Пензенский государственный университет  
Кафедра «Вычислительной техники»

**Отчет**по лабораторной работе №2  
по курсу "ЛиОАвИЗ"

**Выполнили студенты группы 20ВВ1:**

Вяльмисов М.О.

Кирюшин А.

**Приняла:**

Юрова О.В.

Пенза 2021

**Лабораторные задания:**

**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.

**Решение задач:**

**1 задание.**

Алгоритм перемножения матриц имеет сложность O(n^3)

Тесты:

|  |  |
| --- | --- |
| Размер | Время выполнения (с) |
| 50 | 0.001 |
| 100 | 0.003 |
| 200 | 0.027 |
| 300 | 0.087 |
| 450 | 0.358 |
| 600 | 0.936 |
| 1200 | 6.959 |
| 2400 | 77.474 |
|  |  |

На графике парабола => сложность алгоритма O(n^3)

**2 задание.**

Тесты с кол-вом элементов 100 000.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Алгоритм Шелла (с) | Алгоритм быстрой сортировки(с) | Функция qsort (с) |
| Случайные числа | 0.850 | 0.015 | 0.067 |
| Возрастающие числа | 0.002 | 0.006 | 0.042 |
| Убывающие числа | 1.775 | 0.006 | 0.041 |
| ½ убывает и ½ возрастает | 1.069 | 0.006 | 0.039 |
|  |  |  |  |

**Листинг:**

**1.**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define SIZE 1000

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;

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

start = clock();

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

while (i < SIZE)

{

while (j < SIZE)

{

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 < SIZE)

{

while (j < SIZE)

{

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

j++;

}

i++;

}

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

{

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

{

elem\_c = 0;

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

{

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

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

}

}

}

end = clock();

printf("%d", (end - start) / CLOCKS\_PER\_SEC);

getchar();

return(0);

}

**2.**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define SIZE 10

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

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

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

void GenerateArray(int\* arr, int size);

void PrintArray(int\* arr, int size);

int main(void)

{

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

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

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

int arr[SIZE];

GenerateArray(arr, SIZE);

start = clock();

shell(arr, SIZE);

end = clock();

printf("\n Shell algorithm: %f \n", difftime(end, start)/ CLOCKS\_PER\_SEC);

printf("\n \n");

GenerateArray(arr, SIZE);

start = clock();

qs(arr,0, SIZE - 1);

end = clock();

printf("\n Fast Sort algorithm: %f \n", difftime(end, start) / CLOCKS\_PER\_SEC);

printf("\n \n");

GenerateArray(arr, SIZE);

start = clock();

qsort(arr, SIZE, sizeof(int), compare);

end = clock();

printf("\n qsort function algorithm: %f \n", difftime(end, start) / CLOCKS\_PER\_SEC);

printf("\n \n");

getchar();

return(0);

}

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

}

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 GenerateArray(int\* arr, int size) {

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

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

arr[i] = rand();

}

}

void PrintArray(int\* arr, int size) {

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

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

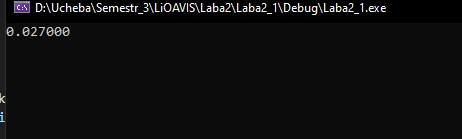
printf("%d ", arr[i] );

}

}

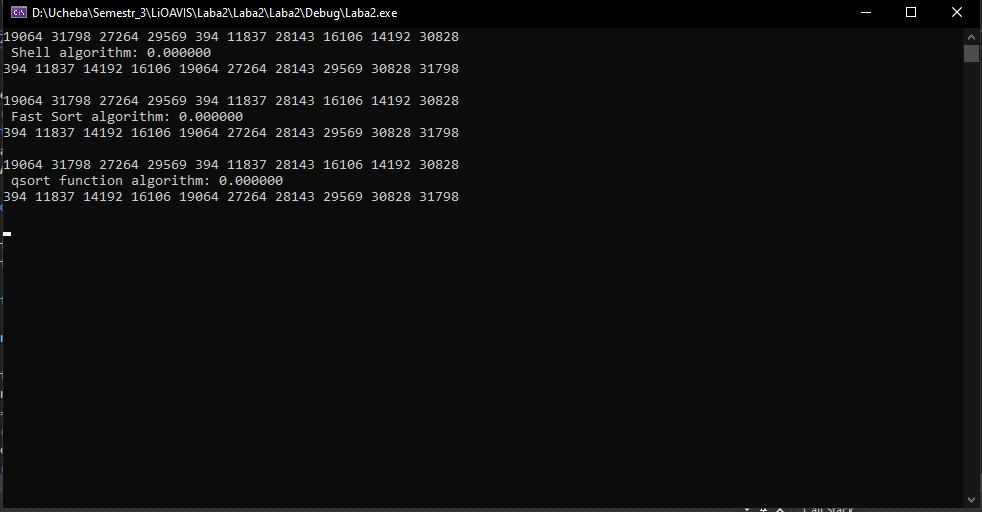
**Пример работы программ:**

**1.**

****

**2.**

**Сортировка массива 3 алгоритмами.**

****