Министерство высшего образования и науки Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Логика и основы алгоритмизации в ИЗ»

на тему: «Оценка времени выполнения программ».

Выполнилистуденты

группы 21ВВ1:  
Федулов Е.Д.  
Жбанников Д.Н.

Приняли:  
Акифьев И.В.  
Юрова О.В.

Пенза, 2022

**Цель работы:** Изучение темы времени выполнения программы, знакомство с библиотекой time.h., а также знакомство с О-символикой.

**Лабораторное 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.

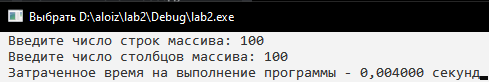
**Ход работы:**

1.Вычисление порядка сложности программы:

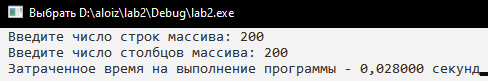
Сложность данной программы будет n3

2.Оценка времени выполнения программы и кода, выполняющего перемножение матриц размерами 100, 200, 400, 1000, 2000, 4000, 10000.

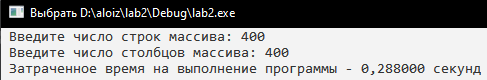
**Результат для размера 100:**



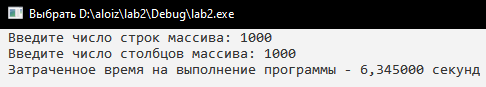
**Результат для размера 200:**



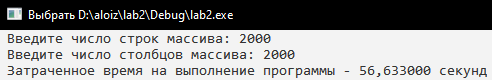
**Результат для размера 400:**



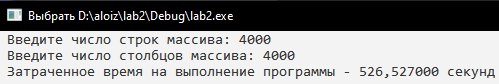
**Результат для размера 1000:**



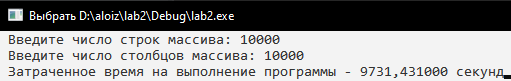
**Результат для размера 2000:**



**Результат для размера 4000:**



**Результат для размера 10000:**



3.График зависимости времени выполнения программы от размера матриц

Теоретическая оценка

**Лабораторное 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.

**Листинг программы:**

#define\_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include"Windows.h"

#include<locale.h>

#include"iostream"

#pragmacomment(linker,"/STACK:1468006400")

#defineSIZE50000

clock\_tstartRand,startPr,startObr,startSmesh,

endSRand,endQRand,endQsortRand,

endSPr,endQPr,endQsortPr,

endSObr,endQObr,endQsortObr,

endSSm,endQSm,endQsortSm,startSRand,startQRand,startQsortRand;

intcomp(constvoid\*,constvoid\*);

usingnamespacestd;

FILE\*D1;

voidquickSort(int\*numbers,intleft,intright)

{

intpivot;

intl\_hold=left;

intr\_hold=right;

pivot=numbers[left];

while(left<right)

{

while((numbers[right]>=pivot)&&(left<right))

right--;

if(left!=right)

{

numbers[left]=numbers[right];

left++;

}

while((numbers[left]<=pivot)&&(left<right))

left++;

if(left!=right)

{

numbers[right]=numbers[left];

right--;

}

}

numbers[left]=pivot;

pivot=left;

left=l\_hold;

right=r\_hold;

if(left<pivot)

quickSort(numbers,left,pivot-1);

if(right>pivot)

quickSort(numbers,pivot+1,right);

}

voidshellSort(int\*num,intsize)

{

intinc=3;

while(inc>0)

{

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

{

intj=i;

inttemp=num[i];

while((j>=inc)&&(num[j-inc]>temp))

{

num[j]=num[j-inc];

j=j-inc;

}

num[j]=temp;

}

if(inc>1)

inc=inc/2;

elseif(inc==1)

break;

}

}

voidrandfunc(int\*mas)

{

for(inti=0;i<10000;i++)

{

mas[i]=rand()%100+1000;

}

}

voidprfunc(int\*mas)

{

for(inti=0;i<10000;i++)

{

mas[i]=rand()%100+1000+i;

}

}

voidobrfunc(int\*mas)

{

for(inti=0;i<10000;i++)

{

mas[i]=rand()%100+1000-i;

}

}

voidsmeshfunc(int\*mas)

{

for(inti=0;i<10000;i++)

{

if(i<=5000)

mas[i]=rand()%100+1000+i;

else

mas[i]=rand()%100+1000-i;

}

}

intmain()

{

setlocale(0,"Russian");

intRand[SIZE];

intPr[SIZE];

intObr[SIZE];

intSmesh[SIZE];

randfunc(Rand);

startSRand=clock();

shellSort(Rand,SIZE);

endSRand=clock();

randfunc(Rand);

quickSort(Rand,0,SIZE-1);

endQRand=clock();

randfunc(Rand);

qsort(Rand,SIZE,sizeof(int),comp);

endQsortRand=clock();

//----------------------------------------—

prfunc(Pr);

startPr=clock();

shellSort(Pr,SIZE);

endSPr=clock();

prfunc(Pr);

quickSort(Pr,0,SIZE-1);

endQPr=clock();

prfunc(Pr);

qsort(Pr,SIZE,sizeof(int),comp);

endQsortPr=clock();

//------------------------------------------—

obrfunc(Obr);

startObr=clock();

shellSort(Obr,SIZE);

endSObr=clock();

obrfunc(Obr);

quickSort(Obr,0,SIZE-1);

endQObr=clock();

obrfunc(Obr);

qsort(Obr,SIZE,sizeof(int),comp);

endQsortObr=clock();

//---------------------------------------—

smeshfunc(Smesh);

startSmesh=clock();

shellSort(Smesh,SIZE);

endSSm=clock();

smeshfunc(Smesh);

quickSort(Smesh,0,SIZE-1);

endQSm=clock();

smeshfunc(Smesh);

qsort(Smesh,SIZE,sizeof(int),comp);

endQsortSm=clock();

doubletimeSRand=(endSRand-startSRand)/1000.0;

doubletimeQRand=(endQRand-endSRand)/1000.0;

doubletimeQsortRand=(endQsortRand-endQRand)/1000.0;

doubletimeSPr=(endSPr-startPr)/1000.0;

doubletimeQPr=(endQPr-endSPr)/1000.0;

doubletimeQsortPr=(endQsortPr-endQPr)/1000.0;

doubletimeSObr=(endSObr-startObr)/1000.0;

doubletimeQObr=(endQObr-endSObr)/1000.0;

doubletimeQsortObr=(endQsortObr-endQObr)/1000.0;

doubletimeSSm=(endSSm-startSmesh)/1000.0;

doubletimeQSm=(endQSm-endSSm)/1000.0;

doubletimeQsortSm=(endQsortSm-endQSm)/1000.0;

printf("\n");

printf("+---------------+--------------------+----------------+----------------------+\n");

printf("|---------------|----Быстрая сорт.---|---Сорт. Шелла--|---Быстрая сорт(ф-я)--|\n");

printf("|-Прямая посл.--| %lf | %lf | %lf |\n",timeQPr,timeSPr,timeQsortPr);

printf("|-Обратная посл.| %lf | %lf | %lf |\n",timeQObr,timeSObr,timeQsortObr);

printf("|--Смеш. посл.--| %lf | %lf | %lf |\n",timeQSm,timeSSm,timeQsortSm);

printf("|----Рандом.----| %lf | %lf | %lf |\n",timeQRand,timeSRand,timeQsortRand);

printf("+---------------+--------------------+----------------+----------------------+\n");

D1=fopen("11.txt","w");

fprintf(D1,"\n");

fprintf(D1,"+---------------+--------------------+----------------+----------------------+\n");

fprintf(D1,"|---------------|----Быстрая сорт.---|---Сорт. Шелла--|---Быстрая сорт(ф-я)--|\n");

fprintf(D1,"|-Прямая посл.--| %lf | %lf | %lf |\n",timeQPr,timeSPr,timeQsortPr);

fprintf(D1,"|-Обратная посл.| %lf | %lf | %lf |\n",timeQObr,timeSObr,timeQsortObr);

fprintf(D1,"|--Смеш. посл.--| %lf | %lf | %lf |\n",timeQSm,timeSSm,timeQsortSm);

fprintf(D1,"|----Рандом.----| %lf | %lf | %lf |\n",timeQRand,timeSRand,timeQsortRand);

fprintf(D1,"+---------------+--------------------+----------------+----------------------+\n");

fclose(D1);

getchar();

return0;

}

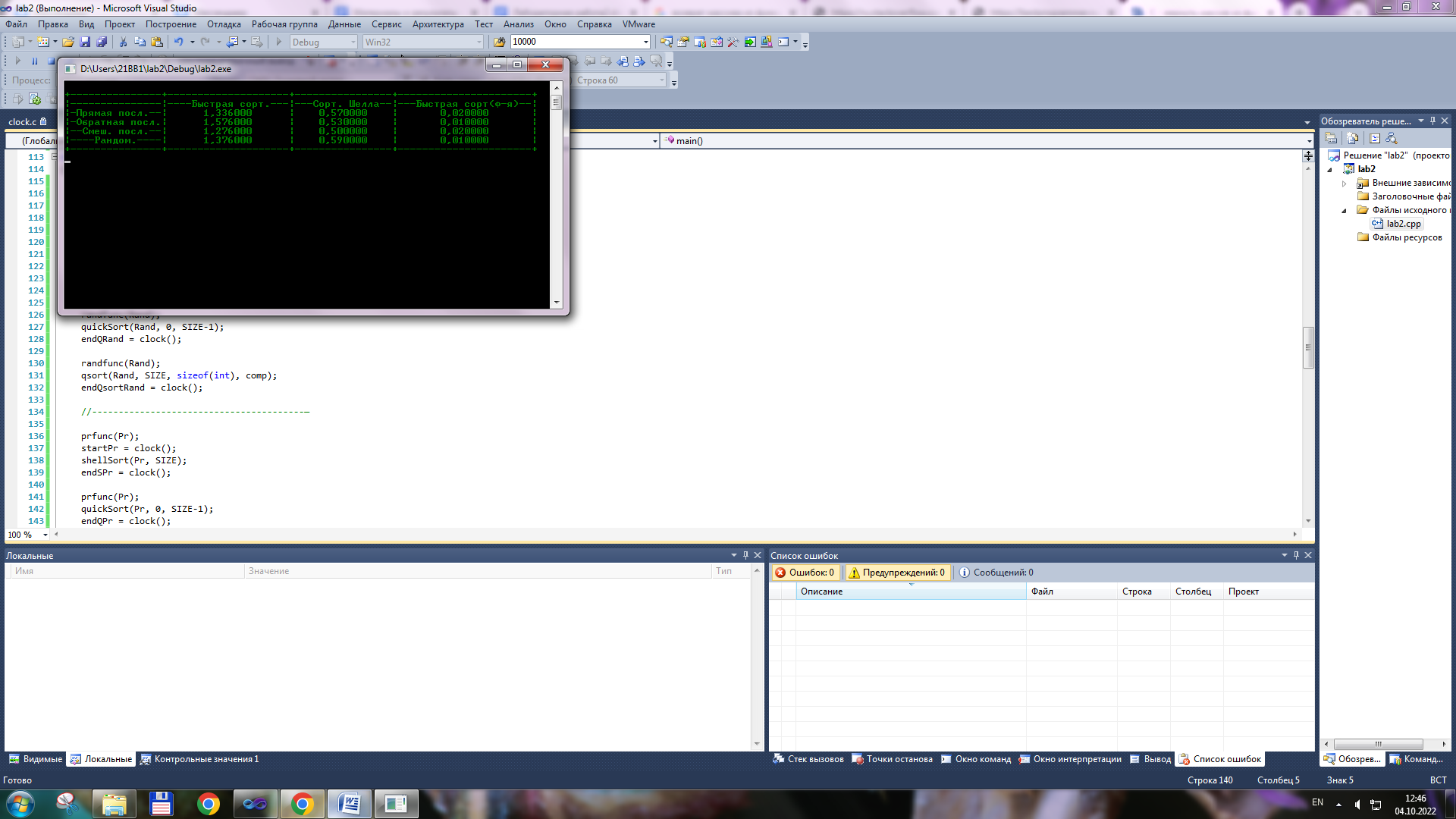
intcomp(constvoid\*i,constvoid\*j)

{

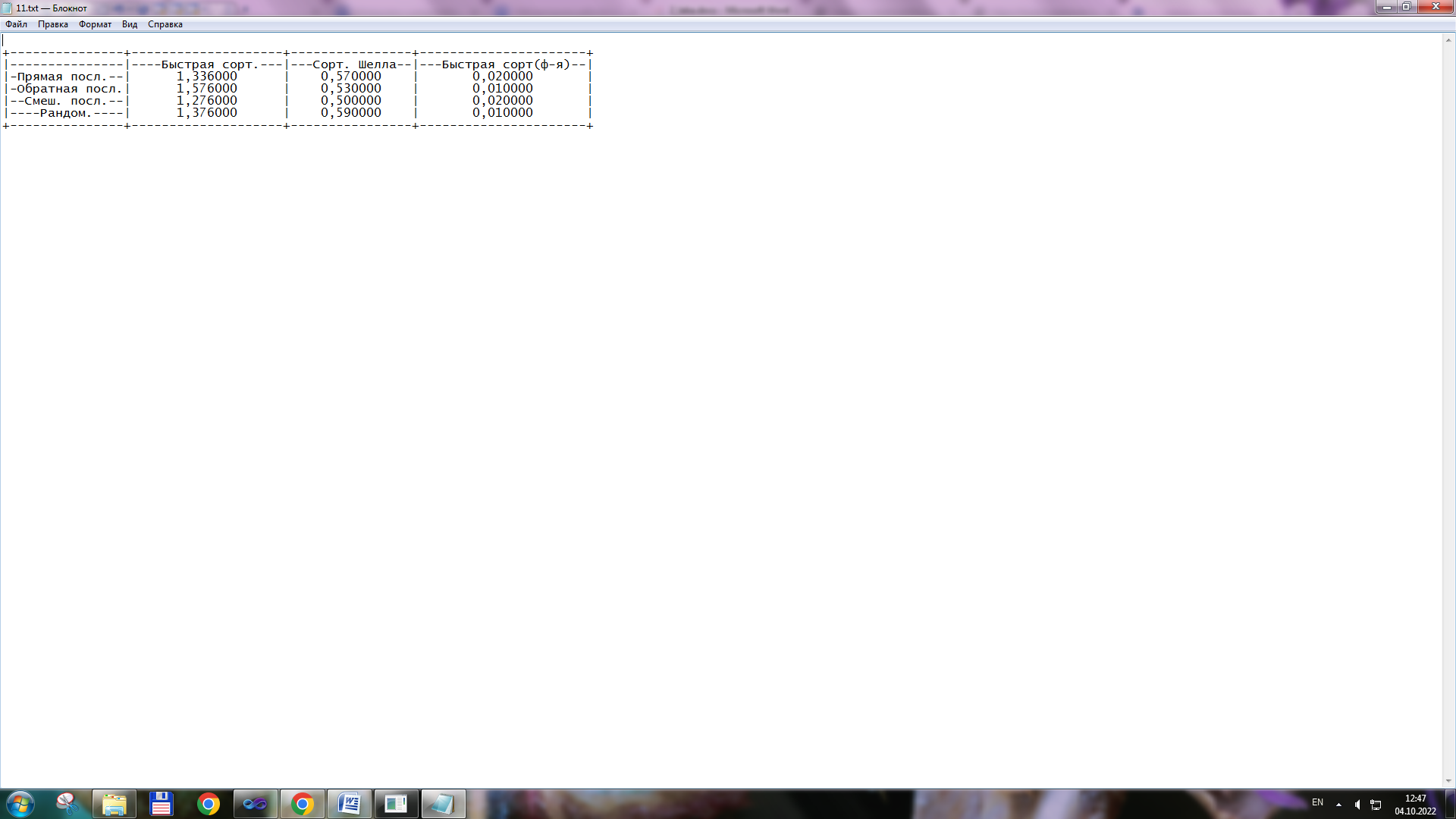
return\*(int\*)i-\*(int\*)j;

}

**Результат работы:**

****

**Запись в файл:**



**Вывод:** в лабораторной работе изучили языковые средства и принципы организации простых структур на языке Си, приобрели навыки работы с массивами, строками, структурами.

**Вывод по 1 лабораторному заданию:**

1. Сложность программы будет O(n3), поскольку всего имеется 3 вложенных цикла в программе.
2. Время расчета программы увеличивается при увеличении массива

**Вывод по 2 лабораторному заданию:**

1. Для Сортировки Шелла самый оптимальный способ это прямая последовательность.
2. Для быстрой сортировки самый оптимальный способ это смешанная последовательность.
3. Для функции быстрой сортировки самые оптимальные способы это случайный способ и обратная последовательность.