**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Кафедра «Вычислительная техника»**

**ОТЧЁТ**

**по дисциплине «Л и ОА в ИЗ»**

**по лабораторной работе №2 «Оценка времени выполнения программ»**

Выполнил студент гр. 22ВВВ3:

Гречихин П.П.

Приняли:

к.т.н., доцент Юрова О.В.

к.э.н., доцент Акифьев И.В.

Пенза

2023

**Цель:**

В данной лабораторной работе мы измеряем время выполнения разных алгоритмов, такие как перемножение двух динамических двумерных массивов, сортировки Шелла и быстрой сортировки.

**Общие сведения:**

Для оценки времени выполнения программ языка Си или их частей могут использоваться средства, предоставляемые библиотекой time.h. Данная библиотека содержит описания типов и прототипы функций для работы с датой и временем.

Типы данных:

1. clock\_t - возвращается функцией clock(). Обычно определён как int или long int.

2. time\_t - возвращается функцией time(). Обычно определён как int или long int.

3. struct tm - нелинейное, дискретное календарное представление времени.

Основные функции:

1. clock\_t clock(void) - возвращает время, измеряемое процессором в тактах от начала выполнения программы, или −1, если оно не известно. Пересчет этого времени в секунды выполняется по формуле:

clock() / CLOCKS\_PER\_SEC

где CLOCKS\_PER\_SEC – константа, определяющая количество тактов системных часов в секунду.

2. time\_t time(time\_t \*tp)

Возвращает текущее календарное время или −1, если это время не известно. Если указатель tp не равен NULL, то возвращаемое значение записывается также и в \*tp.

3. double difftime(time\_t time2,time\_t time1)

Возвращает разность time2-time1, выраженную в секундах.

**Практическая часть:**

**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.

**Расчёты:**

**Задание 1:**

|  |  |  |
| --- | --- | --- |
| График времени перемножения двух матриц n-го порядка | | |
| № | n | Время, мС |
| 1 | 100 | 24 |
| 2 | 200 | 57 |
| 3 | 400 | 338 |
| 4 | 1000 | 6034 |
| 5 | 2000 | 105260 |
| 6 | 4000 | 1055871 |

**Задание 2:**

|  |  |  |  |
| --- | --- | --- | --- |
| Метод заполнения | Сортировка Шелла | Быстрая сортировка | |
| Случайными значениями (1 000) | 692 | 601 | 703 |
| Возрастающими значениями (1 000) | 650 | 644 | 744 |
| Убывающими значениями (1 000) | 651 | 635 | 737 |
| Наполовину возрастающими наполовину убывающими значениями (1 000) | 639 | 639 | 742 |
| Случайными значениями (10 000) | 6542 | 6500 | 7557 |
| Возрастающими значениями (10 000) | 7514 | 7600 | 8815 |
| Убывающими значениями (10 000) | 7693 | 7518 | 8863 |
| Наполовину возрастающими наполовину убывающими значениями (10 000) | 9766 | 9766 | 11394 |

Видя полную картину можно сделать вывод - что быстрая сортировка действительно быстрее, но не слишком сильно и только том случае если она написана не через библиотеку.

**Заключение:**

В данной лабораторной работе мы измерили время выполнения разных алгоритмов, такие как перемножение двух динамических двумерных массивов, сортировки Шелла и быстрой сортировки.

**Листинг:**

#include <Windows.h> // для русификации

#include <iostream> // для using namespace std; cout; cin;

#include <cstdlib> // очистка демонической памяти святой водой

#include <time.h> // для секундомера

#include <stdlib.h> // для бастрой сортировки qsort

using namespace std;

void menu();

void task1();

void Shell();

void QuickSort();

void quicksort(int\*, int, int);

void QuickSort2();

int comp(const void\*, const void\*);

int n;

void main() {

srand(time(NULL));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << " Доброго времяни суток пользователь программы!!!" << endl

<< "Эта программа преднозначена для заполнения одинаковых квадратных двумерных" << endl

<< "массивов случайными числами и последующее их перемножение на время," << endl

<< "а также для сравнения быстродействия сортировки Шелла и быстрой сортировки." << endl;

system("pause");

system("cls");

menu();

while (n != 0) {

switch (n) {

case 1:

task1();

system("pause");

system("cls");

menu();

break;

case 2:

Shell();

system("pause");

system("cls");

menu();

break;

case 3:

QuickSort();

system("pause");

system("cls");

menu();

break;

case 4:

QuickSort();

system("pause");

system("cls");

menu();

break;

default:

cout << "Неправильно выбран пункт меню!" << endl;

system("pause");

system("cls");

menu();

break;

}

}

}

void menu() {

cout << "Выберете пинкт меню:" << endl

<< "(1) Создать массивы;" << endl

<< "(2) Сортировка Шелла;" << endl

<< "(3) Быстроя сортировка;" << endl

<< "(4) Быстроя сортировка 2;" << endl

<< "(0) Выход из программы." << endl

<< "Ваш выбор: ";

cin >> n;

system("cls");

}

void task1() {

int a, max = 100, min = 0;

cout << "Квадратная матрица n-го порядка." << endl

<< "n = ";

cin >> a;

int \*\*matrix0, \*\*matrix1, \*\*matrix2; // матрица это миф!!!

matrix0 = new int \*[a]; // массив заполненный указателями

matrix1 = new int \*[a]; // длинна массива = кол-во строк

matrix2 = new int \*[a];

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

matrix0[i] = new int[a]; // массивы готовые к работе

matrix1[i] = new int[a]; // длинна массива = кол-ву столбцов

matrix2[i] = new int[a];

}

clock\_t start, end;

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

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

matrix1[i][j] = min + rand() % (max - min + 1);

matrix2[i][j] = min + rand() % (max - min + 1);

}

}

start = clock();

for (int i = 0, c = 0; i < a; i++) {

for (int j = 0; j < a; j++, c++) {

matrix0[i][j] = 0;

for (int b = 0; b < a; b++) {

matrix0[i][j] += matrix1[i][b] \* matrix2[b][j];

}

cout << "[" << i +1 << "]" << "[" << j + 1 << "]: "

<< "[" << matrix0[i][j] << "]\t";

if (c % 3 == 2) {

cout << endl;

}

}

}

end = clock();

cout << endl << "Программа закончила считать матрицы [" << a << "]["

<< a << "] за " << (end - start) /\*/ CLK\_TCK\*/ << " секунд." << endl;

free(matrix0);

free(matrix1);

free(matrix2);

}

void Shell() {

int count, e, max = 100, min = 0;

cout << "Укажите размер массива: ";

cin >> count;

int \*items = new int[count];

clock\_t start, end;

cout << "Укажите как заполнить массив:" << endl

<< "(1) Рандомными значениями;" << endl

<< "(2) Возрастающими значениями;" << endl

<< "(3) Убывающими значениями;" << endl

<< "(4) Половина возрастающими, половина уменьшающими." << endl

<< "Ваш выбор: ";

cin >> e;

switch (e) {

case 1:

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

items[i] = min + rand() % (max - min + 1);

}

break;

case 2:

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

items[i] = i + 1;

}

break;

case 3:

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

items[i] = count - i;

}

break;

case 4:

for (int i = 0, j = count / 2; i < count; i++) {

if (i < j) {

items[i] = i;

}

if (i >= j) {

items[i] = count - i;

}

}

break;

}

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

cout << "[" << i + 1 << "]" << ": [" << items[i] << "]\t";

if (i % 5 == 4) {

cout << endl;

}

}

start = clock();

for (int i = 5; i > 0; i--) { // радуга, постепенно укорачивается

for (int j = 0; j < i; j++) { // шаг в перёд радуг

for (int b = i + j; b < count; b += i) { // количество радуг

for (int c = b - i; c >= 0 and items[c] > items[c + i]; c -= i) { // сравнение чисел

swap(items[c], items[c + i]); // меняем местами если нужно

}

}

}

}

cout << endl;

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

cout << "[" << i + 1 << "]" << ": [" << items[i] << "]\t";

if (i % 5 == 4) {

cout << endl;

}

}

cout << endl;

end = clock();

cout << "Программа закончила сортировку по методу Шелла за "

<< (end - start) /\*/ CLK\_TCK\*/ << " секунд." << endl;

free(items);

}

void QuickSort() {

int count, e, max = 100, min = 0;

cout << "Укажите размер массива: ";

cin >> count;

int\* items = new int[count];

clock\_t start, end;

cout << "Укажите как заполнить массив:" << endl

<< "(1) Рандомными значениями;" << endl

<< "(2) Возрастающими значениями;" << endl

<< "(3) Убывающими значениями;" << endl

<< "(4) Половина возрастающими, половина уменьшающими." << endl

<< "Ваш выбор: ";

cin >> e;

switch (e) {

case 1:

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

items[i] = min + rand() % (max - min + 1);

}

break;

case 2:

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

items[i] = i + 1;

}

break;

case 3:

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

items[i] = count - i;

}

break;

case 4:

for (int i = 0, j = count / 2; i < count; i++) {

if (i < j) {

items[i] = i;

}

if (i >= j) {

items[i] = count - i;

}

}

break;

}

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

cout << "[" << i + 1 << "]" << ": [" << items[i] << "]\t";

if (i % 5 == 4) {

cout << endl;

}

}

start = clock();

///////////////////////////////////////////////////////////////

int left, right;

left = 0;

right = count - 1;

quicksort(items, left, right);

///////////////////////////////////////////////////////////////

cout << endl;

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

cout << "[" << i + 1 << "]" << ": [" << items[i] << "]\t";

if (i % 5 == 4) {

cout << endl;

}

}

cout << endl;

end = clock();

cout << "Программа закончила сортировку по методу Шелла за "

<< (end - start) /\*/ CLK\_TCK\*/ << " секунд." << endl;

free(items);

}

void quicksort(int\* items, int left, int right) {

int i = left;

int j = right;

int tmp;

int pivot = items[(left + right) / 2];

/\* partition \*/

while (i <= j) {

while (items[i] < pivot)

i++;

while (items[j] > pivot)

j--;

if (i <= j) {

tmp = items[i];

items[i] = items[j];

items[j] = tmp;

i++;

j--;

}

};

/\* recursion \*/

if (left < j)

quicksort(items, left, j);

if (i < right)

quicksort(items, i, right);

}

void QuickSort2() {

int count, e, max = 100, min = 0;

cout << "Укажите размер массива: ";

cin >> count;

int\* items = new int[count];

clock\_t start, end;

cout << "Укажите как заполнить массив:" << endl

<< "(1) Рандомными значениями;" << endl

<< "(2) Возрастающими значениями;" << endl

<< "(3) Убывающими значениями;" << endl

<< "(4) Половина возрастающими, половина уменьшающими." << endl

<< "Ваш выбор: ";

cin >> e;

switch (e) {

case 1:

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

items[i] = min + rand() % (max - min + 1);

}

break;

case 2:

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

items[i] = i + 1;

}

break;

case 3:

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

items[i] = count - i;

}

break;

case 4:

for (int i = 0, j = count / 2; i < count; i++) {

if (i < j) {

items[i] = i;

}

if (i >= j) {

items[i] = count - i;

}

}

break;

}

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

cout << "[" << i + 1 << "]" << ": [" << items[i] << "]\t";

if (i % 5 == 4) {

cout << endl;

}

}

start = clock();

///////////////////////////////////////////////////////////////

int n;

qsort(items, count, sizeof(int), comp);

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

printf("%d ", items[n]);

///////////////////////////////////////////////////////////////

cout << endl;

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

cout << "[" << i + 1 << "]" << ": [" << items[i] << "]\t";

if (i % 5 == 4) {

cout << endl;

}

}

cout << endl;

end = clock();

cout << "Программа закончила сортировку по методу Шелла за "

<< (end - start) /\*/ CLK\_TCK\*/ << " секунд." << endl;

free(items);

}

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

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

}