МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №4**

**з дисципліни «Основи програмної інженерії»**

**на тему: «Класи та об’єкти»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Куроп'ятник О. С.

Дніпро, 2020

**Лабораторна робота №4**

Тема. Класи та об’єкти.

Мета. Ознайомитися з основними принципами та поняттями об’єктно-орієнтованого програмування та отримати практичні навички розробки об’єктно-орієнтованих програм з використанням принципу інкапсуляції.

**Постановка задачі**

1.Загальна постановка: Виконати розробку програми згідно індивідуального завдання лабораторної роботи № 3.

2.Індивідуальна постановка:

16. Дано матрицю, яка заповнена випадковими числами. Виконати циклічний зсув у рядках і стовпцях таким чином, щоб максимальний елемент (один з них) опинився у лівому верхньому кутку.

**Текст програми**

Файл main.cpp

#include <iostream>

#include <fstream>

#include <random>

#include <Windows.h>

#include "Matrix.h"

#include "FileWork.h"

#include "Interface.h"

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int menu;

int max;

int rows, cols;

do {

menu = Interface::options();

switch (menu) {

case 1: {

system("cls");

rows = Interface::input\_check('r');//r-rows

cols = Interface::input\_check('c');//c-cols

Matrix matrix(rows, cols);

matrix.FillByUser();

matrix.output\_matrix();

max = matrix.FindingTheMax\_Min();

int \*\*Array = matrix.getMatrix();

while (true) {

if (Array[0][0] == max) {

Interface::message\_one();

matrix.output\_matrix();

system("pause");

break;

}

else

matrix.MixingElements();

}

break;

}

case 2: {

system("cls");

rows = Interface::input\_check('r');//r-rows

cols = Interface::input\_check('c');//c-cols

Matrix matrix(rows, cols);

matrix.FillByRandom();

matrix.output\_matrix();

max = matrix.FindingTheMax\_Min();

int \*\*Array = matrix.getMatrix();

while (true) {

if (Array[0][0] == max) {

Interface::message\_one();

matrix.output\_matrix();

system("pause");

break;

}

else

matrix.MixingElements();

}

break;

}

case 3: {

system("cls");

FileMatrix arr;

std::ifstream file\_in("array.txt", std::ios::binary);

arr = FileWork::take\_values\_to\_matrix(file\_in);

Matrix matrix(arr.rows, arr.cols);

for (int i = 0; i < arr.rows; i++)

for (int j = 0; j < arr.cols; j++)

matrix.Setij(i, j, arr.matrix[i][j]);

matrix.output\_matrix();

max = matrix.FindingTheMax\_Min();

int \*\*Array = matrix.getMatrix();

while (true) {

if (Array[0][0] == max) {

Interface::message\_one();

matrix.output\_matrix();

system("pause");

break;

}

else

matrix.MixingElements();

}

file\_in.close();

std::ofstream file\_out("redone array.txt", std::ios::binary);

FileWork::write\_values\_to\_file(file\_out, Array, arr.rows, arr.cols);

file\_out.close();

break;

}

case 4:

return 0;

default:

break;

}

} while (menu != 4);

}

Файл Matrix.h

#ifndef \_\_MATRIX\_H\_\_

#define \_\_MATRIX\_H\_\_

class Matrix {

private:

int rows, cols;

int \*\*arr;

int \*\*tmp\_arr;

public:

Matrix(int new\_rows, int new\_cols);

~Matrix();

void Setij(int i, int j, int el);

int\*\* getMatrix();

void FillByUser();

void FillByRandom();

void output\_matrix();

int FindingTheMax\_Min();

void MixingElements();

friend class FileWork;

friend class Interface;

};

#endif

Файл Matrix.cpp

#include <iostream>

#include <random>

#include "FileWork.h"

#include "Matrix.h"

#include "Interface.h"

Matrix::Matrix(int new\_rows, int new\_cols) {

rows = new\_rows;

cols = new\_cols;

arr = new int\*[rows];

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

arr[i] = new int[cols];

}

Matrix::~Matrix() {

delete[] arr;

}

void Matrix::Setij(int i, int j, int el) {

arr[i][j] = el;

}

int\*\* Matrix::getMatrix() {

return arr;

}

void Matrix::FillByUser() {//заповнення масива з клавіатури

for (int i = 0; i < rows; i++)//заповнення масива нулями

for (int j = 0; j < cols; j++)

arr[i][j] = 0;

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

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

std::system("cls");

output\_matrix();

std::cout << "arr[" << i + 1 << "][" << j + 1 << "] = ";

std::cin >> arr[i][j];

}

}

system("cls");

}

void Matrix::FillByRandom() {//заповнення масива випрадковими числами

int Lb, Rb;

Lb = Interface::left\_border();

Rb = Interface::right\_border();

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_int\_distribution<int> dis(Lb, Rb);

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

for (int j = 0; j < cols; j++)

arr[i][j] = dis(gen);

system("cls");

}

void Matrix::output\_matrix() {//друк масива

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

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

std::cout << arr[i][j] << "\t";

}

std::cout << std::endl;

}

}

int Matrix::FindingTheMax\_Min() {//функция знаходження мінімального числа

int max, m = 1;

max = arr[0][0];

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

for (int j = 0; j < cols; j++)

if (arr[i][j] > max)

max = arr[i][j];

return max;

}

void Matrix::MixingElements() {//переміщення елементів вліво на одну позицію

int temp;

temp = arr[0][0];

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

for (int j = 0; j < cols; j++)

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

if (i != rows - 1)

arr[i][cols - 1] = arr[i + 1][0];

}

arr[rows - 1][cols - 1] = temp;

}

Файл FileWork.h

#ifndef \_\_FILEWORK\_H\_\_

#define \_\_FILEWORK\_H\_\_

class Matrix;

struct FileMatrix {

int rows, cols;

int \*\*matrix;

};

class FileWork {

public:

static FileMatrix take\_values\_to\_matrix(std::ifstream& file);

static void write\_values\_to\_file(std::ofstream& file, int \*\*arr, int new\_rows, int new\_cols);

};

#endif

Файл FileWork.cpp

#include <fstream>

#include <iomanip>

#include <iostream>

#include "FileWork.h"

#include "Matrix.h"

FileMatrix FileWork::take\_values\_to\_matrix(std::ifstream& file) {

FileMatrix ar;

file >> ar.rows;

file >> ar.cols;

ar.matrix = new int\*[ar.rows];

for (int i = 0; i < ar.rows; i++)

ar.matrix[i] = new int[ar.cols];

for (int i = 0; i < ar.rows; i++)

for (int j = 0; j < ar.cols; j++)

file >> ar.matrix[i][j];

return ar;

}

void FileWork::write\_values\_to\_file(std::ofstream& file, int \*\*arr, int new\_rows, int new\_cols) {

for (int i = 0; i < new\_rows; i++) {

for (int j = 0; j < new\_cols; j++)

file << arr[i][j] << "\t";

file << std::endl;

}

}

Файл Interface.h

#ifndef \_\_INTERFACE\_H\_\_

#define \_\_INTERFACE\_H\_\_

class Interface {

private:

public:

static int options() {

int menu;

while (true) {

system("cls");

std::cout << "1. Заповнювати матрицю з клавіатури" << std::endl;

std::cout << "2. Випадкове заповнення матриці" << std::endl;

std::cout << "3. Отримати значення з файлу" << std::endl;

std::cout << "4. Вихід" << std::endl;

std::cin >> menu;

if ((std::cin.peek() != '\n') || !(menu > 0)) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else

return menu;

}

}

static void message\_one() {

std::cout << "Перероблений масив:" << std::endl;

}

static int input\_check(char choice) {

int ArrayParameters;

while (true) {

if (choice == 'r')

std::cout << "Введіть кількість рядків: ";

else

std::cout << "Введіть кількість стовпців: ";

std::cin >> ArrayParameters;

if ((std::cin.peek() != '\n') || !(ArrayParameters > 1)) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else

return ArrayParameters;

}

}

static int left\_border() {

int Lb;

while (true) {

std::cout << "Введіть ліву границю: ";

std::cin >> Lb;

if ((std::cin.peek() != '\n')) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else

return Lb;

}

}

static int right\_border() {

int Rb;

while (true) {

std::cout << "Введіть праву границю: ";

std::cin >> Rb;

if ((std::cin.peek() != '\n')) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else

return Rb;

}

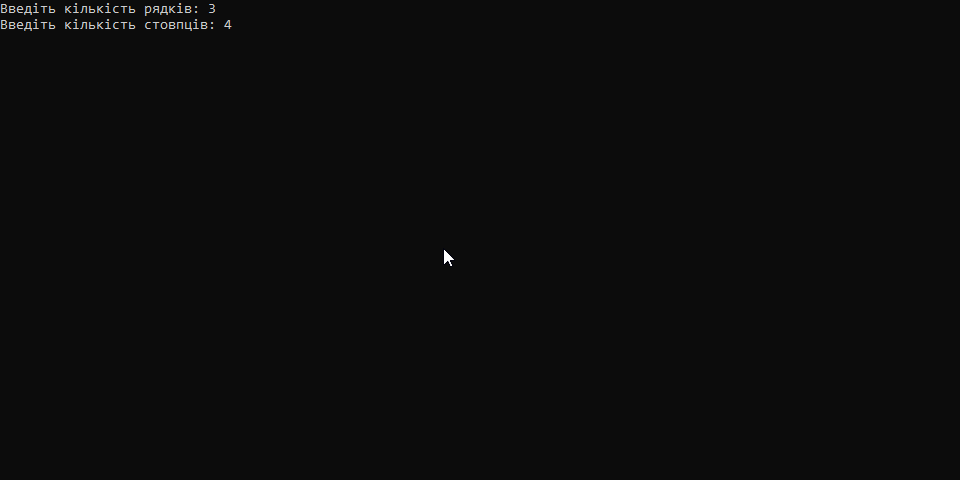
}

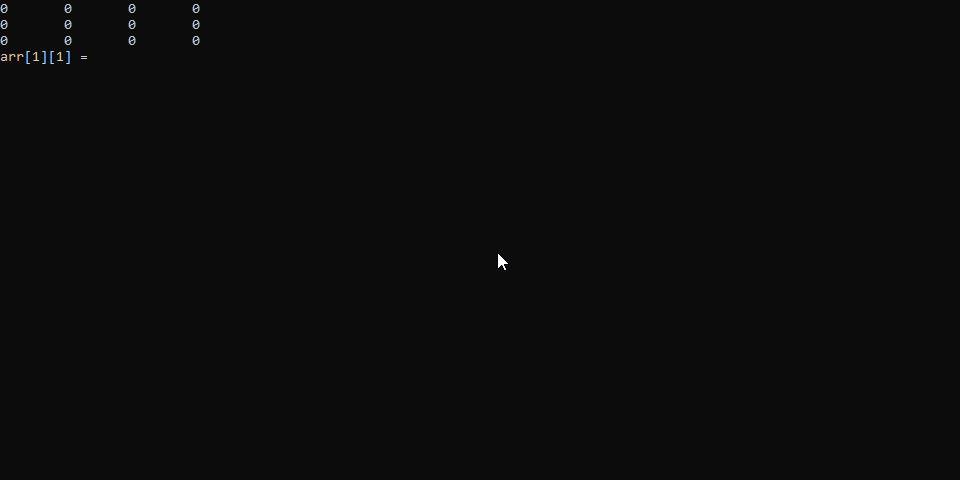
};

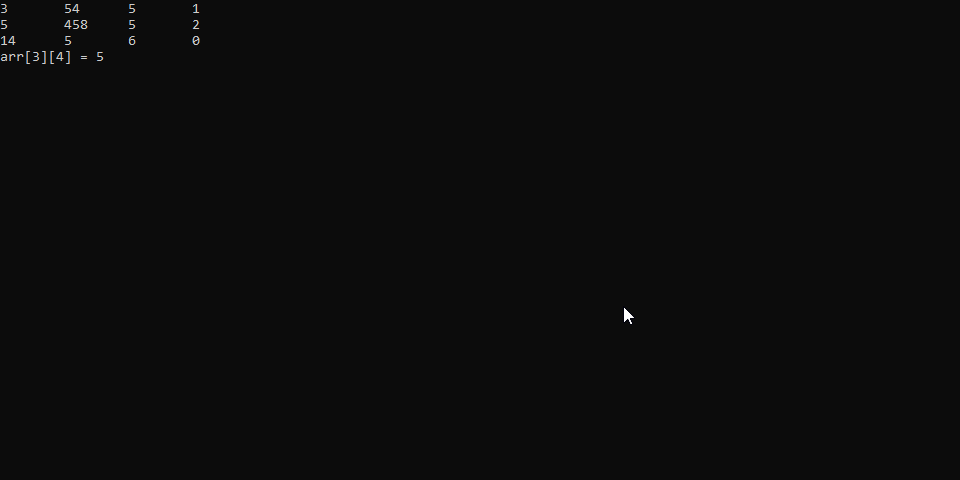
#endif

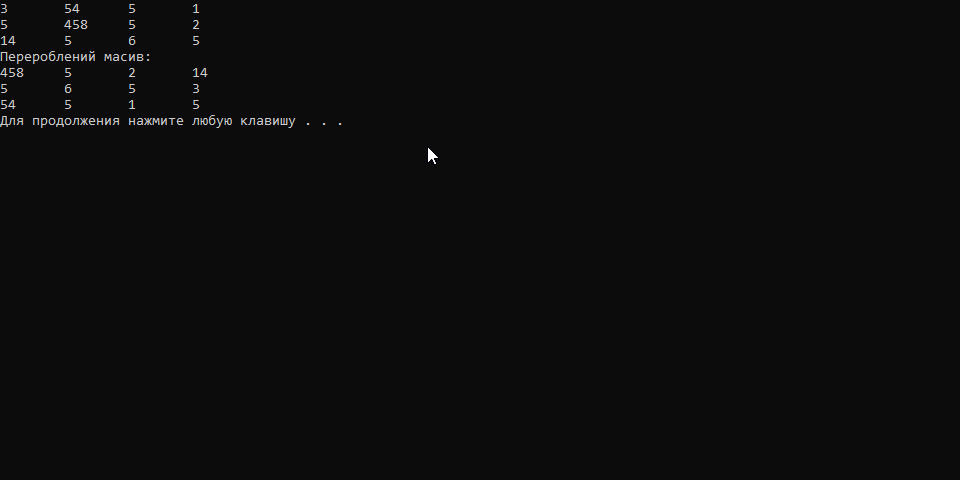
**Результати виповнення програми**

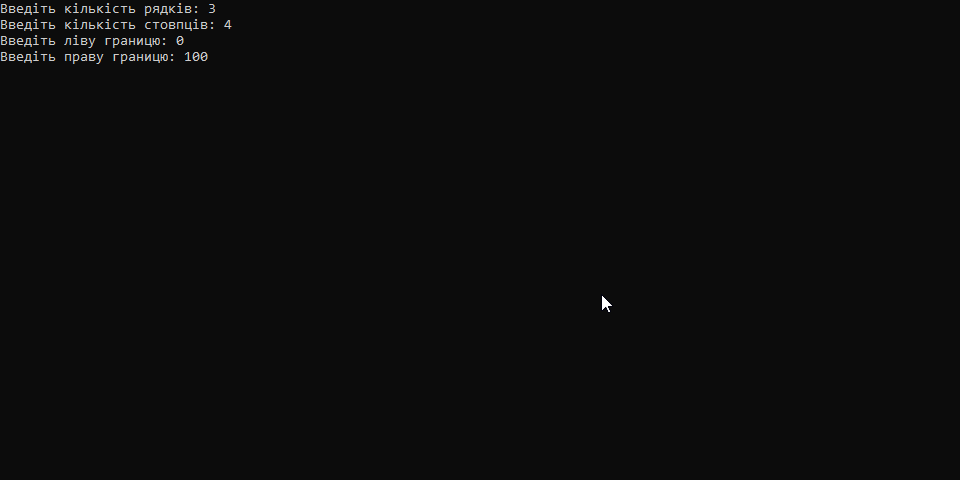
1.Заповнення матриці з клавіатури

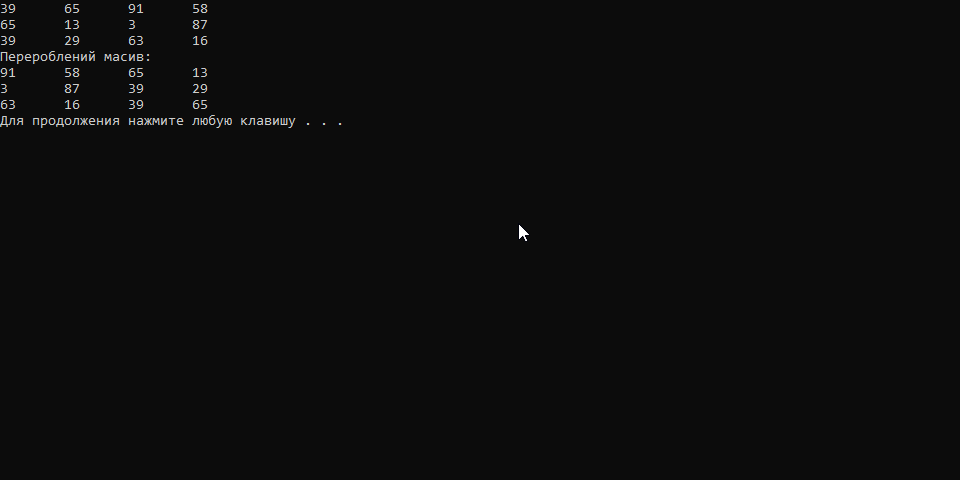




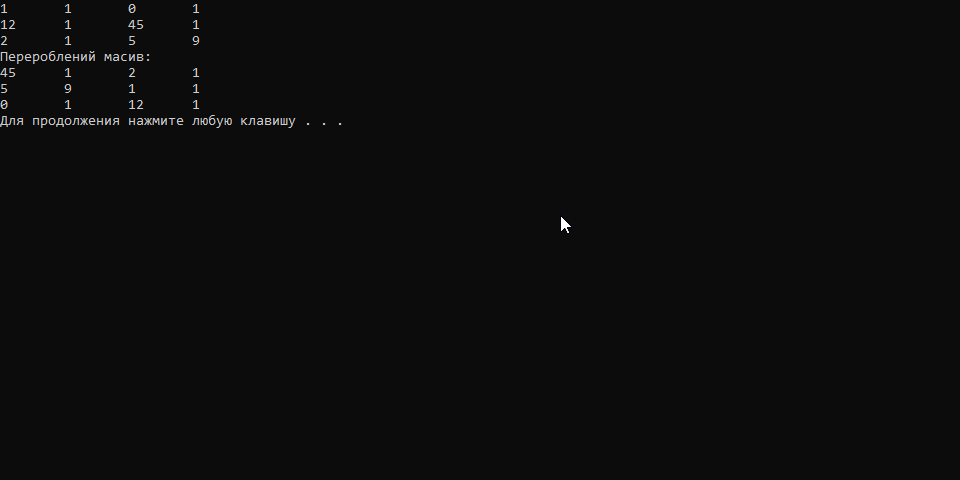
****

****

2.Випадкове заповнення матирці

****

3.Отримання значення з файлу

****

**Висновок**: Переваги ООП:

– неузгоджені параметри підпрограм

– повторного використання існуючого коду для вирішення модифікованої завдання;

– повторне використання і для вирішення інших завдань у даній галузі.

В обох випадках ООП дає переваги. За рахунок використання класів легко модифікувати існуючі елементи без зміни вже готових.

Недоліки ООП:

Ускладнення методології. Застосування об'єктно-орієнтованого підходу вимагає введення додаткових способів подання інформації про предметну область і методів її аналізу. Мова UML включає більше 100 різних умовних позначень. Для успішного використання подібного механізму потрібна наявність певного рівня кваліфікації у фахівців. Для невеликих проектів більш ефективним може виявитися застосування класичних методів розробки. Розробка проектів, для яких найважливішим завданням є опис предметної області, і для яких неможливо знайти людину, яка розуміє цю предметну область в цілому також вимагає використання традиційних підходів, на увазі їх більшої доступності для неспеціалістів.

Складність реалізації. Об'єктно-орієнтовані проекти і їх програмна реалізації на об'єктно-орієнтованої мови, вимагають великих витрат часу і призводять до побудови більш складної і вимогливою до ресурсів програми, ніж класичні методи, які можуть виявитися більш ефективними для деяких завдань.

При об'єктно-орієнтованому підході в якості будівельних блоків використовуються об'єкти, що містять свої власні коди і дані. Структура програм при об'єктно-орієнтованому підході представляється графом взаємодії об'єктів, а не деревом ієрархії, як це має місце в структурному проектуванні.

Об'єктно-орієнтований підхід дає нові властивості об'єктно-орієнтованого підходу, які дозволяють створювати дійсно незалежні від завдання опису предметної області у вигляді набору об'єктів.