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 21вв1.1:**

Кузнецов А.С.

Гришин С.В.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания:**

**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.

**Описание метода решения задачи:**

**Задание 1:**

Вычислили сложность программы .Задаём массивы случайными элементами ,запускаем таймер далее производим умножение матриц , останавливаем таймер и выводим время затраченное на умножение матриц. Оцениваем время для всех заданных размеров массива и строим график зависимости времени выполнения программы от размера массива.

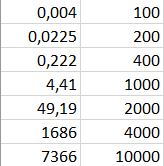
**Задание 2:**

Задаём три сортировки (сортировка шелла , быстрая сортировка и стандартная быстрая сортировка).Оцениваем время работы сортировок на указанный наборах данных.

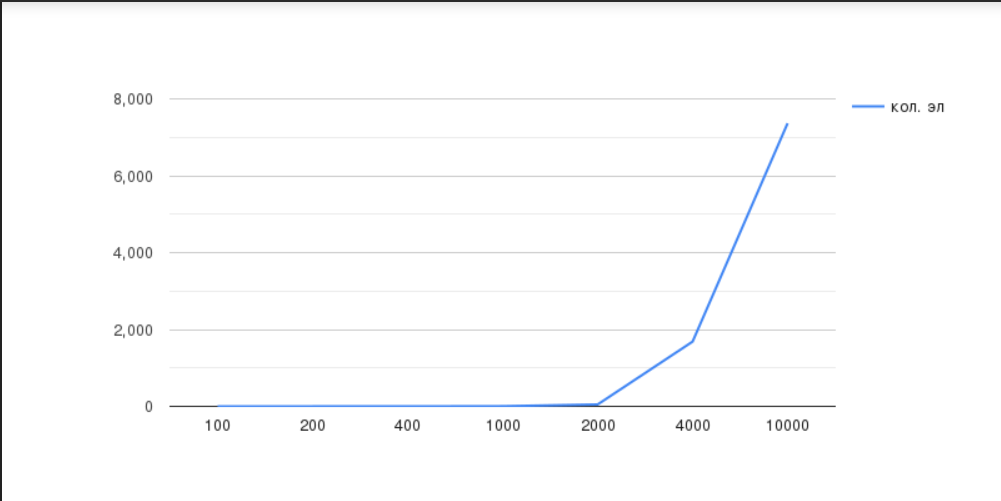
**Задание 1:**

1)Сложность программы – O(n3)

2)Оценили время работы программы на разных размерах матриц



3)Построили график зависимости времени от количества элементов



**Задание 2:** Реализовали сортировки шелла ,быструю сортировку в отдельной функции. Затем поочередно вызывали три сортировки (сортировка шелла , быстрая сортировка , стандартная функция быстрой сортировки) , замеряя время выполнения на разных наборах данных.

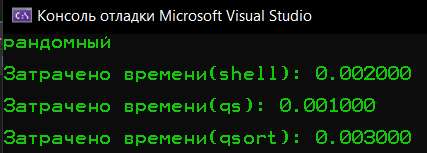
1)Проверили три различные сортировки на

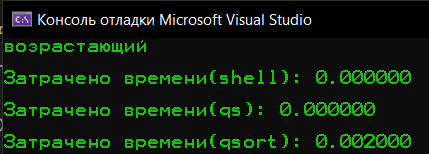
случайных наборах данных;

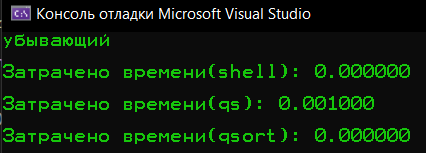
возрастающей последовательности;

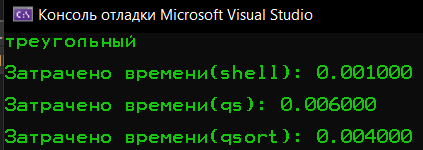
убывающей последовательности;

последовательности ,первая половина которой возрастает , а вторая убывает;









**Задание 1:** На основе построенного графика зависимости времени от количества элементов, сделали вывод о том, что время увеличивается пропорционально размеру массива, а сложность программы: O(n3).

**Задание 2:** Измерив время на разных наборах данных при разных сортировках, мы сделали вывод, что для случайного набора данных эффективней оказалась сортировка qs, для всех остальных наборов данных самой результативной оказалась сортировка шелла.

**Вывод:** В ходе выполнения лабораторной работы были разработаны программа, выполняющая функции с применением статичных и динамических массивов, а также функций. Результаты работы программ совпали с результатами трассировки, следовательно программы работают без ошибок. Также научились измерять время работы отдельных частей программы. Получили опыт в создании проектов в среде MicrosoftVisualStudio, научились писать и отлаживать программы с использованием функций и массивов на языке Си.

**Листинг:**

**Задание 1 :**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main(void)

{

system("chcp 1251");

int i = 0, j = 0, g = 0, col = 7, cch;

int\*\* amass, \*\* bmass, \*\* cmass;

printf("введите размер массивов: ");

scanf\_s("%d", &col);

system("cls");

amass = (int\*\*)malloc(col \* sizeof(int\*));

for (int i = 0; i < col; i++) { amass[i] = (int\*)malloc(col \* 4); }

bmass = (int\*\*)malloc(col \* sizeof(int\*));

for (int i = 0; i < col; i++) { bmass[i] = (int\*)malloc(col \* 4); }

cmass = (int\*\*)malloc(col \* sizeof(int\*));

for (int i = 0; i < col; i++) { cmass[i] = (int\*)malloc(col \* 4); }

srand(time(NULL));

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

{

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

{

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

}

}

srand(time(NULL));

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

{

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

{

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

}

}

clock\_t start = clock();

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

{

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

{

cch = 0;

for (g = 0; g < col; g++)

{

cch = cch + amass[i][g] \* bmass[g][j];

cmass[i][j] = cch;

}

}

}

clock\_t end = clock();

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

printf("\n\n Затрачено времени: %f", ts);

}

**Задание 2:**

#include <iostream>

#include <time.h>

#include <stdlib.h>

void shell(int\* array, int size) {

for (int s = size / 2; s > 0; s /= 2) {

for (int i = s; i < size; ++i) {

for (int j = i - s; j >= 0 && array[j] > array[j + s]; j -= s) {

int temp = array[j];

array[j] = array[j + s];

array[j + s] = temp;

}

}

}

}

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 compare(const void\* x1, const void\* x2)

{

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

}

int main()

{

system("chcp 1251");

int col,j;

int chs;

printf("введите размер: ");

scanf\_s("%d", &col);

system("cls");

printf("вибери тип:\n1)рандомный\n2)возрастающий\n3)убывающий\n4)треугольный\n");

scanf\_s("%d", &chs);

int\* mass = (int\*)malloc(col \* sizeof(int));

if (chs == 1)

{

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

{

mass[i] = rand() % 1000;

}

}

if (chs == 2)

{

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

{

mass[i] = i;

}

}

if (chs == 3)

{

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

{

mass[i] = col - 1;

}

}

if (chs == 4)

{

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

{

mass[i] = i;

j = i;

}

for (int i = j; i < col; i++)

{

j--;

mass[i] = j;

}

}

clock\_t start = clock();

shell(mass, col);

clock\_t end = clock();

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

printf("\nЗатрачено времени(shell): %f\n", ts);

if (chs == 1)

{

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

{

mass[i] = rand() % 1000;

}

}

if (chs == 2)

{

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

{

mass[i] = i;

}

if (chs == 3)

{

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

{

mass[i] = col - 1;

}

}

if (chs == 4)

{

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

{

mass[i] = i;

j = i;

}

for (int i = j; i < col; i++)

{

j--;

mass[i] = j;

}

}

start = clock();

qs(mass, 0, col - 1);

end = clock();

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

printf("\nЗатрачено времени(qs): %f\n", ts);

if (chs == 1)

{

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

{

mass[i] = rand() % 1000;

}

}

if (chs == 2)

{

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

{

mass[i] = i;

}

}

if (chs == 3)

{

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

{

mass[i] = col - 1;

}

}

if (chs == 4)

{

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

{

mass[i] = i;

j = i;

}

for (int i = j; i < col; i++)

{

j--;

mass[i] = j;

}

}

start = clock();

qsort(mass, col,sizeof(int),compare);

end = clock();

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

printf("\nЗатрачено времени(qsort): %f\n", t

}