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:

Файл Lab2\_1.cpp

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string>

#include <fstream>

#include <iostream>

#include <conio.h>

using namespace std;

int main(void)

{

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

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

setlocale(LC\_ALL, "Rus");

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

//clock() - это команда, возвращающая кол-во тиков за определенный промежуток программы

int i = 0, j = 0, r;

int a[20][20], b[20][20], c[20][20], elem\_c;

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

while (i < 20)

{

while (j < 20)

{

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

cout<<a[i][j]<<" ";

j++;

}

i++;

j = 0;

cout<<endl;

}

i = 0; j = 0;

cout<<endl;

while (i < 20)

{

while (j < 20)

{

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

cout<<b[i][j]<<" ";

j++;

}

i++;

j = 0;

cout<<endl;

}

//Перемножение матриц

clock\_t start2 = clock(); // объявляем переменные для определения времени выполнения

cout<<endl;

getch();

for (i = 0; i < 20; i++)

{

for (j = 0; j < 20; j++)

{

elem\_c = 0;

for (r = 0; r < 20; r++)

{

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

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

}

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

}

printf("\n");

}

clock\_t end = clock();//вторая временная метка

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;//изначально эта операция целочисленная, но с помощью (double) мы делаем её с плавающей точкой

double seconds2 = (double)(end - start2) / CLOCKS\_PER\_SEC;

printf("Время: %f сек\n", seconds);

printf("Время(Умножение): %f сек\n", seconds2);

getch();

return(0);

}

Пояснительный текст к программе:

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

Результаты работы программы:

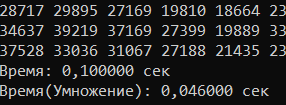


Рисунок №1 – Работа программы из задания №1 (для размера матриц 100):

Время выполнения программы: 0,100000 сек;

Время на умножение: 0,046000 сек.

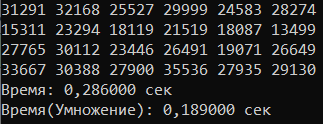


Рисунок №2 – Работа программы из задания №1 (для размера матриц 200):

Время выполнения программы: 0,286000 сек;

Время на умножение: 0,189000 сек.

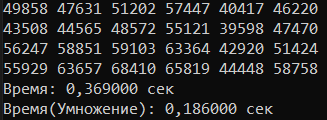


Рисунок №3 – Работа программы из задания №1 (для размера матриц 400):

Время выполнения программы: 0,369000 сек;

Время на умножение: 0,186000 сек.

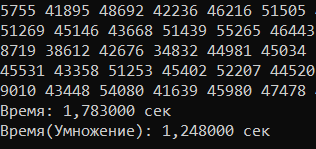


Рисунок №4 – Работа программы из задания №1 (для размера матриц 1000):

Время выполнения программы: 1,783000 сек;

Время на умножение: 1,248000 сек.

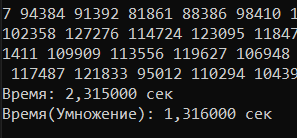


Рисунок №5 – Работа программы из задания №1 (для размера матриц 2000):

Время выполнения программы: 2,315000 сек;

Время на умножение: 1,316000 сек.

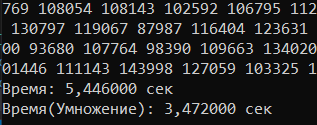


Рисунок №6 – Работа программы из задания №1 (для размера матриц 4000):

Время выполнения программы: 5,446000 сек;

Время на умножение: 3,472000 сек.

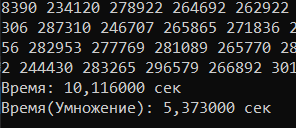


Рисунок №7 – Работа программы из задания №1 (для размера матриц 10000):

Время выполнения программы: 10,116000 сек;

Время на умножение: 5,373000 сек.

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

Рисунок №9 – График зависимости времени выполнения умножения от размера матриц.

**Задание 2:**

Листинг:

ФайлLab2\_2.cpp

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string>

#include <fstream>

#include <iostream>

#include <windows.h>

#include <conio.h>

using namespace std;

int Mass[20], chis = 20;

int time2, time3;

FILE\* ANTON;

int a;

//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()

{

setlocale(LC\_ALL, "Russian");

cout << "1)Рандомный " << endl;

cout << "2)Возрастающий " << endl;

cout << "3)Убывающий " << endl;

cout << "4)1/2 возрастающий, 1/2 убывающий " << endl;

cout << "5)1/2 убывающий, 1/2 возрастающий " << endl;

cout << "Выберите тип массива: ";

cin >> a;

ANTON = fopen("file.txt", "r");

fseek(ANTON, 59\*(a-1), SEEK\_SET);//60 - среднее к-во символов в строке

for (int i = 0; i < 20; i++) {

fscanf(ANTON, "%d", &Mass[i]);

cout << Mass[i] << " ";

}

shell(Mass, chis);

cout << "\n";

time2 = clock();

cout << "Work time 1: ";

cout.precision(20);//вывод доп знаков

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

qs(Mass, 0, chis - 1);//параметры: указатель на начало массива, индекс эл-та массива, с которого наинаем сортировку, индекс эл-та массива, на котором заканчиваем сортировку

time3 = clock();

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

fclose(ANTON);

getch();

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:

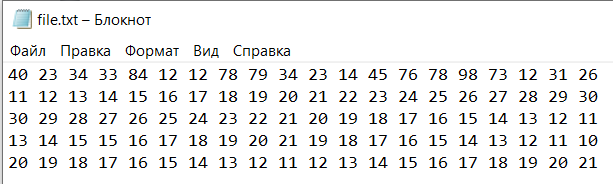


Рисунок №9 – Файл с данными

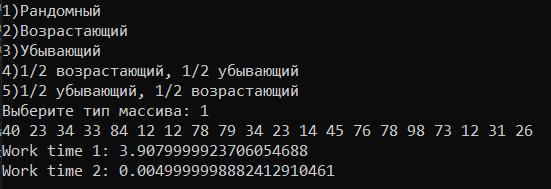


Рисунок №10 – Работа программы из задания №1: время работы для случайного набора данных

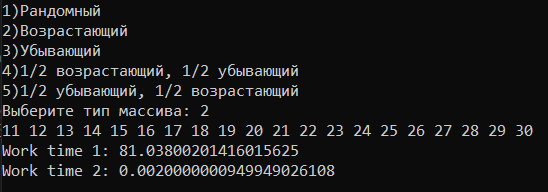


Рисунок №11 – Работа программы из задания №2: время работы для возрастающей последовательности чисел

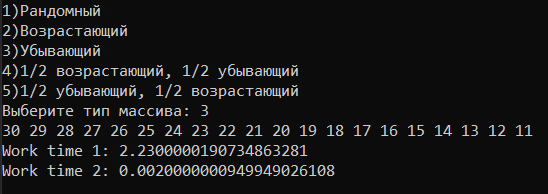


Рисунок №12 – Работа программы из задания №3: время работы для убывающей последовательности чисел

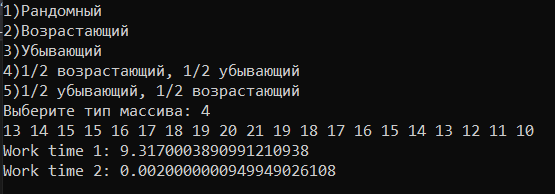


Рисунок №13 – Работа программы из задания №4: время работы для ½ возрастающей и ½ убывающей последовательности чисел

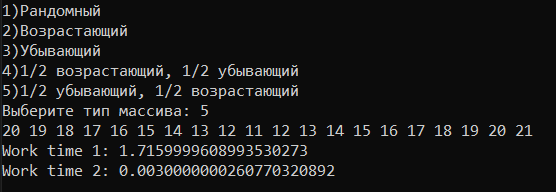


Рисунок №14 – Работа программы из задания №5: время работы для ½ убывающей и ½ возрастающей последовательности чисел

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