Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Оценка времени выполнения программ»

Выполнили:

студенты группы 20ВВ2

Киреев Б.П.

Верховский М.В.

Лукин В.Д.

Приняли:

д.т.н. Митрохин М.А.

к.т.н. доцент Юрова О.В.

Пенза 2021

Название: Оценка времени выполнения программ.

Цель работы: Научиться оценивать время выполнения программ и их сложность.

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.

**Задание 1:**

Листинг:

ФайлLab2\_1.cpp

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int main(void)

{

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

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

clock\_tstart = clock(); // объявляем переменные для определения времени выполнения

inti = 0, j = 0, r;

int a[100][100], b[100][100], c[100][100], elem\_c;

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

while (i < 100)

{

while (j < 100)

{

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 < 100)

{

while (j < 100)

{

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

j++;

}

i++;

}

for (i = 0; i< 100; i++)

{

for (j = 0; j < 100; j++)

{

elem\_c = 0;

for (r = 0; r < 100; r++)

{

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

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

}

}

}

for (i = 0; i< 100; i++) {

for (j = 0; j < 100; j++) {

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

}

printf("\n");

}

clock\_t end = clock();

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

printf("Vremya: %f secunds\n", seconds);

return(0);

}

Пояснительный текст к программе:

Данная программа оценивает время выполнения программы, выполняющей перемножение матриц с помощью функции библиотеки time.h.

Результаты работы программы:

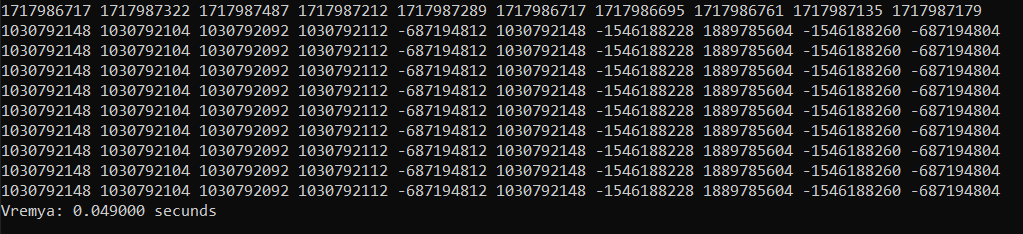


Рисунок №1 – Работа программы из задания №1: для размера матриц 100-0,049000 сек.

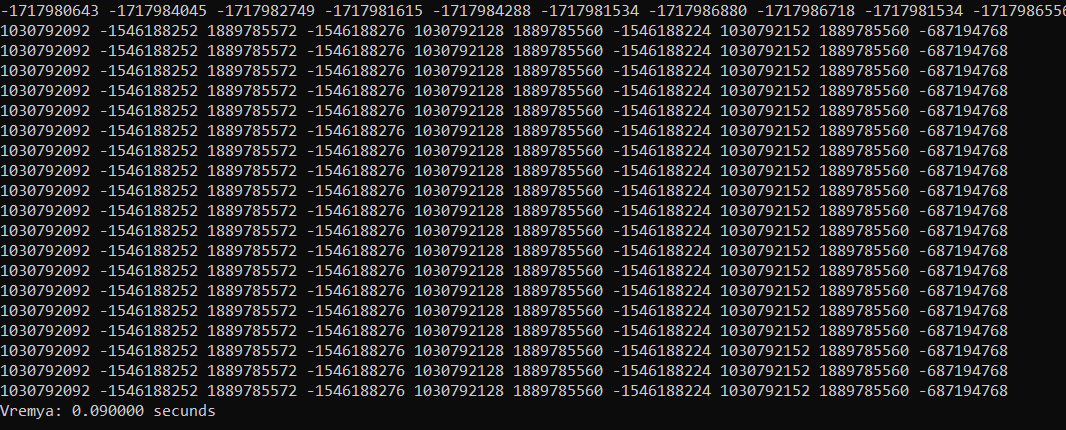


Рисунок №2 – Работа программы из задания №1: для размера матриц 200 - 0,090000сек.

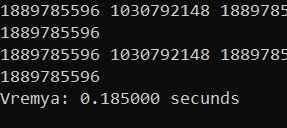


Рисунок №3 – Работа программы из задания №1: для размера матриц 400 - 0,185000сек.

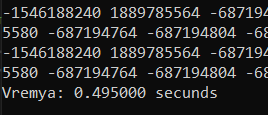


Рисунок №4 – Работа программы из задания №1: для размера матриц 1000 - 0,495000сек.

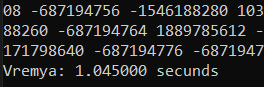


Рисунок №5 – Работа программы из задания №1: для размера матриц 2000 - 1,045000сек.

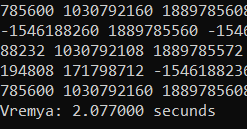


Рисунок №6 – Работа программы из задания №1: для размера матриц 4000 - 2,077000сек.

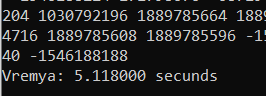


Рисунок №7 – Работа программы из задания №1: для размера матриц 10000 - 5,118000 сек.

Рисунок №8 – График зависимости времени выполнения программы от размера матриц.

**Задание 2:**

Листинг:

ФайлLab2\_2.cpp

#include <iostream>

#include <time.h>

using namespace std;

int Mass[20] = { 40,23,3,6,84,12,1,78,79,34,23,14,45,76,78,98,73,12,31,26 }, chis = 20;

int time2, time3;

int Mass1[20] = { 40,23,3,6,84,12,1,78,79,34,23,14,45,76,78,98,73,12,31,26 };

//Сортировка Шелла по возрастанию

void shell(int\* items, int count) { //int count - к-во эл-тов в массиве, int\* items - указатель на нулевой элемент массива (на начало массива)

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;

}

}

}

}

//Быстрая сортировка

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

int i, j;

int x, y;

i = left; j = right;

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 main()

{

shell(Mass, chis);

time2 = clock();

cout << "Work time 1: ";

cout.precision(20);

cout << (float)time2 / CLOCKS\_PER\_SEC << endl;

qs(Mass1, 0, chis - 1);

time3 = clock();

cout << "Work time 2: " << ((float)time3 - (float)time2) / CLOCKS\_PER\_SEC << endl;

return 0;

}

Пояснительный текст к программе:

Данная программа оценивает время выполнения функций быстрой сортировки и сортировки Шелла с помощью функции библиотеки time.h.

Результаты работы программы:

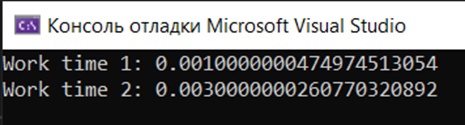


Рисунок №9 – Работа программы из задания №1: время работы для случайного набора данных

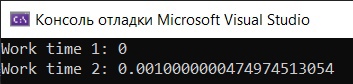


Рисунок №10 – Работа программы из задания №2: время работы для возрастающей последовательности чисел

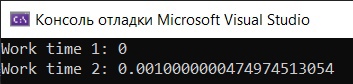


Рисунок №11 – Работа программы из задания №3: время работы для убывающей последовательности чисел

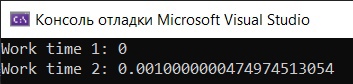


Рисунок №12 – Работа программы из задания №4: время работы для ½ убывающей и ½ возрастающей последовательности чисел

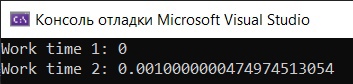


Рисунок №13 – Работа программы из задания №5: время работы для ½ возрастающей и ½ убывающей последовательности чисел

Вывод: При выполнении данной лабораторной работы, мы смогли научиться оценивать время выполнения программ и их сложность.