Пензенский государственный университет

Кафедра "Вычислительной техники"

Отчёт

По лабораторной работе №2

По дисциплине: “[Логика и основы алгоритмизации в инженерных задачах](http://moodle.pnzgu.ru/course/view.php?id=48560)”.

На тему «Оценка времени выполнения программы.»

Выполнили студенты группы 21ВВ1.2:

Аракчеев Артём

Приняли:

Митрохин М.А.

Юрова О.А.

Пенза 2022

*Цель работы:*

Оценить времени выполнения программы.

*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, представляющемсобой 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.

1)Порядок сложности (*О* -Символику)

#include "stdafx.h"

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<locale>

int main(void)

{

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

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

setlocale (LC\_ALL,"Rus");

system("cls");

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

int i=0, j=0, r;

int m,xxx;

int\*\* A;

int\*\* B;

printf("Введите размер массива: ");

scanf("%d",&m);

A = (int\*\*)malloc(m \* sizeof(int));

B = (int\*\*)malloc(m \* sizeof(int));

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

{

A[i]=(int\*)malloc(m \* sizeof(int));

B[i]=(int\*)malloc(m \* sizeof(int));

/\*Сложность О (N^2)\*/

for(int j=0;j<m;j++)

{

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

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

}

}

int\*\* C;

C = (int\*\*)malloc(m \* sizeof(int));

unsigned int start\_time = clock();

/\*Сложность О (N^3)\*/

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

{

C[i]=(int\*)malloc(m \* sizeof(int));

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

{

xxx=0;

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

{

xxx=xxx+A[i][r]\*B[r][j];

C[i][j]=xxx;

}

}

}

unsigned int end\_time = clock();

double search\_time = (end\_time - start\_time)/1000.0;

printf("%lf\n\n",search\_time);

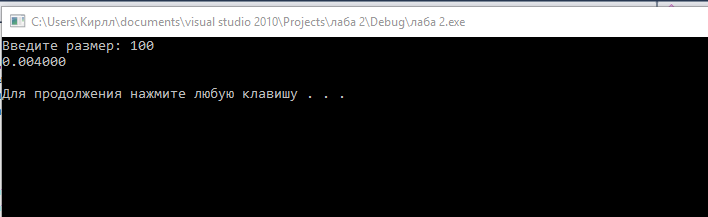
system("pause");

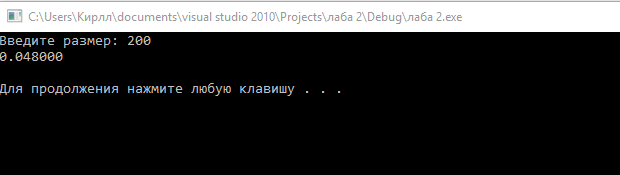
}

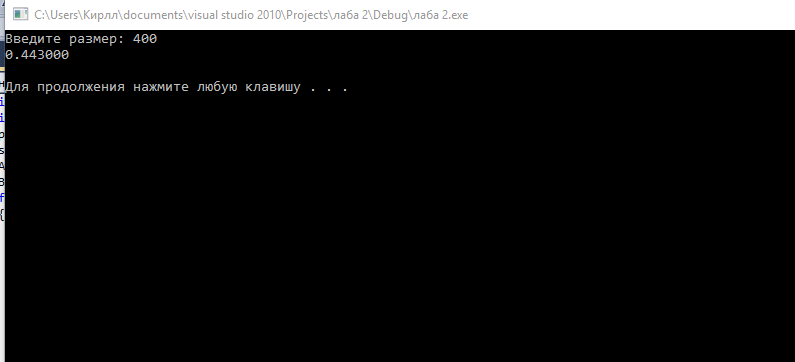
2)Оценить время выполнения программы и кода, выполняющего перемножение

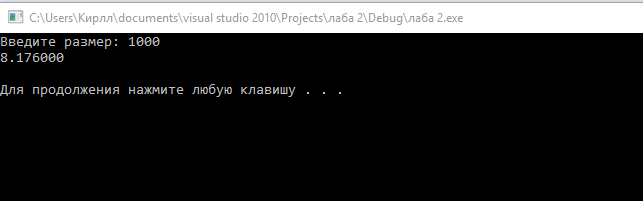
матриц, используя функции библиотеки time.h для матриц размерами от 100 ,200, 400,

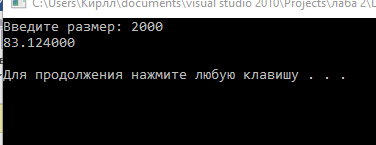
1000,2000,4000.

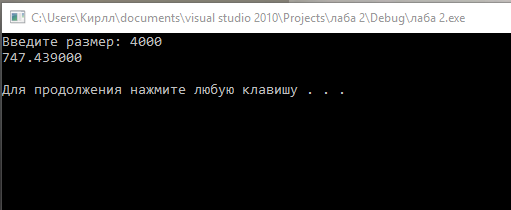




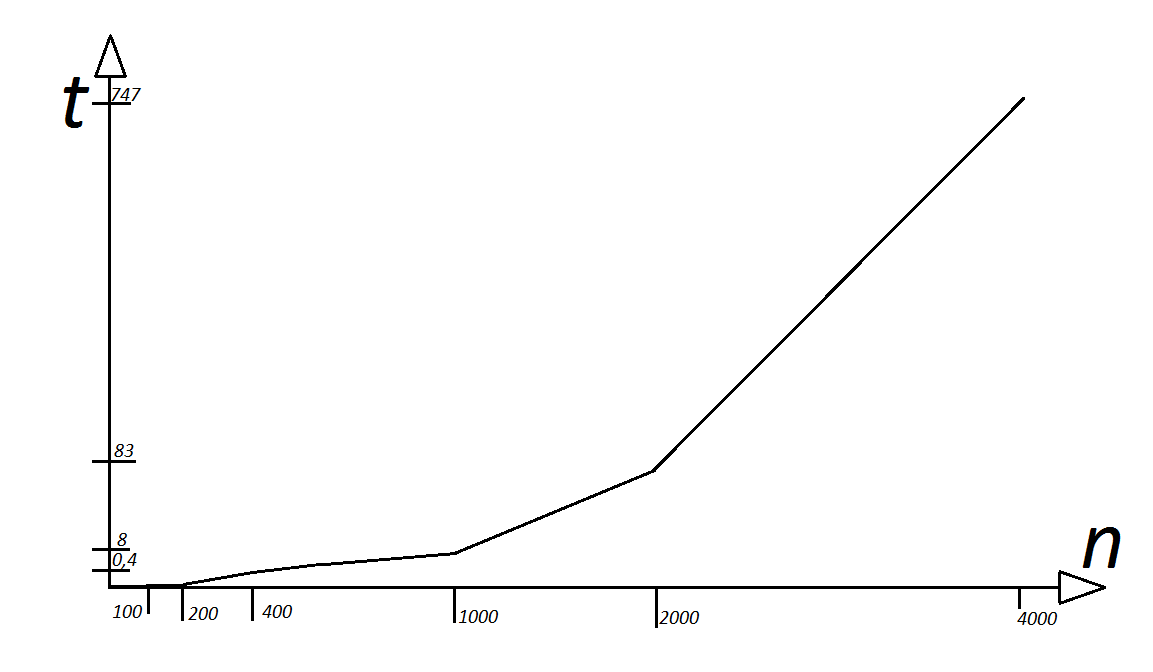






**

3) Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.



Задание 2:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale>

int main(void)

{

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

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

setlocale(LC\_ALL,"Rus");

system("cls");

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

int i=0, j=0, r;

int m,xxx;

int\*\* A;

int\*\* B;

printf("Введите размер массива: ");

scanf("%d",&m);

A = (int\*\*)malloc(m \* sizeof(int));

B = (int\*\*)malloc(m \* sizeof(int));

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

{

A[i]=(int\*)malloc(m \* sizeof(int));

B[i]=(int\*)malloc(m \* sizeof(int));

for(int j=0;j<m;j++)

{

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

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

}

}

int\*\* C;

C = (int\*\*)malloc(m \* sizeof(int));

unsigned int start\_time = clock();

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

{

C[i]=(int\*)malloc(m \* sizeof(int));

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

{

xxx=0;

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

{

xxx=xxx+A[i][r]\*B[r][j];

C[i][j]=xxx;

}

}

}

unsigned int end\_time = clock();

double search\_time = (end\_time - start\_time)/1000.0;

printf("%lf\n\n",search\_time);

system("pause");

}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

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);

}

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;

}

}

}

int main(void)

{

srand(time(NULL));

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

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

system("chcp 1251");

system("cls");

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

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;

int a[100000],b[100000],n=100000;

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

{

a[i]=rand() %800;

b[i]=a[i];

}

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

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

}

printf("\n\n\n");

unsigned int start\_time = clock();

shell(a,n);

unsigned int end\_time = clock();

double search\_time = (end\_time - start\_time)/1000.0;

printf("Время сортировки рандомного массива shell :%lf\n",search\_time);

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

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

}

unsigned int start4\_time = clock();

qs(b,0,n-1);

unsigned int end4\_time = clock();

double search4\_time = (end4\_time - start4\_time)/1000.0;

printf("Время сортировки рандомного массива qs:%lf\n",search4\_time);

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

{

a[i]=i+1;

b[i]=a[i];

}

unsigned int start1\_time = clock();

shell(a,n);

unsigned int end1\_time = clock();

double search1\_time = (end1\_time - start1\_time)/1000.0;

printf("Время сортировки возрастающего массива shell :%lf\n",search1\_time);

unsigned int start5\_time = clock();

qs(b,0,n-1);

unsigned int end5\_time = clock();

double search5\_time = (end5\_time - start5\_time)/1000.0;

printf("Время сортировки возрастающего массива qs:%lf\n",search5\_time);

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

{

a[i]=100000-i;

b[i]=a[i];

}

unsigned int start2\_time = clock();

shell(a,n);

unsigned int end2\_time = clock();

double search2\_time = (end2\_time - start2\_time)/1000.0;

printf("Время сортировки убывающего массива shell :%lf\n",search2\_time);

unsigned int start6\_time = clock();

qs(b,0,n-1);

unsigned int end6\_time = clock();

double search6\_time = (end6\_time - start6\_time)/1000.0;

printf("Время сортировки убывающего массива qs:%lf\n",search6\_time);

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

{

if(i<50000){a[i]=i;}

if(i>=50000){a[i]=a[i-1]-1;}

b[i]=a[i];

}

unsigned int start8\_time = clock();

shell(a,n);

unsigned int end8\_time = clock();

double search8\_time = (end8\_time - start8\_time)/1000.0;

printf("Время сортировки сначала возрастающего а потом убывающего массива shell :%lf\n",search8\_time);

unsigned int start7\_time = clock();

qs(b,0,n-1);

unsigned int end7\_time = clock();

double search7\_time = (end7\_time - start7\_time)/1000.0;

printf("Время сортировки сначала возрастающего а потом убывающего массива qs:%lf\n",search7\_time);

system("pause");

}

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include <locale>

void sort (int random Number[], 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;

}

}

}

}

int main(void)

{

srand(time(NULL));

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

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

setlocale(LC\_ALL,"Rus");

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

clock\_t start1, end1;

clock\_t start2, end2;

clock\_t start3, end3;

int a[100000],n=100000;

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

{

a[i]=rand() %800;

}

printf("\n\n\n");

unsigned int start\_time = clock();

sort (a,n);

unsigned int end\_time = clock();

double search\_time = (end\_time - start\_time)/1000.0;

printf("Время сортировки рандомного массива easy :%lf\n",search\_time);

unsigned int start1\_time = clock();

sort (a,n);

unsigned int end1\_time = clock();

double search1\_time = (end1\_time - start1\_time)/1000.0;

printf("Время сортировки возрастающего массива easy :%lf\n",search1\_time);

unsigned int start2\_time = clock();

sort (a,n);

unsigned int end2\_time = clock();

double search2\_time = (end2\_time - start2\_time)/1000.0;

printf("Время сортировки убывающего массива easy :%lf\n",search2\_time);

unsigned int start3\_time = clock();

sort (a,n);

unsigned int end3\_time = clock();

double search3\_time = (end3\_time - start3\_time)/1000.0;

printf("Время сортировки сначала возрастающего а потом убывающего массива easy :%lf\n",search3\_time);

system("pause");

}

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

Время выполнения программы Быстрой сортировкой 0,011

Время выполнения программы сортировкой Шелли 1,002

Время выполнения программы Быстрой сортировкой из библиотеки 0,039

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

Время выполнения программы Быстрой сортировкой 0,007

Время выполнения программы сортировкой Шелли 0,003

Время выполнения программы Быстрой сортировкой из библиотеки 0,026

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

Время выполнения программы Быстрой сортировкой 0,06

Время выполнения программы сортировкой Шелли 1,968

Время выполнения программы Быстрой сортировкой из библиотеки 0,026

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

Время выполнения программы Быстрой сортировкой 2,373000

Время выполнения программы сортировкой Шелли 0,977

Время выполнения программы Быстрой сортировкой из библиотеки 0,025

Вывод:

В ходе данной работы я научился оценивать время работы программы.