Пензенский государственный университет

Кафедра ”Вычислительной техники ”

**Отчет**

по лабораторной работе №2

по дисциплине: “ Логика и основы алгоритмизации в инженерных задачах ”

## по теме: “Оценка времени выполнения программы”

## *Выполнил студент группы 19ВВ1:*

## Гусев В

## *Приняли:*

## Митрохин М.А

ПЕНЗА 2020

##### Общие сведения.

Для оценки времени выполнения программ языка Си или их частей могут использоваться средства, предоставляемые библиотекой time.h. Данная библиотека содержит описания типов и прототипы функций для работы с датой и временем.

Типы данных:

1. clock\_t - возвращается функцией clock(). Обычно определён как int или long int.

2. time\_t - возвращается функцией time(). Обычно определён как int или long int.

3. struct tm - нелинейное, дискретное календарное представление времени.

Основные функции:

1. clock\_t clock(void) - возвращает время, измеряемое процессором в тактах от начала выполнения программы, или −1, если оно не известно. Пересчет этого времени в секунды выполняется по формуле: clock() / CLOCKS\_PER\_SEC где CLOCKS\_PER\_SEC – константа, определяющая количество тактов системных часов в секунду.

2. time\_t time(time\_t \*tp)

Возвращает текущее календарное время или −1, если это время не известно. Если указатель tp не равен NULL, то возвращаемое значение записывается также и в \*tp.

3. double difftime(time\_t time2,time\_t time1)

Возвращает разность time2-time1, выраженную в секундах.

###### Задание 1)

Вычислить порядок сложности программы (О-символику).

while(i<n){

while(i<n){

a[i][j] = rand()%100+1;

j++;

}

i++;

}

Сложность O(n^2)

while(i<n){

while(i<n){

b[i][j] = rand()%100+1;

j++;

}

i++;

}

Сложность O(n^2)

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

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

elem\_c = 0;

for(int r =0;r<n;r++){

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

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

}

}

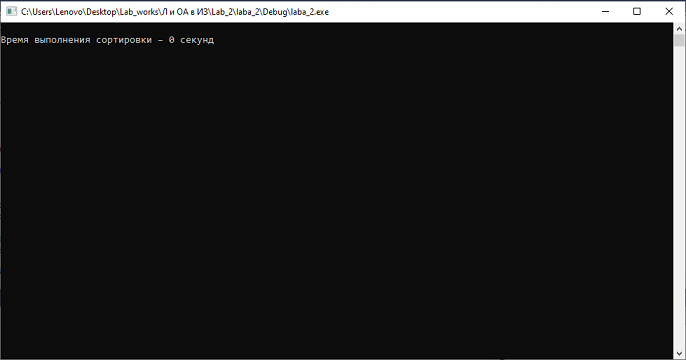
}

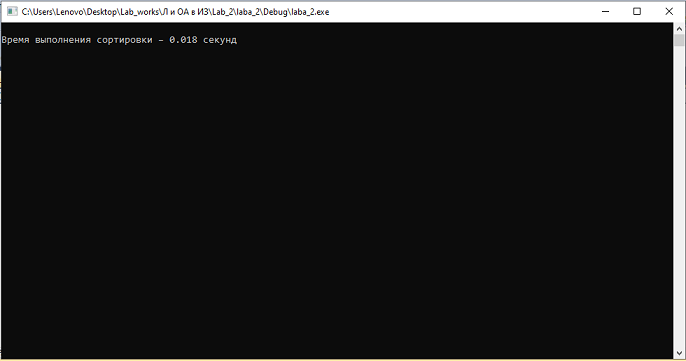
Сложность O(n^3)

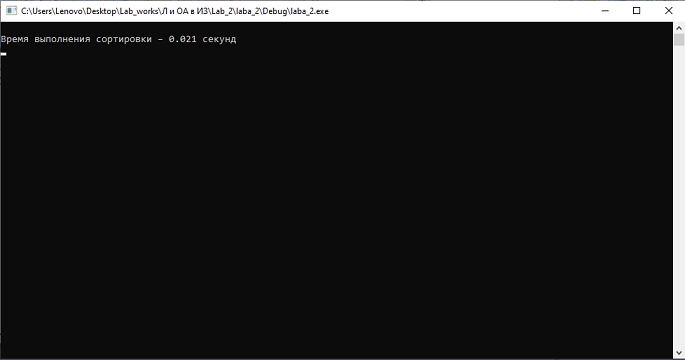
###### Задание 2)

Оценить время выполнения программы и кода, выполняющего перемножение

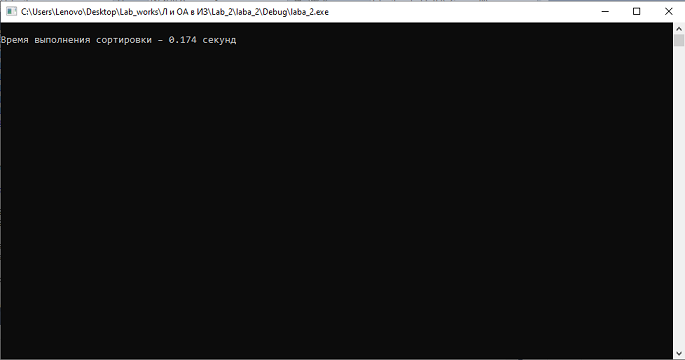
матриц, используя функции библиотеки time.h для матриц размерами от 10, 100 , 150, 200, 290.







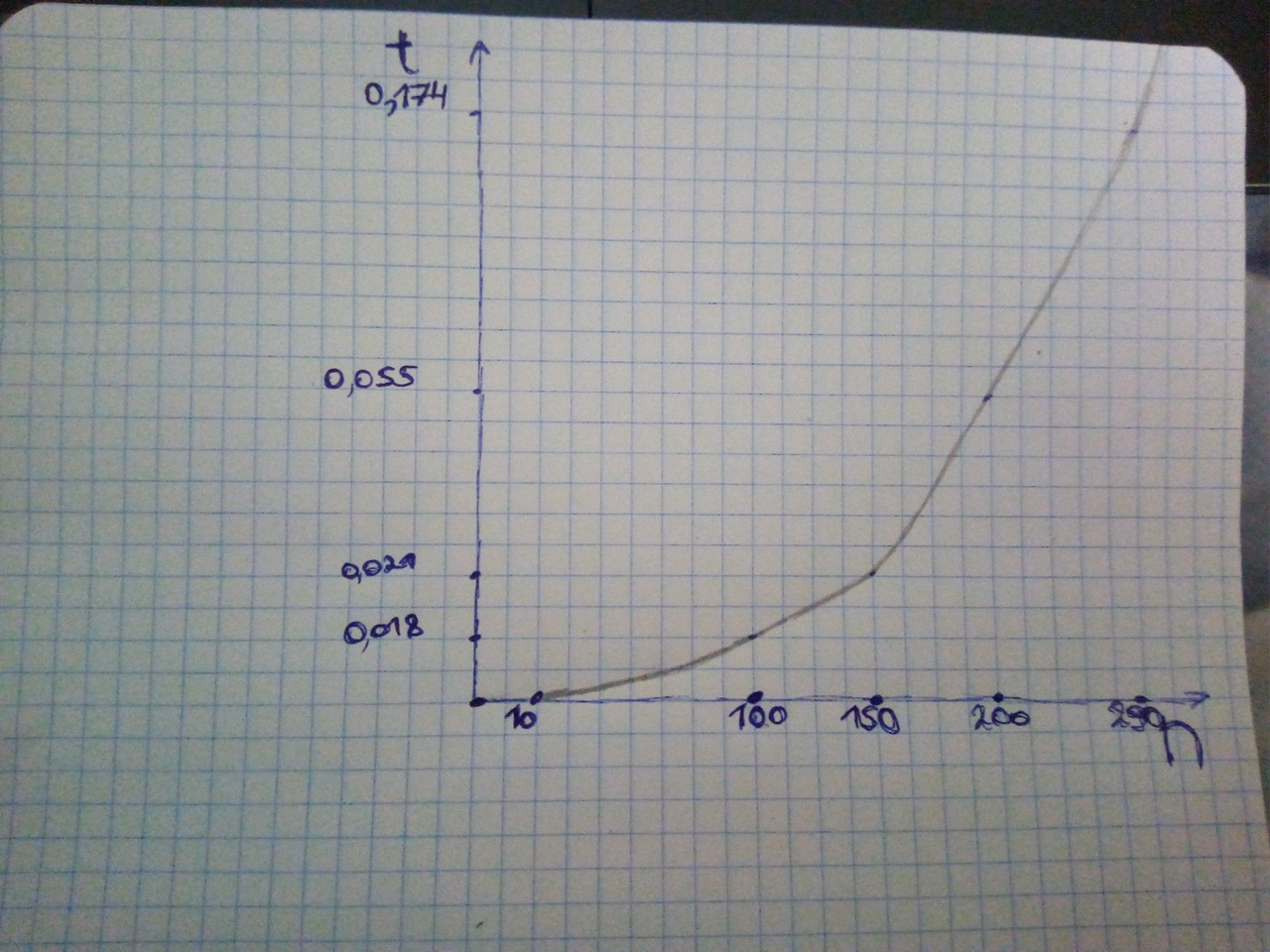




###### Задание 3)

Построить график зависимости времени выполнения программы от размера матриц

и сравнить полученный результат с теоретической оценкой.



Задание 2)

// lab\_2.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <stdio.h>

#include <locale>

#include <malloc.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

void ShellNumber(int arr, int randomNumber[]);

void qs(int randomNumber[], int left, int right);

void easy(int randomNumber[], int arr);

int main()

{

setlocale(LC\_ALL, "Rus");

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

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

srand(time(NULL));

system("chcp 1251");

system("cls");

int \*randomNumber;

int arr;

clock\_t start, end;

clock\_t start1, end1;

clock\_t start2, end2;

printf("Размер массива:");

scanf("%d", &arr);

//Выделяем память

randomNumber = (int\*)malloc(arr \* sizeof(int));

FILE \*file = fopen("text.txt", "a");

fprintf(file, "\nВыведем время выполнения программы массив с рандомными числами\n\n");

//Массив с рандомными числами

printf("\n\n Вводим возростающий за тем убывающий массив\n");

for(int i = 1; i<arr/2; i++){

randomNumber[i]=i;

printf("%d\n", randomNumber[i]);

}

for(int i = arr/2; i>0; i--){

randomNumber[i]=i;

printf("%d\n", randomNumber[i]);

}

start = clock();

ShellNumber(arr, randomNumber);

printf("Сортировка массива :\n\n");

printf("Метод Шерли:\n ");

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

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

fprintf(file, "%d ", randomNumber[i]);

}

end = clock();

printf("\n\nВыведем время выполнения программы %lf\n\n", (double)(end - start)/CLOCKS\_PER\_SEC);

fprintf(file, "\n\nВыведем время выполнения программы %lf\n\n", (double)(end - start)/CLOCKS\_PER\_SEC);

start1 = clock();

qs(randomNumber, 0, arr-1);

printf("Метод qs: \n ");

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

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

fprintf(file, "%d ", randomNumber[i]);

}

end1 = clock();

printf("\n\nВыведем время выполнения программы %lf\n\n", (double)(end1 - start1)/CLOCKS\_PER\_SEC);

fprintf(file, "\n\nВыведем время выполнения программы %lf\n\n", (double)(end1 - start1)/CLOCKS\_PER\_SEC);

start2 = clock();

easy(randomNumber, arr);

printf("Быстрая сортировка:\n");

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

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

fprintf(file,"%d ", randomNumber[i]);

}

end2 = clock();

printf("\n\nВыведем время выполнения программы %lf\n\n", (double)(end2 - start2)/CLOCKS\_PER\_SEC);

fprintf(file, "\n\nВыведем время выполнения программы %lf\n\n", (double)(end2 - start2)/CLOCKS\_PER\_SEC);

fprintf(file, "------------------------------------------------------------------------------------------------------------------------------------------------\n");

/\*

printf("\n\nВводим возрастающую последовательнусть \n");

for(int i=1; i<=arr; i++){

randomNumber[i]=i;

printf("%d\n", randomNumber[i]);

}

printf("\nВводим массив с рандомными числами\n");

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

printf("%d\n", randomNumber[i] = rand() %100);

}

printf("\n\n Вводим убывающую последовательность\n");

for(int i=arr;i>0;i--){

randomNumber[i]=i;

printf("%d\n", randomNumber[i] );

}

\

}\*/

fclose(file);

\_getch();

free(randomNumber);

return 0;

}

// Сортировка Шелла

void ShellNumber(int arr, int randomNumber[])

{

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 < arr; k++) {

gap = a[k];

for(i=gap; i < arr; ++i) {

x = randomNumber[i];

for(j=i-gap; (x < randomNumber[j]) && (j >= 0); j=j-gap)

randomNumber[j+gap] = randomNumber[j];

randomNumber[j+gap] = x;

}

}

}

// Сортировка qs

void qs(int randomNumber[], int left, int right) //вызов функции: qs(items, 0, count-1);

{

int i, j;

int x, y;

i = left; j = right;

/\* выбор компаранда \*/

x = randomNumber[(left+right)/2];

do {

while((randomNumber[i] < x) && (i < right)) i++;

while((x < randomNumber[j]) && (j > left)) j--;

if(i <= j) {

y = randomNumber[i];

randomNumber[i] = randomNumber[j];

randomNumber[j] = y;

i++; j--;

}

} while(i <= j);

if(left < j) qs(randomNumber, left, j);

if(i < right) qs(randomNumber, i, right);

}

void easy(int randomNumber[], int arr){

for(int i = 0 ; i < arr - 1; i++) {

// сравниваем два соседних элемента.

for(int j = 0 ; j < arr - i - 1 ; j++) {

if(randomNumber[j] > randomNumber[j+1]) {

// если они идут в неправильном порядке, то

// меняем их местами.

int tmp = randomNumber[j];

randomNumber[j] = randomNumber[j+1] ;

randomNumber[j+1] = tmp;

}

}

}

}

Задание 1

Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.

Выведем время выполнения программы массив с рандомными числами

39 183 107 121 187 43 159 167 153 72 186 155 27 84 9 177 104 155 85 77 6 38 17 142 125 169 156 60 46 116 25 116 76 173 102 9 162 147 11 53 134 86 147 112 12 149 183 23 61 112

Выведем время выполнения программы Быстрой сортировкой 2,373000

Выведем время выполнения программы сортировкой Шелли 1,901000

2 задание

Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.

Выведем время выполнения программы массив с возростающей последовательностью

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99

Выведем время выполнения программы Быстрой сортировкой 5,214000

Выведем время выполнения программы сортировкой Шелли 1,771000

3 задание

Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел**.**

Выведем время выполнения программы массив с убывающей последовательностью

50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Выведем время выполнения программы Быстрой сортировкой 3,744000

Выведем время выполнения программы сортировкой Шелли 1,675000

4 задание

Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

Выведем время выполнения программы массив с начала возростающей потом убывающей последовательностью

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Выведем время выполнения программы Быстрой сортировкой 3,327000

Выведем время выполнения программы сортировкой Шелли 1,287000