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;

}

}

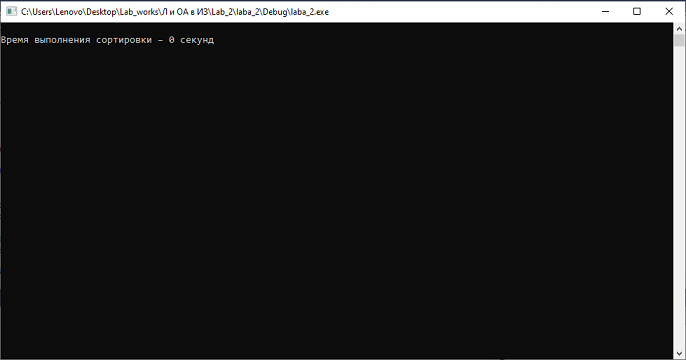
}

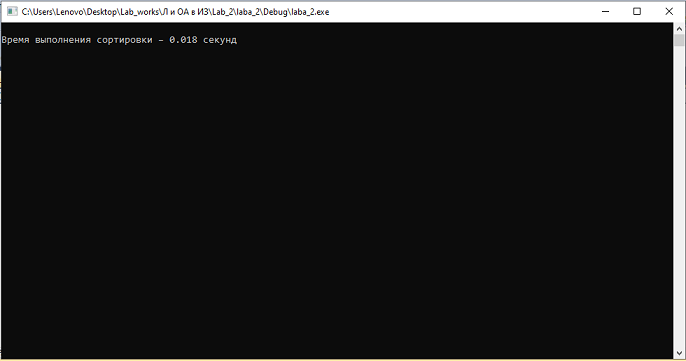
Сложность O(n^3)

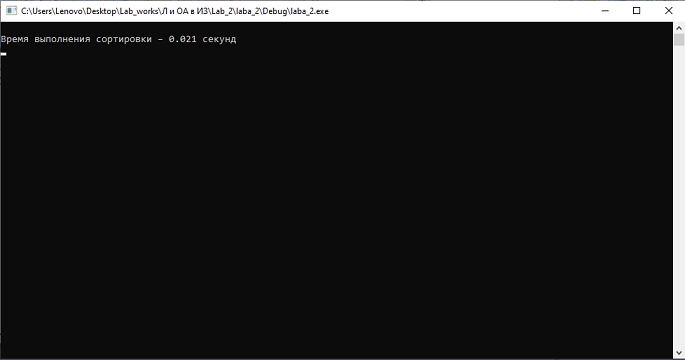
###### Задание 2)

Оценить время выполнения программы и кода, выполняющего перемножение

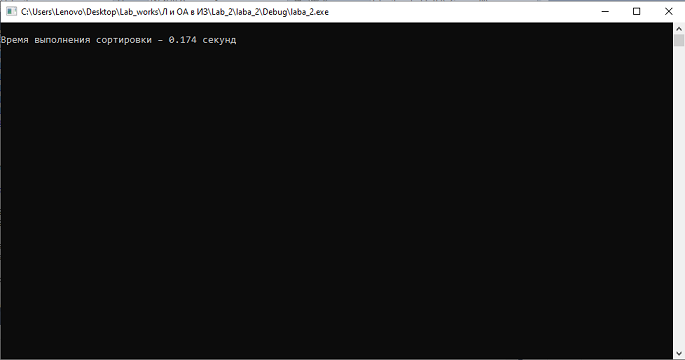
матриц, используя функции библиотеки time.h для матриц размерами от 10, 100 , 150, 200, 290.







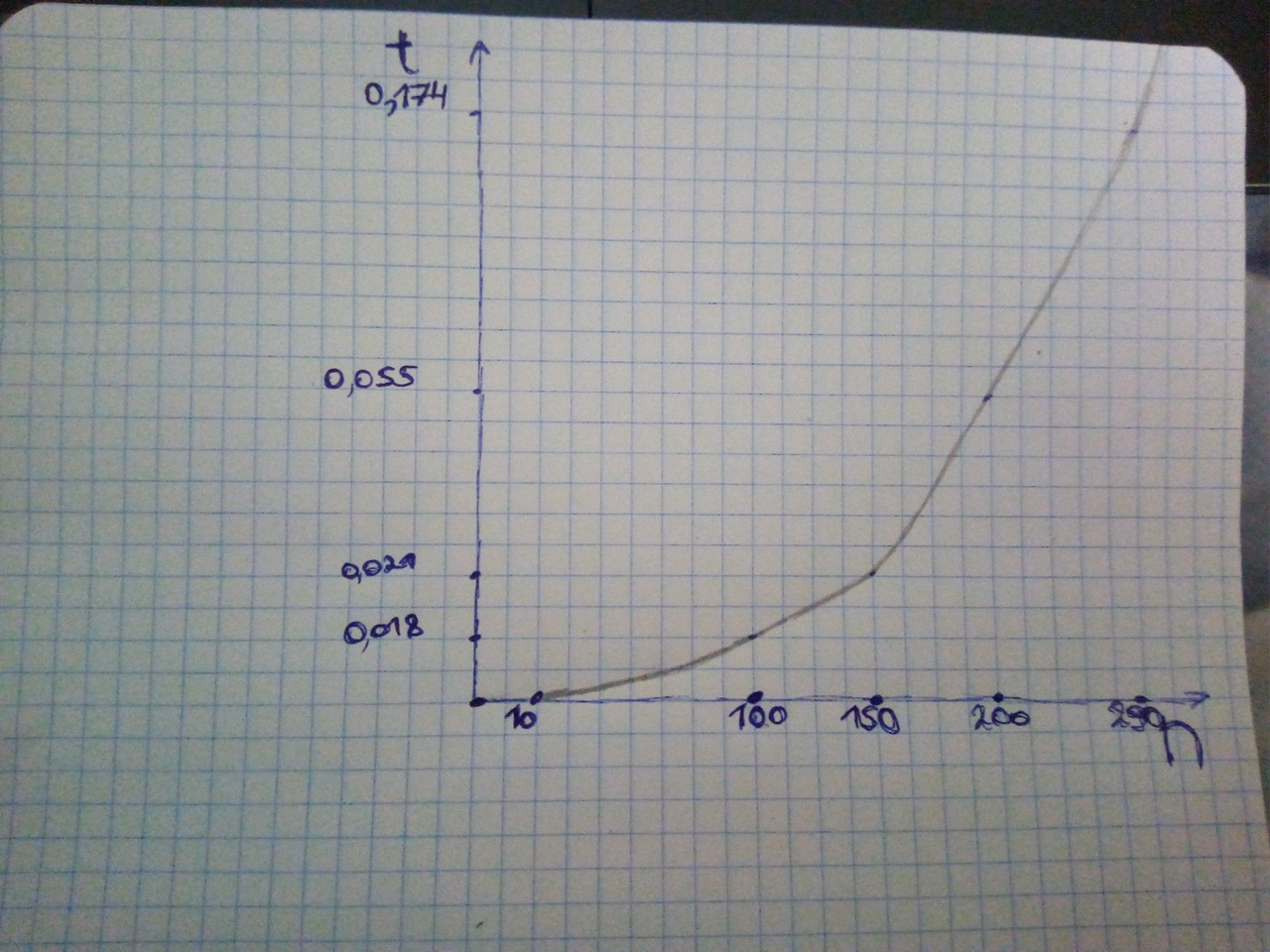




###### Задание 3)

Построить график зависимости времени выполнения программы от размера матриц

и сравнить полученный результат с теоретической оценкой.



Задание 2)

// test.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <conio.h>

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include<iomanip>

#include<cstdlib>

#include<string.h>

#include<Windows.h>

// Сортировка Шелла

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 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;

}

}

}

}

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 \_tmain(int argc, \_TCHAR\* argv[])

{

srand(time(NULL));

clock\_t start1, end1;

clock\_t start2, end2;

clock\_t start3, end3;

clock\_t start4, end4;

clock\_t start5, end5;

clock\_t start6, end6;

clock\_t start7, end7;

clock\_t start8, end8;

setlocale(LC\_ALL,"Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int \*arr; // указатель на массив

int i,n;

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

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

printf("Введите размер массива: ");

scanf("%d", &n);

// Выделение памяти

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

// Ввод элементов массива

srand(time(0));

for(int i=n; i>0; i--)

{

arr[i]=i;

fprintf(file, "%d\t",arr[i]);

}

printf("\n");

//====================// Определяем время быстрой сортировки на случайном массиве//====================//

start1 = clock();

qs(arr , 0 , n-1);

printf("Метод qs: \n ");

end1 = clock();

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

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

//====================// Определяем время сортировки Шелли на случайном массиве//====================//

start2 = clock();

shell(arr, n);

printf("Сортировка массива :\n\n");

printf("Метод Шерли:\n ");

end2 = clock();

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

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

start3 = clock();

easy(arr,n);

printf("Быстрая сортировка:\n ");

end3 = clock();

printf("\n\nВыведем время выполнения программы на случайном массиве %lf\n\n", (double)(end3 - start3)/CLOCKS\_PER\_SEC);

fprintf(file, "\n\nВыведем время выполнения программы на случайном массиве %lf\n\n", (double)(end3 - start3)/CLOCKS\_PER\_SEC);

//====================// Определяем время быстрой сортировки на возрастающем массиве//====================//

/\* fprintf(file, "Выведем время выполнения программы массив с возрастающими числами\n\n");

for(int i = 1; i<=n; i++)

{

arr[i]=i;

fprintf(file, "%d\t",arr[i]);

}

start3 = clock();

qs(arr , 0 , n-1);

end3 = clock();

printf("\n\nВыведем время выполнения программы на массиве с возрастающими числами %lf\n\n", (double)(end3 - start3)/CLOCKS\_PER\_SEC);

fprintf(file, "\n\nВыведем время выполнения QS на массиве с возрастающими числами %lf\n\n", (double)(end3 - start3)/CLOCKS\_PER\_SEC);

start4 = clock();

shell(arr, n);

printf("Сортировка массива :\n\n");

printf("Метод Шерли:\n ");

end4 = clock();

printf("\n\nВыведем время выполнения программы на массиве с возрастающими числами %lf\n\n", (double)(end4 - start4)/CLOCKS\_PER\_SEC);

fprintf(file, "\n\nВыведем время выполнения Shell на массиве с возрастающими числами %lf\n\n", (double)(end4 - start4)/CLOCKS\_PER\_SEC);\*/

//====================// Определяем время быстрой сортировки на убывающем массиве//====================//

/\* for(int i=n; i>0; i--)

{

arr[i]=i;

fprintf(file, "%d\t",arr[i]);

}

start5 = clock();

qs(arr , 0 , n-1);

end5 = clock();

printf("\n\nВыведем время выполнения программы на массиве с убывающими числами %lf\n\n", (double)(end5 - start5)/CLOCKS\_PER\_SEC);

fprintf(file, "\n\nВыведем время выполнения QS на массиве с убывающими числами %lf\n\n", (double)(end5 - start5)/CLOCKS\_PER\_SEC);

start6 = clock();

shell(arr, n);

printf("Сортировка массива :\n\n");

printf("Метод Шерли:\n ");

end6 = clock();

printf("\n\nВыведем время выполнения программы на массиве с убывающими числами %lf\n\n", (double)(end6 - start6)/CLOCKS\_PER\_SEC);

fprintf(file, "\n\nВыведем время выполнения Shell на массиве с убывающими числами %lf\n\n", (double)(end6 - start6)/CLOCKS\_PER\_SEC);

getch();

}

/\*Возрастающий массив

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

{

arr[i]=i;

fprintf(file, "%d\t",arr[i]);

}

\*/

/\*Убывающий массив

for(int i=n; i>0; i--)

{

arr[i]=i;

fprintf(file, "%d\t",arr[i]);

}

\*/

/\*РАндомный массив

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

{

arr[i]=i;

fprintf(file, "%d\t",arr[i]);

}

\*/

/\*

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

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

randomNumber[i]=i;

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

}

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

arr[i]=i;

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

}

\*/

getch();

}

}

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

Выведем время выполнения программы Быстрой сортировкой из библиотеки 0,530970

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

Выведем время выполнения программы Быстрой сортировкой из библиотеки 0,599400

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

Выведем время выполнения программы Быстрой сортировкой из библиотеки 1,032000

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

Выведем время выполнения программы Быстрой сортировкой из библиотеки 0,729000