Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

по лабораторной работе № 4

«Определение среднего времени выполнения обобщенной

элементарной операции методом тестовых прогонов»

Выполнил

ст. гр. И12д

Серегин А.В.

Проверил:

ст.пр. Шишкевич В.Е.

Севастополь

2015

1. Цель работы

Экспериментальная оценка обобщенного времени выполнения элементарных операций методом тестовых прогонов программы. Получение практических навыков оценки времени выполнения элементарных операций в программе на языке высокого уровня.

2. Варианты заданий

Вариант 6:

A: сортировка прямым обменом;

B: количество четных положительных чисел двумерного массива размером n\*n.

3. Структурная схема



Рисунок 1 – Функция main.

4. Код программы

main.cpp

#include <iostream>

#include <conio.h>

#include <Windows.h>

#include "Array.h"

#include "Matrix.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

srand(GetTickCount64());

Array<int> arr;

Matrix<int> matrix;

char key;

int n, m, max;

ULONGLONG StTime,EndTime;

do

{

system("cls");

cout << "1. Заполнение массива случайными числами." << endl;

cout << "2. Заполнение матрицы случайными числами." << endl;

cout << "3. Ввод массива." << endl;

cout << "4. Ввод матрицы." << endl;

if (!arr.Empty())

{

cout << "5. Отобразить массив." << endl;

cout << "6. Сортировать массив по возрастанию." << endl;

cout << "7. Сортировать массив по убыванию." << endl;

}

if (!matrix.Empty())

{

cout << "8. Отобразить матрицу." << endl;

cout << "9. Вывести количество четных

положительных эл-тов матрицы." << endl;

}

cout << "Esc. Выход." << endl;

key = \_getch();

system("cls");

switch (key)

{

case '1':

cout << "Введите длинну массива и макс значение эл-та:" << endl;

cin >> n >> max;

arr.RandArray(n, max);

break;

case '2':

cout << "Введите N, M и макс значение эл-та:" << endl;

cin >> n >> m >> max;

matrix.RandMatrix(n , m, max);

break;

case '3':

cout << "Введите длинну массива:" << endl;

cin >> n;

arr.ScanArray(n);

break;

case '4':

cout << "Введите N, M:" << endl;

cin >> n >> m;

matrix.ScanMatrix(n, m);

break;

case '5':

if (!arr.Empty())

{

cout << "Массив:" << endl;

arr.PrintArray();

cout << endl;

}

break;

case '6':

if (!arr.Empty())

{

cout << "Выполняется сортировка..." << endl;

StTime = GetTickCount64();

arr.SortBubbleLH();

EndTime = GetTickCount64();

system("cls");

cout << "Время выполнения: " << EndTime – StTime

<< "ms." << endl;

cout << "Готово." << endl;

}

break;

case '7':

if (!arr.Empty())

{

cout << "Выполняется сортировка..." << endl;

StTime = GetTickCount64();

arr.SortBubbleHL();

EndTime = GetTickCount64();

system("cls");

cout << "Время выполнения: " << EndTime – StTime

<< "ms." << endl;

cout << "Готово." << endl;

}

break;

case '8':

if (!matrix.Empty())

{

cout << "Матрица:" << endl;

matrix.PrintMatrix();

}

break;

case '9':

if (!matrix.Empty())

{

cout << "Выполняется подсчет четных положительных

элементов..." << endl;

StTime = GetTickCount64();

int count = matrix.EvenPositiveCount();

EndTime = GetTickCount64();

system("cls");

cout << "В матрице " << count

<< " четных положительных эл-тов." << endl;

cout << "Время выполнения: " << EndTime – StTime

<< "ms." << endl;

}

break;

}

system("pause");

} while (key != 27);

return 0;

}

Array.h

#pragma once

#include <vector>

template <typename T>

class Array

{

private:

std::vector<T> arr;

int N;

public:

void RandArray(const int, const int);

void ScanArray(const int);

void PrintArray();

void SortBubbleLH();

void SortBubbleHL();

bool Empty();

int EvenPositiveCount();

~Array();

};

Array.cpp

#include "Array.h"

#include <iostream>

template class Array < int > ;

template <typename T>

void Array<T>::RandArray(const int n, const int rand\_max)

{

N = n;

arr.clear();

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

{

arr.push\_back(rand() % (rand\_max+1));

}

}

template <typename T>

void Array<T>::ScanArray(const int n)

{

N = n;

arr.clear();

int temp;

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

{

std::cin >> temp;

arr.push\_back(temp);

}

}

template <typename T>

bool Array<T>::Empty()

{

if (arr.empty()) return true;

else return false;

}

template <typename T>

void Array<T>::PrintArray()

{

for (auto a : arr)

std::cout << a << ' ';

}

template <typename T>

void Array<T>::SortBubbleLH()

{

for (int j(N - 1); j > 0; j--)

for (int i(0); i < j; i++)

if (arr[i] > arr[i + 1])

{

int x = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = x;

}

}

template <typename T>

void Array<T>::SortBubbleHL()

{

for (int j(N - 1); j > 0; j--)

for (int i(0); i < j; i++)

if (arr[i] < arr[i + 1])

{

int x = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = x;

}

}

template <typename T>

int Array<T>::EvenPositiveCount()

{

int count = 0;

for (T a : arr)

if (a>0 && a % 2 == 0) count++;

return count;

}

template <typename T>

Array<T>::~Array()

{

arr.clear();

}

Matrix.h

#pragma once

#include <vector>

#include "Array.h"

template <typename T>

class Matrix

{

private:

std::vector<Array<T>> matrix;

int N;

int M;

public:

void RandMatrix(const int,const int,const int);

void ScanMatrix(const int, const int);

void PrintMatrix();

bool Empty();

int EvenPositiveCount();

};

Matrix.cpp

#include "Matrix.h"

#include <iostream>

template class Matrix < int >;

template <typename T>

void Matrix<T>::RandMatrix(const int n, const int m, const int max)

{

matrix.clear();

N = n;

M = m;

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

{

Array<T> temp;

temp.RandArray(m, max);

matrix.push\_back(temp);

}

}

template <typename T>

void Matrix<T>::ScanMatrix(const int n, const int m)

{

matrix.clear();

N = n;

M = m;

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

{

Array<T> temp;

temp.ScanArray(m);

matrix.push\_back(temp);

}

}

template <typename T>

void Matrix<T>::PrintMatrix()

{

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

{

matrix[i].PrintArray();

std::cout << std::endl;

}

}

template <typename T>

bool Matrix<T>::Empty()

{

if (matrix.empty()) return true;

else return false;

}

template <typename T>

int Matrix<T>::EvenPositiveCount()

{

int count = 0;

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

count += matrix[i].EvenPositiveCount();

return count;

}

5. Тестовые примеры

На рисунках ниже представлено время выполнения сортировки пузырьком 1000 элементов в 3х случаях и поиск четных положительных элементов в матрице 1000\*1000.

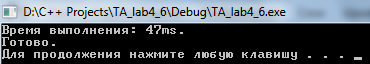


Рисунок 2 – Сортировка в лучшем случае.

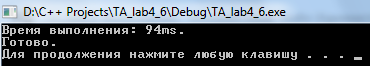


Рисунок 3 – Сортировка в среднем случае.

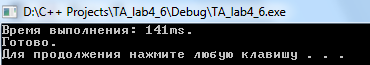


Рисунок 4 – Сортировка в худшем случае.

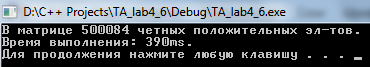


Рисунок 5 – Поиск четных положительных

элементов матрицы.

Выводы

С помощью языков высокого уровня можно легко оценить время выполнения программы. Научились узнавать время начала выполнения функции и конец выполнения, а потом из большего вычитать меньшее в результате получая время выполнения функции.