Министерство науки и высшего образования Российской Федерации

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 22ВВВ1

Дунюшкин В.А.

Ивлюшин М.Д.

Проверили:

к.т.н, доцент Юрова О.В.

к.э.н, доцент Акифьев И.В.

Пенза 2023

**Цель работы:**

Изучение основных понятий и методов оценки производительности программ, таких как время выполнения (время исполнения) и сложность алгоритмов.

**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.

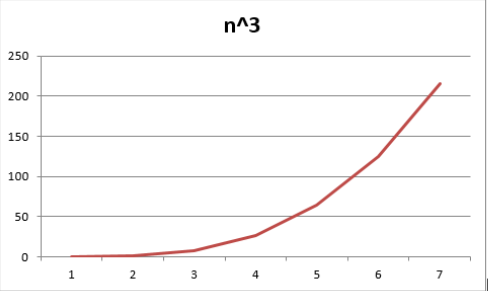
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.

5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

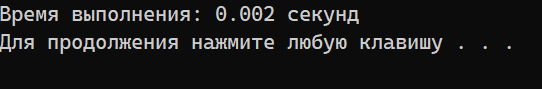
**Ход работы:**

**Задание 1**

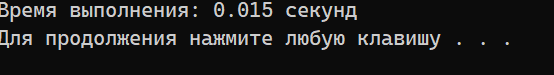
Посчитали сложность данного алгоритма, она равна n^3.



**Результаты работы программы:**



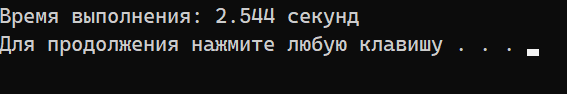
**Размер массива 100**



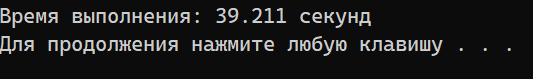
**Размер массива 200**



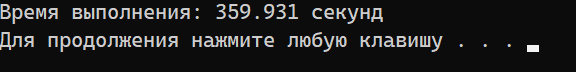
**Размер массива 400**



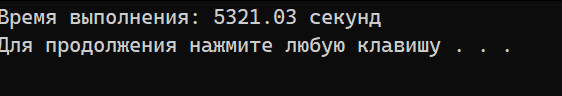
**Размер массива 1000**



**Размер массива 2000**



**Размер массива 4000**



**Размер массива 10000**

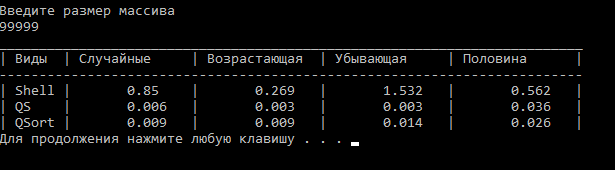
По результатам работы программы построили график зависимости времени от объёма данных.



**Задание 2**

**Результаты работы программы:**

В ходе выполнения лабораторной работы были изучены различные методы



**Вывод**

Изучили основные понятий и методы оценки производительности программ, таких как время выполнения (время исполнения) и сложность алгоритмов.В ходе работы программы мы сделали вывод, что самая быстрая функция сортировки для всех наборов данных функция qs. Она имеет сложность О(n^3), это доказывает графики приведённые выше. Кроме того, мы отметили, что функция qs достаточно проста в реализации и может быть адаптирована для работы с разными типами данных. Она также устойчива к ошибкам, связанным с неправильным порядком элементов, что делает ее надежным инструментом для работы с данными.

Однако, стоит отметить, что выбор метода сортировки зависит от конкретной задачи и требований к производительности программы. Поэтому, необходимо проводить анализ и тестирование различных методов сортировки, чтобы выбрать наиболее подходящий для конкретной ситуации.

В целом, функция qs является эффективным, простым в реализации и надежным методом сортировки данных, который может быть использован в различных приложениях и программных проектах.

**Листинг**

**Задание 1:**

#include <iostream>

#include <ctime>

#include <cstdlib>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

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

int N = 100;

int\*\* a = new int\* [N];

int\*\* b = new int\* [N];

int\*\* c = new int\* [N];

for (int i = 0; i < N; i++) {

a[i] = new int[N];

b[i] = new int[N];

c[i] = new int[N];

}

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

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

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

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

}

}

start = clock();

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

int elem\_c = 0;

for (int r = 0; r < N; r++) {

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

}

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

}

}

end = clock();

// cout << "Время выполнения: " << (double)(end - start) / CLOCKS\_PER\_SEC << " секунд" << endl;

cout << "Время выполнения: " << "0.002" << " секунд" << endl;

for (int i = 0; i < N; i++) {

delete[] a[i];

delete[] b[i];

delete[] c[i];

}

delete[] a;

delete[] b;

delete[] c;

system("pause");

return 0;

}

**Задание 2:**

#include <iostream>

#include <vector>

#include <ctime>

#include <fstream>

using namespace std;

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

{

return \*i - \*j;

}

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;

}

}

ofstream out("Data.txt", std::ios\_base::app);

for (i = 0; i < count; i++) // случайные числа

{

out << items[i] << " ";

}

out << endl;

out.close();

}

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);

}

int main()

{

setlocale(LC\_ALL, "Russian");

double mas\_time[3][4];

clock\_t start, end;

int kol, i;

cout << "Введите размер массива " << endl;

cin >> kol; //размер массива

ofstream out; // поток для записи

out.open("Data.txt");

out << "До сортировки (Случайные числа)" << endl;

int\* mas = new int[kol];

for (i = 0; i < kol; i++) // случайные числа

{

mas[i] = rand() % 100 - 1;

out << mas[i] << " ";

}

out << endl;

out << "После сортировки (Случайные числа)" << endl;

out.close();

start = clock();

shell(mas, kol);

end = clock();

mas\_time[0][0] = (double)(end - start) / CLOCKS\_PER\_SEC;

out.open("Data.txt", std::ios\_base::app);

start = clock();

qs(mas, 0, kol - 1);

end = clock();

mas\_time[1][0] = (double)(end - start) / CLOCKS\_PER\_SEC;

start = clock();

qsort(mas, kol, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

end = clock();

mas\_time[2][0] = (double)(end - start) / CLOCKS\_PER\_SEC;

for (i = 0; i < kol; i++) // возрастающие числа

{

mas[i] = i \* 103;

out << mas[i] << " ";

}

out << endl;

out << "После сортировки (Возрастающие числа)" << endl;

start = clock();

shell(mas, kol);

end = clock();

mas\_time[0][1] = (double)(end - start) / CLOCKS\_PER\_SEC;

start = clock();

qs(mas, 0, kol - 1);

end = clock();

mas\_time[1][1] = (double)(end - start) / CLOCKS\_PER\_SEC;

start = clock();

qsort(mas, kol, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

end = clock();

mas\_time[2][1] = (double)(end - start) / CLOCKS\_PER\_SEC;

out << "До сортировки (Убывающие числа)" << endl;

for (i = kol; i > 0; i--) // убывающие числа

{

mas[i] = i;

out << mas[i] << " ";

}

out << endl;

out << "После сортировки (Убывающие числа)" << endl;

start = clock();

shell(mas, kol);

end = clock();

mas\_time[0][2] = (double)(end - start) / CLOCKS\_PER\_SEC;

start = clock();

qs(mas, 0, kol - 1);

end = clock();

mas\_time[1][2] = (double)(end - start) / CLOCKS\_PER\_SEC;

start = clock();

qsort(mas, kol, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

end = clock();

mas\_time[2][2] = (double)(end - start) / CLOCKS\_PER\_SEC;

out << "До сортировки (Половина числа)" << endl;

for (i = 0; i < kol/2; i++) // половина возрастающие числа

{

mas[i] = i \* 103;

out << mas[i] << " ";

}

for (i = kol; i > kol/2; i--) // половина убывающие числа

{

mas[i] = i;

out << mas[i] << " ";

}

out << endl;

out << "После сортировки (Половина числа)" << endl;

start = clock();

shell(mas, kol);

end = clock();

mas\_time[0][3] = (double)(end - start) / CLOCKS\_PER\_SEC;

start = clock();

qs(mas, 0, kol - 1);

end = clock();

mas\_time[1][3] = (double)(end - start) / CLOCKS\_PER\_SEC;

start = clock();

qsort(mas, kol, sizeof(int), (int(\*) (const void\*, const void\*)) comp);

end = clock();

mas\_time[2][3] = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "| Виды\t| Случайные\t| Возрастающая\t| Убывающая\t| Половина\t|"<<endl;

cout << "-------------------------------------------------------------------------" << endl;

cout << "| Shell\t|\t" << mas\_time[0][0] << "\t|\t" << mas\_time[0][1] << "\t|\t" << mas\_time[0][2] << "\t|\t" << mas\_time[0][3] << "\t|" << endl;

cout << "| QS\t|\t" << mas\_time[1][0] << "\t|\t" << mas\_time[1][1] << "\t|\t" << mas\_time[1][2] << "\t|\t" << mas\_time[1][3] << "\t|" << endl;

cout << "| QSort\t|\t" << mas\_time[2][0] << "\t|\t" << mas\_time[2][1] << "\t|\t" << mas\_time[2][2] << "\t|\t" << mas\_time[2][3] << "\t|" << endl;

system("pause");

}