Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра "Вычислительная техника"

Отчет

по лабораторной работе №2

по курсу “Л и ОА в ИЗ”

на тему “Оценка времени выполнения программ”

Выполнил студенты группы 22ВВС1:

Денисов Д.К.

Токарев А.А.

Приняли:

Юрова О.В.

Акифьев И.В.

Пенза 2023

### 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, 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**

Длязадания 1

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

System::Void Project1::MyForm::button1\_Click(System::Object^ sender, System::EventArgs^ e)

{

inti = 0;

intn[7] = { 1,2,4,10,20,40,100 };

this->chart1->Series[0]->Points->Clear();

while (i< 7) {

double t = multi(n[i] \* 100);

printf("time %lf , kolvo %d \n", t, n[i] \* 100);

this->chart1->Series[0]->Points->AddXY(n[i] \* 100, t);

i++;

}

returnSystem::Void();

}

double Project1::MyForm::multi(int size)

{

srand(time(NULL));

clock\_t start, end;

int r = 0;

inti = 0;

int j = 0;

int s = 0;

doubleelem\_c;

int\*\* a = (int\*\*)malloc(sizeof(int\*) \* size);

int\*\* b = (int\*\*)malloc(sizeof(int\*) \* size);

int\*\* c = (int\*\*)malloc(sizeof(int\*) \* size);

while (size >i) {

j = 0;

b[i] = (int\*)malloc(sizeof(int) \* size);

a[i] = (int\*)malloc(sizeof(int) \* size);

c[i] = (int\*)malloc(sizeof(int) \* size);

while (size > j) {

a[i][j] = rand() % 1000;

b[i][j] = rand() % 1000;

c[i][j] = rand() % 1000;

j++;

}

i++;

}

start = clock();

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

{

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

{

elem\_c = 0;

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

{

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

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

}

}

}

end = clock();

free(a);

free(b);

free(c);

while (size >i) {

free(a[i]);

free(b[i]);

free(c[i]);

i++;

}

double diff = (end - start);

double t = diff / CLOCKS\_PER\_SEC;

return(t);

}

Длязадания 2

#define\_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<time.h>

#include<stdlib.h>

voidqs(int\* items, intleft, intright);

voidshell(int\* items, intcount);

intcompare(constvoid\* x1, constvoid\* x2);

intmain() {

srand(time(NULL));

clock\_t start, end;

int\* mas = (int\*)malloc(sizeof(int) \* 100000);

for (inti = 0; i< 100000; i++) {

mas[i] = i;

}

printf("/ \n");

start = clock();

shell(mas, 100000);

end = clock();

double shelltime1 = end - start;

shelltime1 = shelltime1 / CLOCKS\_PER\_SEC;

printf("shell %lf\n", shelltime1);

start = clock();

qs(mas, 0, 100000);

end = clock();

double qs1 = end - start;

qs1 = qs1 / CLOCKS\_PER\_SEC;

printf("qs %lf\n", qs1);

start = clock();

qsort(mas, 100000, sizeof(int), compare);

end = clock();

double qsort1 = end - start;

qsort1 = qsort1 / CLOCKS\_PER\_SEC;

printf("qsort %lf\n", qsort1);

printf("\n");

int j = 99999;

for (inti = 0; i< 100000; i++) {

mas[i] = j;

j--;

}

printf("\\ \n");

start = clock();

shell(mas, 100000);

end = clock();

shelltime1 = end - start;

shelltime1 = shelltime1 / CLOCKS\_PER\_SEC;

printf("shell %lf\n", shelltime1);

start = clock();

qs(mas, 0, 100000);

end = clock();

qs1 = end - start;

qs1 = qs1 / CLOCKS\_PER\_SEC;

printf("qs %lf\n", qs1);

start = clock();

qsort(mas, 100000, sizeof(int), compare);

end = clock();

qsort1 = end - start;

qsort1 = qsort1 / CLOCKS\_PER\_SEC;

printf("qsort %lf\n", qsort1);

printf("\n");

j = 50000;

for (inti = 0; i< 100000; i++) {

if (i< 50000) {

mas[i] = i;

}

else {

mas[i] = j;

j--;

}

}

printf("/\\ \n");

start = clock();

shell(mas, 100000);

end = clock();

shelltime1 = end - start;

shelltime1 = shelltime1 / CLOCKS\_PER\_SEC;

printf("shell %lf\n", shelltime1);

start = clock();

qs(mas, 0, 100000);

end = clock();

qs1 = end - start;

qs1 = qs1 / CLOCKS\_PER\_SEC;

printf("qs %lf\n", qs1);

start = clock();

qsort(mas, 100000, sizeof(int), compare);

end = clock();

qsort1 = end - start;

qsort1 = qsort1 / CLOCKS\_PER\_SEC;

printf("qsort %lf\n", qsort1);

printf("\n");

for (inti = 0; i< 100000; i++) {

mas[i] = rand() % 100000;

}

printf("rand \n");

start = clock();

shell(mas, 100000);

end = clock();

shelltime1 = end - start;

shelltime1 = shelltime1 / CLOCKS\_PER\_SEC;

printf("shell %lf\n", shelltime1);

start = clock();

qs(mas, 0, 100000);

end = clock();

qs1 = end - start;

qs1 = qs1 / CLOCKS\_PER\_SEC;

printf("qs %lf\n", qs1);

start = clock();

qsort(mas, 100000, sizeof(int), compare);

end = clock();

qsort1 = end - start;

qsort1 = qsort1 / CLOCKS\_PER\_SEC;

printf("qsort %lf\n", qsort1);

}

voidshell(int\* items, intcount)

{

inti, 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;

}

}

}

}

voidqs(int\* items, intleft, intright)

{

inti, 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);

}

intcompare(constvoid\* x1, constvoid\* x2) // функция сравнения элементов массива

{

return (\*(int\*)x1 - \*(int\*)x2); // если результат вычитания равен 0, то числа равны, < 0: x1 < x2; > 0: x1 > x2

}

**Результаты работ программ**

Задание 1

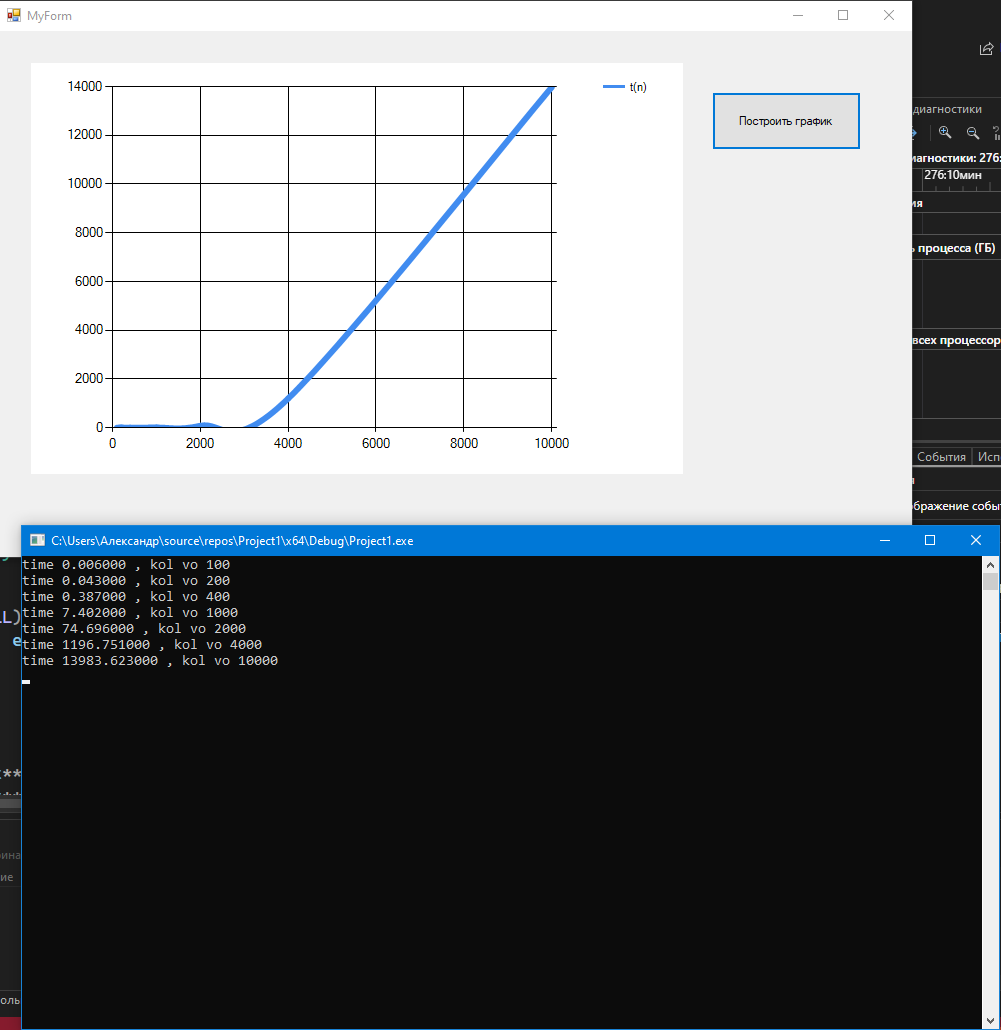
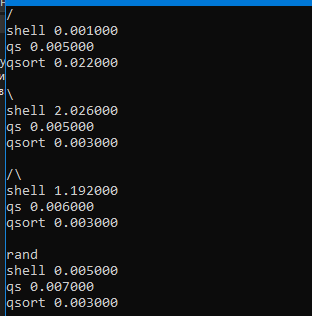


График 1

Задание 2



**Вывод**

### В ходе выполнения лабораторной работы были обобщены знания по оценке сложности и времени выполнения программы. O символика алгоритма перемножения матриц является функция O(n^3). Данная зависимость была экспериментально подтверждена(См. график 1).Был сделан вывод что О символика неплохо справляется с определением сложности и предсказанием времени выполнения алгоритма.

### Лучшего всего с сортировкой справляется алгоритм qsort. Алгоритм Шелла выдаёт хорошие результаты только в уже отсортированном массиве или в массиве случайных чисел. Алгоритм быстрой сортировки является универсальным вариантом, в случае если мы не знаем, что за массив перед нами.