**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“

Выполнил студент группы 23ВВВ1 :

Головушкина А.Б.

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.

100x100 = 0.03

200x200 = 0.028

400х400 = 0.232

1000х1000 = 4.598

2000х2000 = 51,45

4000х4000 = 952.607

10000x10000 = 9115,234

**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)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <Windows.h>

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

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

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

// объявляем переменные для определения времени выполнения

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

int n = 1000;

int maxx = 100;

int minn = -100;

int\*\* a, \*\* b, \*\* c, elem\_c;

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

printf("Введите кол-во элементов в матрице: ");

scanf("%d", &n);

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

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

{

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

}

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

{

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

{

a[i][j] = minn + rand() % (maxx - minn + 1);

}

}

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

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

{

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

}

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

{

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

{

b[i][j] = minn + rand() % (maxx - minn + 1);

}

}

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

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

{

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

}

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

{

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

{

c[i][j] = minn + rand() % (maxx - minn + 1);

}

}

while (i < n)

{

while (j < n)

{

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 < n)

{

while (j < n)

{

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

j++;

}

i++;

}

clock\_t start = clock();

for (i = 0;i < n;i++) // O(n) \* O(n^2) = O(n^3)/10^8

{

for (j = 0;j < n;j++) // O(n)\*O(n) = O(n^2)

{

// O(max(1,n)) = O(n)

elem\_c = 0; // O(1)

for (r = 0;r < n;r++) //O(n\*1) = O(n)

{

elem\_c = elem\_c + a[i][r] \* b[r][j];// O(1)

c[i][j] = elem\_c; // O(1)

}

}

printf("%d\n", i);

}

clock\_t end = clock();

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

printf("seconds = %f", seconds);

scanf("%d", &st);

return(0);

}

**2)**

**#define \_CRT\_SECURE\_NO\_WARNINGS**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**#include <Windows.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);**

**}**

**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**

**}**

**int main(void)**

**{**

**SetConsoleCP(1251);**

**SetConsoleOutputCP(1251);**

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

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

**clock\_t start, end;**

**int i = 0, j = 0, r, min = -1000, max = 1000, c = 10000;**

**int mas1[10000], mas2[10000], mas3[10000];**

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

**{**

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

**mas2[i] = mas1[i];**

**mas3[i] = mas1[i];**

**}**

**clock\_t starttime, endtime;**

**// рандом**

**starttime = clock();**

**shell(mas1, c);**

**endtime = clock();**

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

**printf("Сортировка Шелла на случайно наборе: %f\n", t);**

**starttime = clock();**

**qs(mas2,0,c-1);**

**endtime = clock();**

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

**printf("Быстрая сортировка на случайно наборе: %f\n", t);**

**starttime = clock();**

**qsort(mas3, c, sizeof(int), compare);**

**endtime = clock();**

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

**printf("Функция qsort() на случайно наборе: %f\n", t);**

**// возрастающая**

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

**{**

**mas1[i] = i;**

**mas2[i] = i;**

**mas3[i] = i;**

**}**

**starttime = clock();**

**shell(mas1, c);**

**endtime = clock();**

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

**printf("Сортировка Шелла на возрастающей последовательности чисел: %f\n", t);**

**starttime = clock();**

**qs(mas2, 0, c - 1);**

**endtime = clock();**

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

**printf("Быстрая сортировка на возрастающей последовательности чисел: %f\n", t);**

**starttime = clock();**

**qsort(mas3, c, sizeof(int), compare);**

**endtime = clock();**

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

**printf("Функция qsort() на возрастающей последовательности чисел: %f\n", t);**

**// Убывающая**

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

**{**

**if (i <= c / 2)**

**{**

**mas1[i] = i;**

**}**

**else**

**{**

**mas1[i] = c - i;**

**}**

**mas2[i] = mas1[i];**

**mas3[i] = mas1[i];**

**}**

**starttime = clock();**

**shell(mas1, c);**

**endtime = clock();**

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

**printf("Сортировка Шелла на убывающей последовательности чисел: %f\n", t);**

**starttime = clock();**

**qs(mas2, 0, c - 1);**

**endtime = clock();**

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

**printf("Быстрая сортировка на убывающей последовательности чисел: %f\n", t);**

**starttime = clock();**

**qsort(mas3, c, sizeof(int), compare);**

**endtime = clock();**

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

**printf("Функция qsort() на убывающей последовательности чисел: %f\n", t);**

**// возрастающая - убывающая**

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

**{**

**mas1[i] = c - i;**

**mas2[i] = c - i;**

**mas3[i] = c - i;**

**}**

**starttime = clock();**

**shell(mas1, c);**

**endtime = clock();**

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

**printf("Сортировка Шелла на возраст - убыв последовательности чисел: %f\n", t);**

**starttime = clock();**

**qs(mas2, 0, c - 1);**

**endtime = clock();**

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

**printf("Быстрая сортировка на возраст - убыв последовательности чисел: %f\n", t);**

**starttime = clock();**

**qsort(mas3, c, sizeof(int), compare);**

**endtime = clock();**

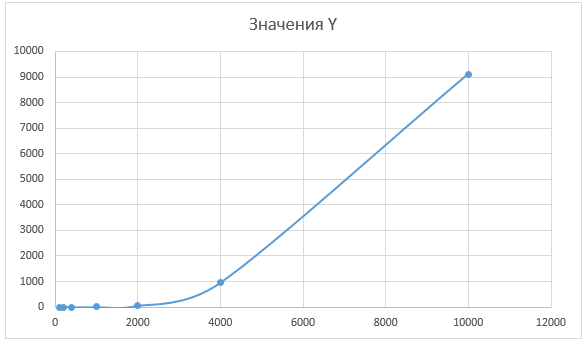
**t = (float)(endtime - starttime) / (float)CLOCKS\_PER\_SEC;**

**printf("Функция qsort() на возраст - убыв последовательности чисел: %f\n", t);**

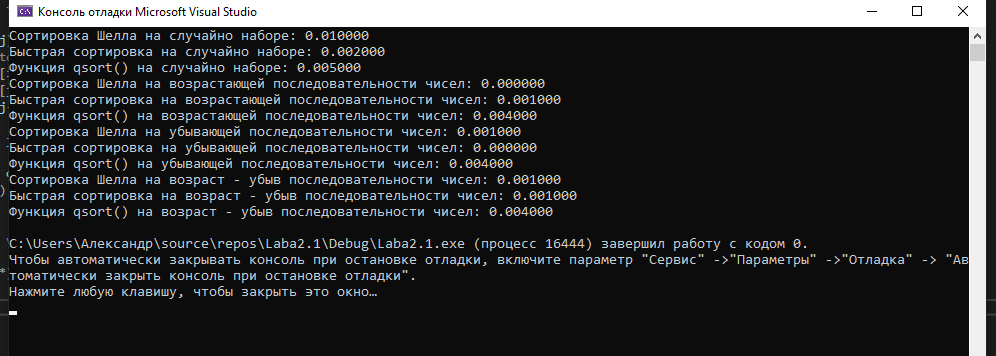
**}**

**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 медленнее написанных вручную функций.