Пензенский государственный университет

Кафедра “Вычислительная техника”

Отчёт  
о выполнении лабораторной работы №2  
по дисциплине "Логика и основы алгоритмизации в инженерных задачах"  
на тему  
**"**Оценка времени выполнения программ**"**

Выполнили студенты гр.21ВВ3:  
Борисова А. В.  
Саломатин В. А.

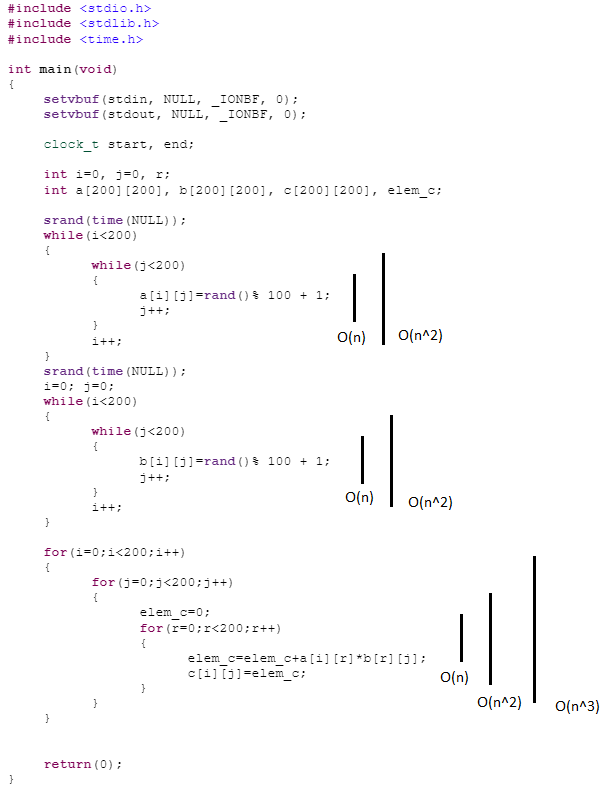
Принял:  
д.т.н. профессор Митрохин М.А. к.т.н. доцент Юрова О.В.

Пенза 2022

Лабораторное задание:

**Задание 1:**

1. Вычислить порядок сложности программы (*О*-символику).



O(max(n, n^2, n^3)) = O(n^3)

1. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 20, 50, 80, 100, 150, 200, 250.

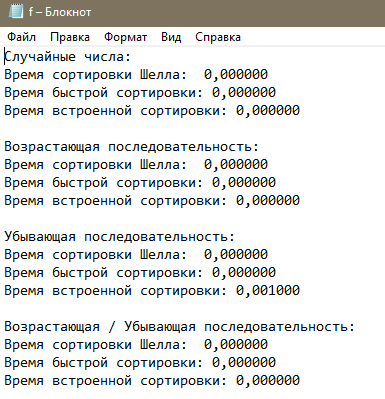
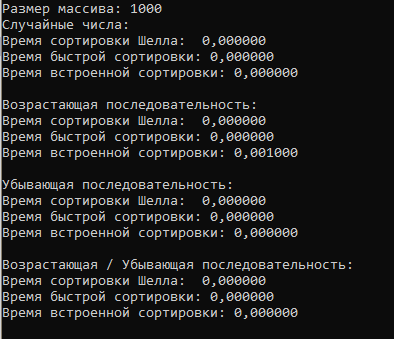
|  |  |  |
| --- | --- | --- |
| № Теста | Размер матриц | Время выполнения программы |
| 1. | 20 | 0,010 |
| 2. | 50 | 0,010 |
| 3. | 80 | 0,010 |
| 4. | 100 | 0,010 |
| 5. | 150 | 0,020 |
| 6. | 200 | 0,040 |
| 7. | 250 | 0,095 |

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

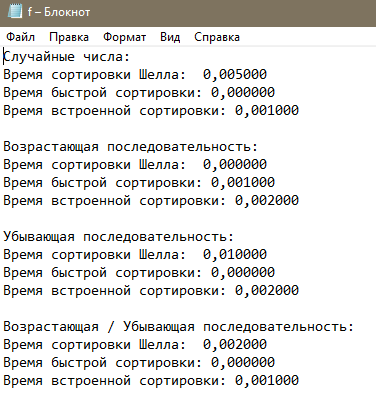
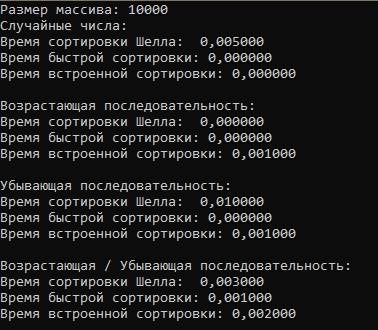
**Задание 2**:

Оценили время работы каждого из реализованных алгоритмов на случайном наборе значений, c возрастающей последовательностью чисел, c убывающей последовательностью чисел, сначала с возрастающей последовательностью чисел, а затем – убывающей. Оценили время работы стандартной функции qsort,

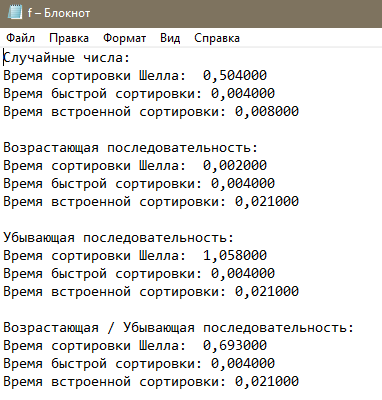
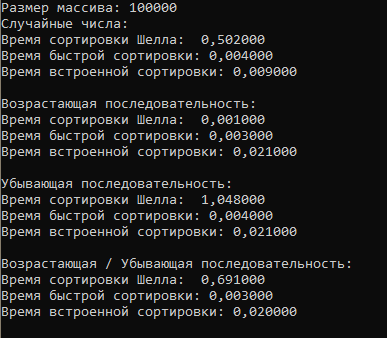
Количество элементов – **1000**:



Количество элементов – **10000**:



Количество элементов – **100000**:



**Листинг:**

#include"stdafx.h"

#define\_CRT\_SECURE\_NO\_WARNINGS

#define\_CTR\_NONSTDC\_NO\_WARNINGS

#include<stdio.h>

#include<conio.h>

#include<time.h>

#include<stdlib.h>

#include<random>

#include<locale>

voidshell(int\*mass,intn)

{

inti,j,gap,k;

intx,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<n;++i){

x=mass[i];

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

mass[j+gap]=mass[j];

mass[j+gap]=x;

}

}

}

voidqs(int\*mass,intleft,intright)

{

inti,j;

intx,y;

i=left;

j=right;

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

do{

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

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

if(i<=j)

{

y=mass[i];

mass[i]=mass[j];

mass[j]=y;

i++;j--;

}

}while(i<=j);

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

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

}

intcomp(constvoid\*i,constvoid\*j){

return\*(int\*)i-\*(int\*)j;

}

voidrandmass(int\*mass,intn){

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

mass[i]=rand()%100;

}

}

voidvozrmass(int\*mass,intn){

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

mass[i]=i;

}

}

voidybmass(int\*mass,intn){

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

mass[i]=n-i;

}

}

voidvozrybmass(int\*mass,intn){

for(inti=0;i<n/2;i++){

mass[i]=mass[i-1]+i;

}

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

mass[i]=n-i;

}

}

voidsort(FILE\*f,int\*mass,intn){

randmass(mass,n);

time\_tstart1=clock();

shell(mass,n);

time\_tstop1=clock();

doubletime1=(stop1-start1)/1000.0;

printf("\n");

printf("Время сортировки Шелла: %lf",time1);

fprintf(f,"Время сортировки Шелла: %lf\n",time1);

printf("\n");

randmass(mass,n);

time\_tstart2=clock();

qs(mass,0,n-1);

time\_tstop2=clock();

doubletime2=(stop2-start2)/1000.0;

printf("Время быстрой сортировки: %lf",time2);

fprintf(f,"Время быстрой сортировки: %lf\n",time2);

printf("\n");

randmass(mass,n);

time\_tstart3=clock();

qsort(mass,n,sizeof(int),comp);

time\_tstop3=clock();

doubletime3=(stop3-start3)/1000.0;

printf("Время встроенной сортировки: %lf",time3);

fprintf(f,"Время встроенной сортировки: %lf\n\n",time3);

}

voidsort1(FILE\*f,int\*mass,intn){

vozrmass(mass,n);

time\_tstart1=clock();

shell(mass,n);

time\_tstop1=clock();

doubletime1=(stop1-start1)/1000.0;

printf("\n");

printf("Время сортировки Шелла: %lf",time1);

fprintf(f,"Время сортировки Шелла: %lf\n",time1);

printf("\n");

vozrmass(mass,n);

time\_tstart2=clock();

qs(mass,0,n-1);

time\_tstop2=clock();

doubletime2=(stop2-start2)/1000.0;

printf("Время быстрой сортировки: %lf",time2);

fprintf(f,"Время быстрой сортировки: %lf\n",time2);

printf("\n");

vozrmass(mass,n);

time\_tstart3=clock();

qsort(mass,n,sizeof(int),comp);

time\_tstop3=clock();

doubletime3=(stop3-start3)/1000.0;

printf("Время встроенной сортировки: %lf",time3);

fprintf(f,"Время встроенной сортировки: %lf\n\n",time3);

}

voidsort2(FILE\*f,int\*mass,intn){

ybmass(mass,n);

time\_tstart1=clock();

shell(mass,n);

time\_tstop1=clock();

doubletime1=(stop1-start1)/1000.0;

printf("\n");

printf("Время сортировки Шелла: %lf",time1);

fprintf(f,"Время сортировки Шелла: %lf\n",time1);

printf("\n");

ybmass(mass,n);

time\_tstart2=clock();

qs(mass,0,n-1);

time\_tstop2=clock();

doubletime2=(stop2-start2)/1000.0;

printf("Время быстрой сортировки: %lf",time2);

fprintf(f,"Время быстрой сортировки: %lf\n",time2);

printf("\n");

ybmass(mass,n);

time\_tstart3=clock();

qsort(mass,n,sizeof(int),comp);

time\_tstop3=clock();

doubletime3=(stop3-start3)/1000.0;

printf("Время встроенной сортировки: %lf",time3);

fprintf(f,"Время встроенной сортировки: %lf\n\n",time3);

}

voidsort3(FILE\*f,int\*mass,intn){

vozrybmass(mass,n);

time\_tstart1=clock();

shell(mass,n);

time\_tstop1=clock();

doubletime1=(stop1-start1)/1000.0;

printf("\n");

printf("Время сортировки Шелла: %lf",time1);

fprintf(f,"Время сортировки Шелла: %lf\n",time1);

printf("\n");

vozrybmass(mass,n);

time\_tstart2=clock();

qs(mass,0,n-1);

time\_tstop2=clock();

doubletime2=(stop2-start2)/1000.0;

printf("Время быстрой сортировки: %lf",time2);

fprintf(f,"Время быстрой сортировки: %lf\n",time2);

printf("\n");

vozrybmass(mass,n);

time\_tstart3=clock();

qsort(mass,n,sizeof(int),comp);

time\_tstop3=clock();

doubletime3=(stop3-start3)/1000.0;

printf("Время встроенной сортировки: %lf",time3);

fprintf(f,"Время встроенной сортировки: %lf\n\n",time3);

}

intmain()

{

FILE\*f;

int\*mass;

intn;

f=fopen("f.txt","w");

setlocale(LC\_ALL,"RUS");

printf("Размер массива: ");

scanf\_s("%d",&n);

srand(time(NULL));

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

printf("Случайные числа:");

fprintf(f,"Случайные числа:\n");

sort(f,mass,n);

printf("\n");

printf("\n");

printf("Возрастающая последовательность:");

fprintf(f,"Возрастающая последовательность:\n");

sort1(f,mass,n);

printf("\n");

printf("\n");

printf("Убывающая последовательность:");

fprintf(f,"Убывающая последовательность:\n");

sort2(f,mass,n);

printf("\n");

printf("\n");

printf("Возрастающая / Убывающая последовательность:");

fprintf(f,"Возрастающая / Убывающая последовательность:\n");

sort3(f,mass,n);

fclose(f);

free(mass);

getchar();

getchar();

return(0);

}

**Вывод:** в ходе лабораторной работы мы оценили порядок сложности программы, которая составила O(n^3).Выяснили, что с увеличением объема данных возрастает время работы программы. А также оценили время, затраченное программой, на осуществление некоторых сортировок. Сравнив показатели, выяснили, что быстрая сортировка затрачивает меньше всего времени в случаях убывающей, случайной и убывающе-возрастающей последовательностей. В случае возрастающей последовательности быстрее всех работает сортировка Шелла.