Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

# по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Оценка времени выполнения программ»

**Выполнили:**

студенты группы 20ВВ3

Баулин Александр

Духнов Олег

Культиясов Ярослав

**Приняли:**

Юрова О.В.

Митрохин М.А.

Пенза 2021

# Название

Оценка времени выполнения программ

**Цель работы** – изучение функций, предназначенных для оценки времени работы 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.

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.

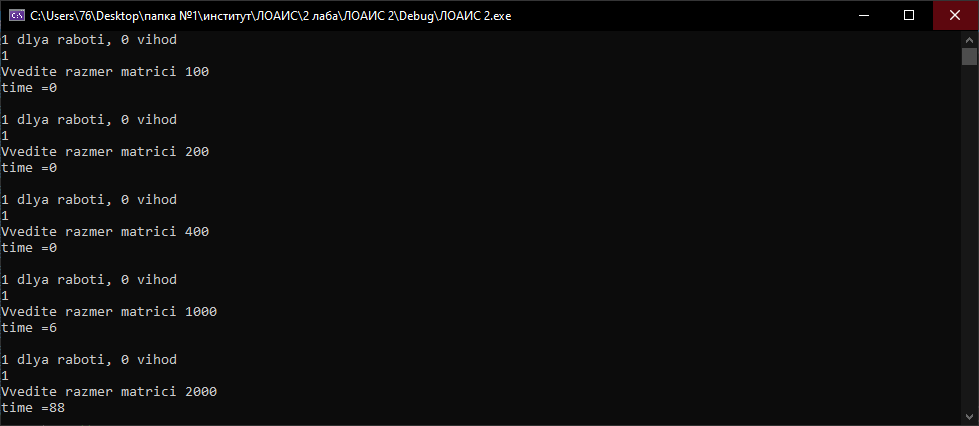
# Результат работы программы

**Задание 1.1.**

Сложность программы O(n3).

**Задание 1.2.**

Результаты работы программы показаны на рисунке 1.



**Рисунок 1 – Результат работы программы**

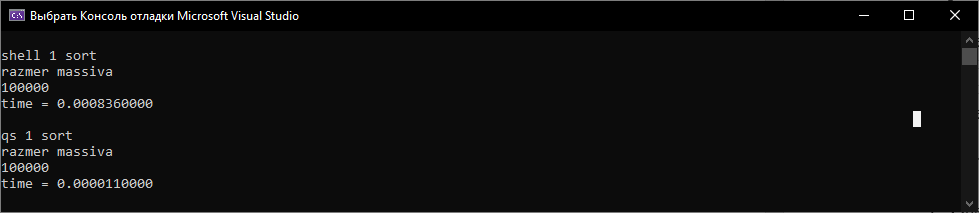
**Задание 1.3.**

**Вывод по первой части**

Аналитика работы программы подтвердила O(n3) сложность.

**Задание 2.1.**

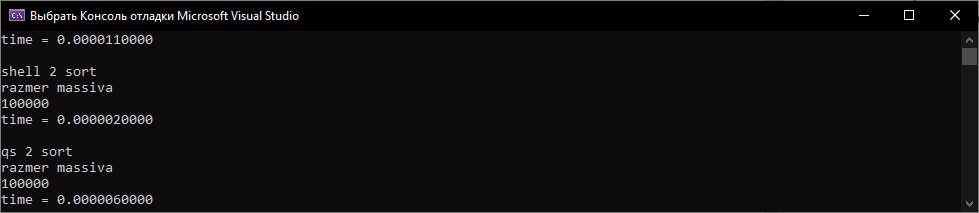
Результаты работы программы показаны на рисунке 2.



**Рисунок 2 – Результат работы программы**

**Задание 2.2.**

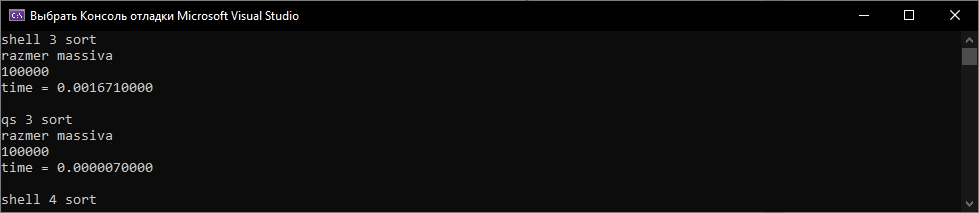
Результаты работы программы показаны на рисунке 3.



**Рисунок 3 – Результат работы программы**

**Задание 2.3.**

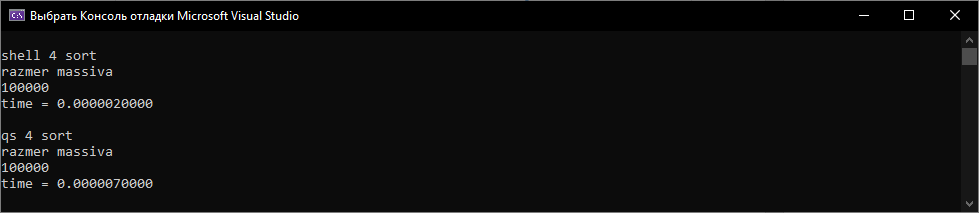
Результаты работы программы показаны на рисунке 4.



**Рисунок 4 – Результат работы программы**

**Задание 2.4.**

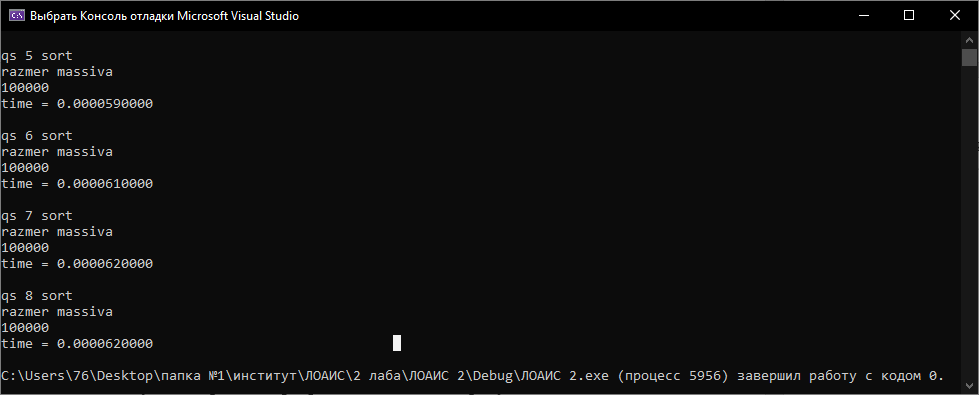
Результаты работы программы показаны на рисунке 5.



**Рисунок 5 – Результат работы программы**

**Задание 2.5.**

Результаты работы программы показаны на рисунке 6.



**Рисунок 6 – Результат работы программы**

**Вывод по второй части**

Провели анализ работы разных методов сортировки в разных условиях.

|  |  |  |  |
| --- | --- | --- | --- |
| **Последовательность** | **Быстрая сортировка** | **Сортировка Шелла** | **Быстрая сортировка (встроенная)** |
| **Случайная** | **0.000011** | **0.000836** | **0.000059** |
| **Возрастающая** | **0.000006** | **0.000002** | **0.000061** |
| **Убывающая** | **0.000007** | **0.001671** | **0.000062** |
| **Возрастающая, убывающая** | **0.000007** | **0.000002** | **0.000062** |

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

#include <malloc.h>

int comp(const void\* i, const void\* j)

{

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

}

void e1(int n)

{

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

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

clock\_t start, end;

int i, j, r;

int\*\* f, \*\* g, \*\* t;

int elem\_c;

float time\_spent;

srand(time(NULL));

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

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

{

f[i] = (int\*)malloc(n \* sizeof(int));

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

{

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

}

}

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

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

{

g[i] = (int\*)malloc(n \* sizeof(int));

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

{

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

}

}

start = clock();

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

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

{

t[i] = (int\*)malloc(n \* sizeof(int));

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

{

elem\_c = 0;

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

{

elem\_c = elem\_c + f[i][r] \* g[r][j];

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

}

}

}

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time =%.10f ", time\_spent);

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

{

free(f[i]);

free(g[i]);

free(t[i]);

}

free(f);

free(g);

free(t);

}

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 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);

}

int main()

{

int k, h, i, r, x1, y1, x2, y2;

int\* a, \* b;

clock\_t start, end;

float time\_spent;

srand(time(NULL));

//e1--------------------------------------------------------------------

do

{

printf("1 dlya raboti, 0 vihod\n");

scanf("%d", &k);

if (k == 0)

{

printf("end\n\n");

break;

}

if (k == 1)

{

printf("Vvedite razmer matrici ");

scanf("%d", &k);

e1(k);

printf("\n\n");

}

} while (true);

//e2--------------------------------------------------------------------

printf("shell 1 sort\n");

printf("razmer massiva\n");

scanf("%d", &h);

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

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

{

a[i] = rand() % 100;

}

start = clock();

shell(a, h);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(a);

printf("\n");

printf("qs 1 sort\n");

printf("razmer massiva\n");

scanf("%d", &r);

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

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

{

b[i] = rand() % 100;

}

start = clock();

qs(b,0,r-1);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(b);

printf("\n");

//e3--------------------------------------------------------------------

printf("shell 2 sort\n");

printf("razmer massiva\n");

scanf("%d", &h);

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

x1 = 1;

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

{

a[i] = x1;

x1++;

}

start = clock();

shell(a, h);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(a);

printf("\n");

printf("qs 2 sort\n");

printf("razmer massiva\n");

scanf("%d", &r);

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

y1 = 1;

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

{

b[i] = y1;

y1++;

}

start = clock();

qs(b, 0, r - 1);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(b);

printf("\n");

//e4--------------------------------------------------------------------

printf("shell 3 sort\n");

printf("razmer massiva\n");

scanf("%d", &h);

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

x1 = h;

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

{

a[i] = x1;

x1--;

}

start = clock();

shell(a, h);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(a);

printf("\n");

printf("qs 3 sort\n");

printf("razmer massiva\n");

scanf("%d", &r);

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

y1 = r;

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

{

b[i] = y1;

y1--;

}

start = clock();

qs(b, 0, r - 1);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(b);

printf("\n");

//e5--------------------------------------------------------------------

printf("shell 4 sort\n");

printf("razmer massiva\n");

scanf("%d", &h);

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

x1 = 1;

x2 = h;

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

{

if ((h / 2) < h)

{

a[i] = x1;

x1++;

}

else

{

a[i] = x2;

x2--;

}

}

start = clock();

shell(a, h);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(a);

printf("\n");

printf("qs 4 sort\n");

printf("razmer massiva\n");

scanf("%d", &r);

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

y1 = 1;

y2 = r;

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

{

if ((r / 2) < r)

{

b[i] = y1;

y1++;

}

else

{

b[i] = y2;

y2--;

}

}

start = clock();

qs(b, 0, r - 1);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(b);

printf("\n");

//e6--------------------------------------------------------------------

printf("qs 5 sort\n");

printf("razmer massiva\n");

scanf("%d", &r);

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

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

{

b[i] = rand() % 100;

}

start = clock();

qsort(b, r, sizeof(int), comp);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(b);

printf("\n");

printf("qs 6 sort\n");

printf("razmer massiva\n");

scanf("%d", &r);

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

y1 = 1;

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

{

b[i] = y1;

y1++;

}

start = clock();

qsort(b, r, sizeof(int), comp);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(b);

printf("\n");

printf("qs 7 sort\n");

printf("razmer massiva\n");

scanf("%d", &r);

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

y1 = r;

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

{

b[i] = y1;

y1--;

}

start = clock();

qsort(b, r, sizeof(int), comp);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(b);

printf("\n");

printf("qs 8 sort\n");

printf("razmer massiva\n");

scanf("%d", &r);

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

y1 = 1;

y2 = r;

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

{

if ((r / 2) < r)

{

b[i] = y1;

y1++;

}

else

{

b[i] = y2;

y2--;

}

}

start = clock();

qsort(b, r, sizeof(int), comp);

end = clock();

time\_spent = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("time = %.10f\n ", time\_spent);

free(b);

return 0;

}