Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №2  
по курсу «Логика и основы алгоритмизации в инженерных задачах»  
на тему «Оценка времени выполнения программ»

Выполнили студенты группы 22ВВВ2:  
Захаров А. С.

Приняли:  
Акифьев И. В.  
Митрохин М. А.

Пенза 2023

Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №1  
по курсу «Логика и основы алгоритмизации в инженерных задачах»  
на тему «Простые структуры данных»

Выполнили студенты группы 22ВВВ2:  
Захаров А. С.

Приняли:  
Акифьев И. В.  
Митрохин М. А.

Пенза 2023

**Название**

Оценка времени выполнения программ.

**Цель работы**

Изучение программного метода измерения времени выполнения программ

**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.

**Метод решения задачи**

Начало измерения будем записывать в переменную start, конец же, в переменную, находящуюся после выполнения работы функции, найдем разницу между началом и концом и выведем полученную разницу.

**Листинг**

Файл main():

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "Header.h"

int main(void)

{

clock\_t start, end, end2; // объявляем переменные для определения времени выполнения

double timeOfProgram, timeOfProgram2;

start = clock();

fun1(100);

end = clock();

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

printf("working hours of the program with array %dx%d = %fs\n",100, 100, timeOfProgram);

start = clock();

fun1(200);

end = clock();

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

printf("working hours of the program with array %dx%d = %fs\n", 200, 200, timeOfProgram);

start = clock();

fun1(400);

end = clock();

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

printf("working hours of the program with array %dx%d = %fs\n", 400, 400, timeOfProgram);

start = clock();

fun1(1000);

end = clock();

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

printf("working hours of the program with array %dx%d = %fs\n", 1000, 1000, timeOfProgram);

start = clock();

fun1(2000);

end = clock();

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

printf("working hours of the program with array %dx%d = %fs\n", 2000, 2000, timeOfProgram);

start = clock();

fun1(4000);

end = clock();

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

printf("working hours of the program with array %dx%d = %fs\n", 4000, 4000, timeOfProgram);

start = clock();

fun1(10000);

end = clock();

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

printf("working hours of the program with array %dx%d = %fs\n", 10000, 10000, timeOfProgram);

printf("\n\nExercise2\n"); //Задание 2.1

start = clock();

end = fun2(50000);

end2 = clock();

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

timeOfProgram2 = (double)(end2 - start) / CLOCKS\_PER\_SEC;

printf("working hours of the program with array %dx%d:\nsort of Shell = %fs\nquick sort = %fs", 50000, 50000, timeOfProgram, timeOfProgram2);

printf("\n\nExercise3\n"); //Задание 2.2

start = clock();

end = fun3(50000);

end2 = clock();

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

timeOfProgram2 = (double)(end2 - start) / CLOCKS\_PER\_SEC;

printf("working hours of the program with array %dx%d:\nsort of Shell = %fs\nquick sort = %fs", 50000, 50000, timeOfProgram, timeOfProgram2);

printf("\n\nExercise4\n"); //Задание 2.3

start = clock();

end = fun4(50000);

end2 = clock();

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

timeOfProgram2 = (double)(end2 - start) / CLOCKS\_PER\_SEC;

printf("working hours of the program with array %dx%d:\nsort of Shell = %fs\nquick sort = %fs", 50000, 50000, timeOfProgram, timeOfProgram2);

printf("\n\nExercise5\n"); //Задание 2.4

start = clock();

end = fun5(25000);

end2 = clock();

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

timeOfProgram2 = (double)(end2 - start) / CLOCKS\_PER\_SEC;

printf("working hours of the program with array %dx%d:\nsort of Shell = %fs\nquick sort = %fs", 25000, 25000, timeOfProgram, timeOfProgram2);

printf("\n\nExercise6\n"); //Задание 2.5

start = clock();

fun6(50000);

end = clock();

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

printf("working hours of the program with array %dx%d:\nquick sort = %fs", 50000, 50000, timeOfProgram);

return 0;

}

Файл fun.cpp:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "Header.h"

int fun1(int size)

{

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

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

int i = 0, j = 0, r;

int elem\_c;

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

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

{

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

}

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

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

{

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

}

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

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

{

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

}

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

while (i < size)

{

while (j < size)

{

a[i][j] = rand() % 100 + 1; // заполняем массив случайными числами

j++;

}

i++;

}

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

i = 0; j = 0;

while (i < size)

{

while (j < size)

{

b[i][j] = rand() % 100 + 1; // заполняем массив случайными числами

j++;

}

i++;

}

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;

}

}

}

return 0;

}

int fun2(int size)

{

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

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

{

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

}

int i=0, j=0;

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

while (i < size)

{

while (j < size)

{

a[i][j] = rand() % 100 + 1; // заполняем массив случайными числами

j++;

}

i++;

}

shell(\*a, size);

clock\_t end = clock();

qs(\*a, 0,size-1);

return end;

}

int fun3(int size)

{

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

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

{

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

}

int i = 0, j = 0;

while (i < size)

{

while (j < size)

{

a[i][j] = i+j+1;

j++;

}

i++;

}

shell(\*a, size);

clock\_t end = clock();

qs(\*a, 0, size - 1);

return end;

}

int fun4(int size)

{

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

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

{

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

}

int i = 0, j = 0;

while (i < size)

{

while (j < size)

{

a[i][j] = size - j-i-1;

j++;

}

i++;

}

shell(\*a, size);

clock\_t end = clock();

qs(\*a, 0, size - 1);

return end;

}

int fun5(int size)

{

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

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

{

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

}

for(int i = 0; i < size/2;i++)

{

for (int j = 0; j < size / 2; j++)

{

a[i][j] = i + j + 1;

}

}

for (int i = size/2; i < size-1; i++)

{

for (int j = size/2; j < size-1; j++)

{

a[i][j] = a[i-1][j-1] - 1;

}

}

shell(\*a, size);

clock\_t end = clock();

qs(\*a, 0, size - 1);

return end;

}

int fun6(int size)

{

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

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

{

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

}

int i = 0, j = 0;

srand(time(NULL)); // инициализируем параметры генератора случайных чисел

while (i < size)

{

while (j < size)

{

a[i][j] = rand() % 100 + 1; // заполняем массив случайными числами

j++;

}

i++;

}

qsort(a, size, sizeof(int), compare);

return 0;

}

int compare(const void\* x1, const void\* x2) // функция сравнения элементов массива

{

return (\*(int\*)x1 - \*(int\*)x2);

}

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) //вызов функции: qs(items, 0, count-1);

{

int i, 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);

}

**Задание 1**

**1. Сложность программы: O(n^3)**



**2.Измерение время выполнения работы (Release)**

**Матрица размером 100х100: 0.001000s**

**Матрица размером 200х200: 0.005000s**

**Матрица размером 400х400: 0.045000s**

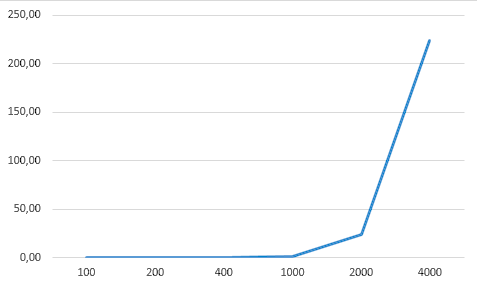
**Матрица размером 1000х1000: 1.338000s**

**Матрица размером 2000х2000: 37.813000s**

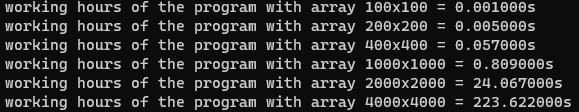
**Матрица размером 4000х4000: 298.407000s**

**Матрица размером 10000х10000: n/a**

**3. График зависимости**



**Результат работы программы**



Задание 2:

Размер 50000

Случайные числа

Shell sort: 0.074000sec

Quick sort: 0.089000sec

Default c sort: 0.111000sec

Возрастающая последовательность чисел

Shell sort: 0.120000sec

Quick sort: 0.114000sec

Default c sort: 0.138000sec

Убывающая последовательность чисел

Shell sort: 0.241000sec

Quick sort: 0.310000sec

Default c sort: 0.259000sec

Половина - убывающая, половина возрастающая последовательность

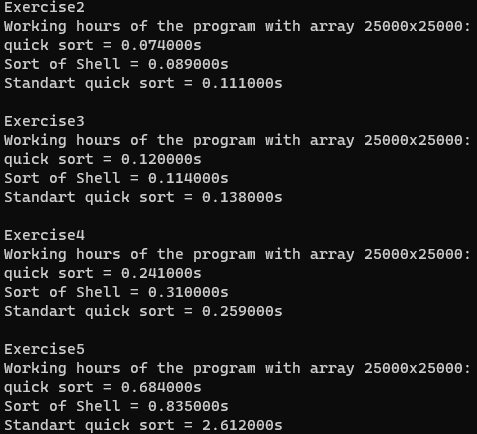
чисел

Shell sort: 0.684000sec

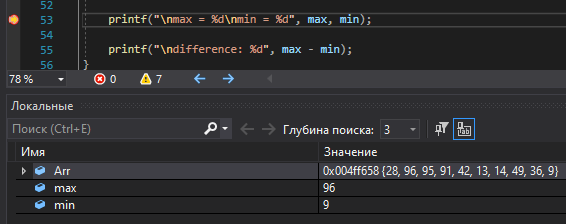
Quick sort: 0.835000sec

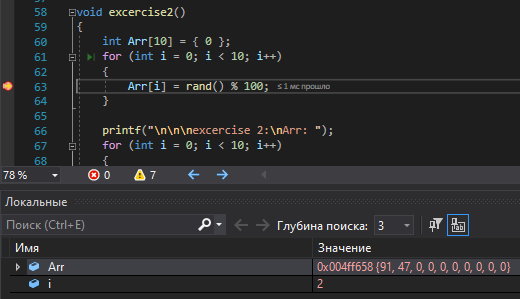
Standart quick sort: 2.612000sec

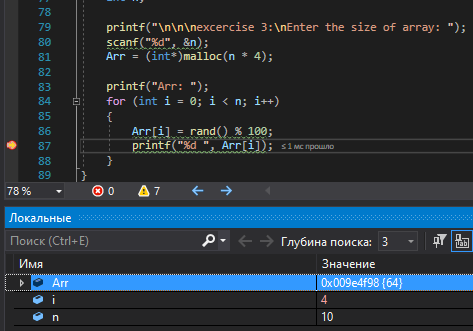
**Результат работы программы**

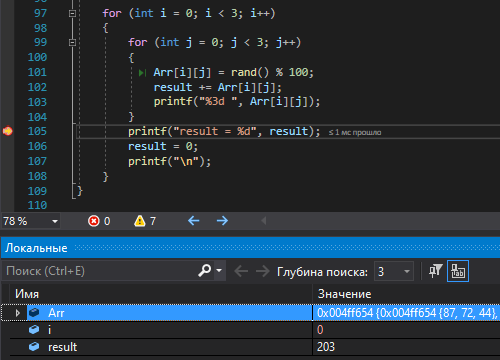


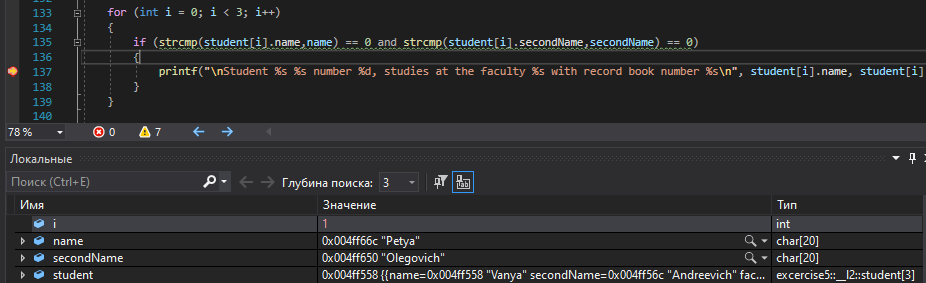
**Протокол трассировки программы**

****

****

****

****

****

**Вывод**

В результате выполнения лабораторной работы я научился оценке времени выполнения программ.