

}

}

}

float t = (float)(clock() - starttime) / (float)CLOCKS\_PER\_SEC;

printf("%f", t);

getchar();

getchar();

getchar();

return(0);

}

**2)**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

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);

}

void main()

{

clock\_t start, end;

int i = 0, j = 0, r, min = -100, max = 100, c = 6000;

int mas1[6000], mas2[6000], mas3[6000], mas4[6000];

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

{

mas1[i] = min + rand() % (max - min + 1);

mas2[i] = i;

mas3[i] = c - i;

if (i <= c / 2)

{

mas4[i] = i;

}

else

{

mas4[i] = c - i;

}

}

clock\_t starttime, endtime;

starttime = clock();

shell(mas1, c);

endtime = clock();

float t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;

printf("shell1: %f\n", t);

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

{

mas1[i] = min + rand() % (max - min + 1);

}

starttime = clock();

qs(mas1, 0, c - 1);

endtime = clock();

t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;

printf("qs1: %f\n", t);

starttime = clock();

shell(mas2, c);

endtime = clock();

t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;

printf("shell2: %f\n", t);

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

{

mas2[i] = i;

}

starttime = clock();

qs(mas2, 0, c - 1);

endtime = clock();

t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;

printf("qs2: %f\n", t);

starttime = clock();

shell(mas3, c);

endtime = clock();

t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;

printf("shell3: %f\n", t);

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

{

mas3[i] = c - i;

}

starttime = clock();

qs(mas3, 0, c - 1);

endtime = clock();

t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;

printf("qs3: %f\n", t);

starttime = clock();

shell(mas4, c);

endtime = clock();

t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;

printf("shell4: %f\n", t);

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

{

if (i <= c / 2)

{

mas4[i] = i;

}

else

{

mas4[i] = c - i;

}

}

starttime = clock();

qs(mas4, 0, c - 1);

endtime = clock();

t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;

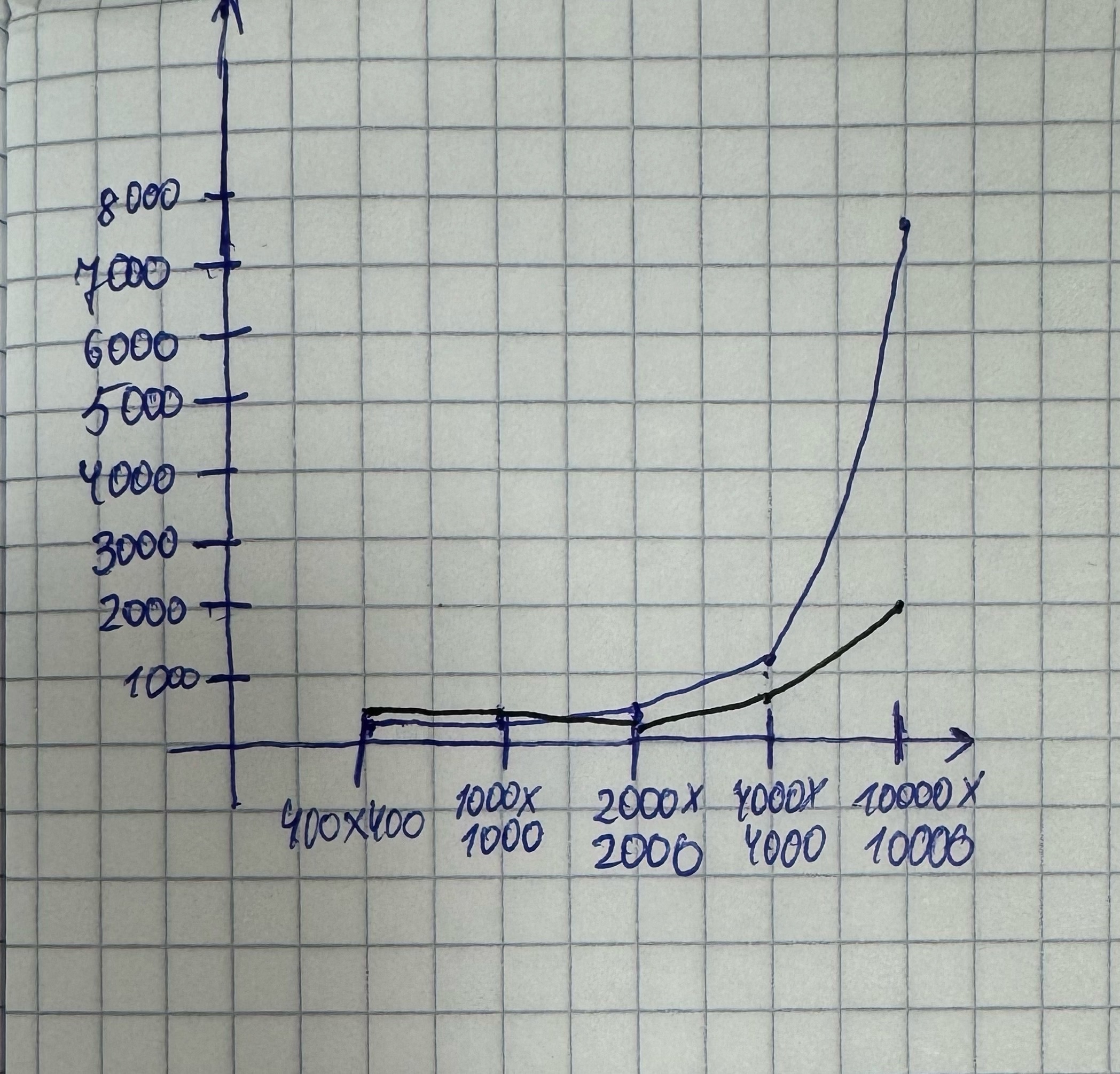
printf("qs4: %f\n", t);

}

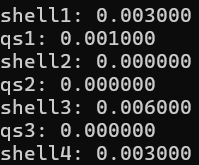
**Результат:**

**1)**





**2)**

****

**Вывод:**

В ходе лабораторной работы мы изучили работу с библиотекой time.h в языке Си и закрепили на практике, также сравнили время работы стандартной библиотеки quick sort c написанной вручную функцией quick sort и shell, получили следующие результаты стандартная библиотека быстрее написанных вручную функций.