Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

**ЗВІТ**

до лабораторної роботи № 9

з дисципліни ООП

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | *ІП-61 Кушка Михайло Олександрович* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | *Головченко М.М.* |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2017

ЗМІСТ

1. Мета роботи 3

2. Постановка задачі 4

3. Аналітичні викладки 5

4. UML-діаграма класів 6

5. Вихідний код програми 7

6. Приклади роботи програми 13

7. Висновки 14

# Мета роботи

Мета роботи - вивчити особливості використання DLL, розглянути принципи статичного і динамічного підключення.

# Постановка задачі

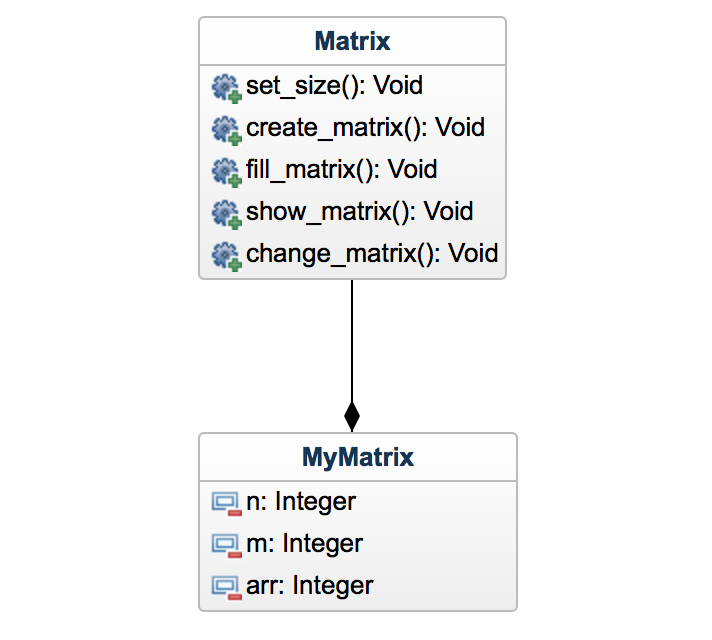
Ввести матрицю розміру m x n. Замінити всі негативні елементи нулем. Підрахувати кількість замін. Вивести вихідну і отриману матриці.

# Аналітичні викладки

DLL (англ. Dynamic Link Library - «бібліотека динамічного компонування», «динамічно підключається») в операційних системах Microsoft Windows і IBM OS/2 - динамічна бібліотека, що дозволяє багаторазове використання різними програмними додатками. До DLL відносяться також елементи управління ActiveX і драйвери. У системах UNIX аналогічні функції виконують так звані загальні об'єкти (англ. shared objects).

Формат файлів DLL дотримується тих самих домовленостей, що і формат виконуваних файлів, поєднуючи код, таблиці і ресурси, відрізняючись лише інтерпретацією деяких полів.

# UML-діаграма класів



# Вихідний код програми

# MatrixDll

**MatrixDll.h**

|  |  |
| --- | --- |
| // MatrixDll.h | |
|  | |  |
|  | #ifdef MATHFUNCSDLL\_EXPORTS | |
|  | #define MATHFUNCSDLL\_API \_\_declspec(dllexport) | |
|  | #else | |
|  | #define MATHFUNCSDLL\_API \_\_declspec(dllimport) | |
|  | #endif | |
|  |  | |
|  | class Matrix { | |
|  | public: | |
|  | virtual void set\_size(int, int) = 0; | |
|  | virtual void create\_matrix() = 0; | |
|  | virtual void fill\_matrix(int min=-9, int max=9) = 0; | |
|  | virtual void show\_matrix() = 0; | |
|  | virtual void change\_matrix() = 0; | |
|  | }; | |

**MatrixDll.cpp**

|  |
| --- |
| // MatrixDll.cpp : Defines the exported functions for the DLL application. |
|  | // |
|  |  |
|  | #include "stdafx.h" |
|  | #include "MatrixDll.h" |
|  |  |
|  | class MyMatrix : public Matrix { |
|  | int n, m; |
|  | int\*\* arr; |
|  | public: |
|  | // MyMatrix constructor |
|  | MyMatrix() {}; |
|  |  |
|  | // MyMatrix destructor |
|  | ~MyMatrix() { |
|  | for (int i = 0; i < n; ++i) { |
|  | delete arr[i]; |
|  | } |
|  | } |
|  |  |
|  | // Set size |
|  | void Matrix::set\_size(int \_n, int \_m) { |
|  | n = \_n; |
|  | m = \_m; |
|  | } |
|  |  |
|  | // Create matrix |
|  | void Matrix::create\_matrix() { |
|  | arr = new int\*[n]; |
|  | for (int i = 0; i < n; ++i) { |
|  | arr[i] = new int[m]; |
|  | } |
|  | } |
|  |  |
|  | // Fill matrix |
|  | void Matrix::fill\_matrix(int min, int max) { |
|  | // Init random |
|  | std::random\_device rd; // only used once to initialise (seed) engine |
|  | std::mt19937 rng(rd()); // random-number engine used (Mersenne-Twister in this case) |
|  | std::uniform\_int\_distribution<int> uni(min,max); // guaranteed unbiased |
|  |  |
|  | for (int i = 0; i < n; ++i) { |
|  | for (int j = 0; j < m; ++j) { |
|  | arr[i][j] = uni(rng); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // Show matrix |
|  | void Matrix::show\_matrix() { |
|  | cout << endl; |
|  | for (int i = 0; i < n; ++i) { |
|  | for (int j = 0; j < m; ++j) { |
|  | cout << setw(4) << arr[i][j] << " "; |
|  | } |
|  | cout << endl; |
|  | } |
|  | cout << endl; |
|  | } |
|  |  |
|  | // Change matrix |
|  | void Matrix::change\_matrix() { |
|  | int counter = 0; |
|  | for (int i = 0; i < n; ++i) { |
|  | for (int j = 0; j < m; ++j) { |
|  | if (arr[i][j] < 0) { |
|  | arr[i][j] = 0; |
|  | counter++; |
|  | } |
|  | } |
|  | } |
|  | cout << "Number of numbers less than zero: " << counter << endl; |
|  | } |
|  | }; |
|  |  |
|  | extern "C" \_\_declspec(dllexport) Matrix\* \_\_cdecl create\_klass() |
|  | { |
|  | return new MyMatrix(); |
|  | } |

**stdafx.h**

|  |
| --- |
| // stdafx.h: включаемый файл для стандартных системных включаемых файлов |
|  | // или включаемых файлов для конкретного проекта, которые часто используются, но |
|  | // не часто изменяются |
|  | // |
|  |  |
|  | #pragma once |
|  |  |
|  | #include "targetver.h" |
|  |  |
|  | #define WIN32\_LEAN\_AND\_MEAN |
|  | #include <windows.h> |
|  | #include <iostream> |
|  | #include <random> |
|  | #include <iomanip> |
|  |  |
|  | using namespace std; |

# MatrixExec

**MatrixExec.cpp**

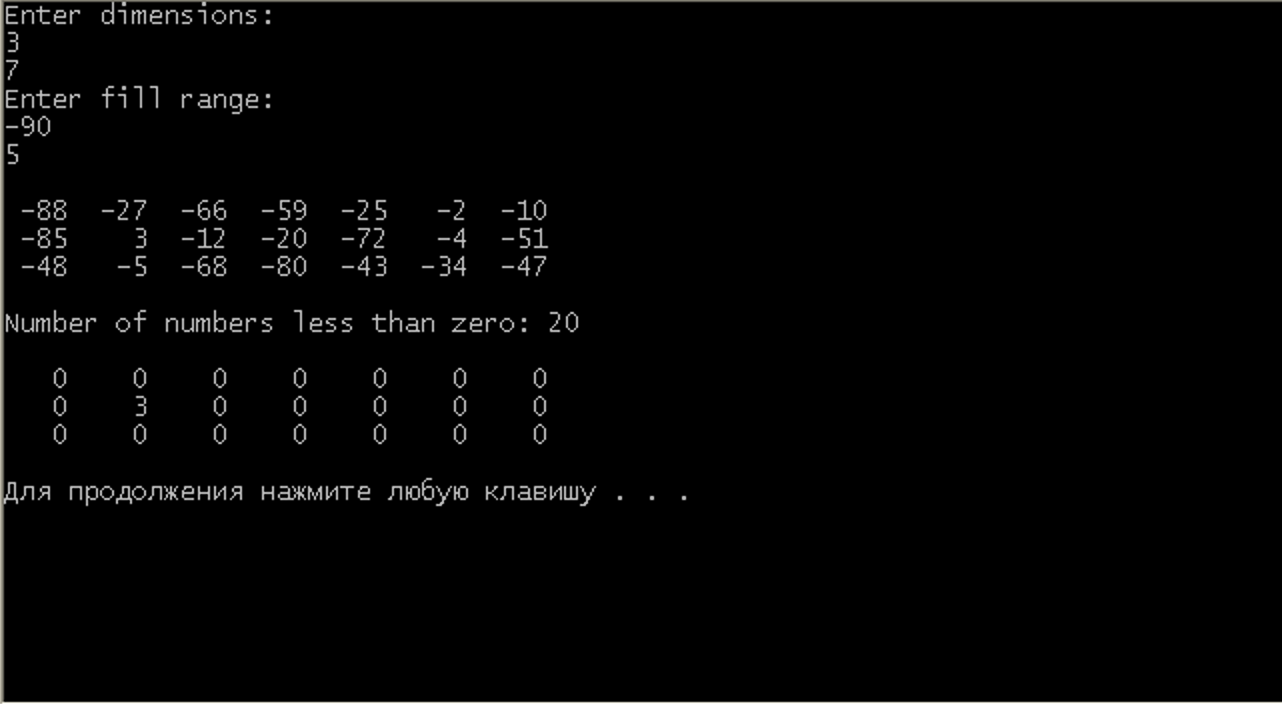
/

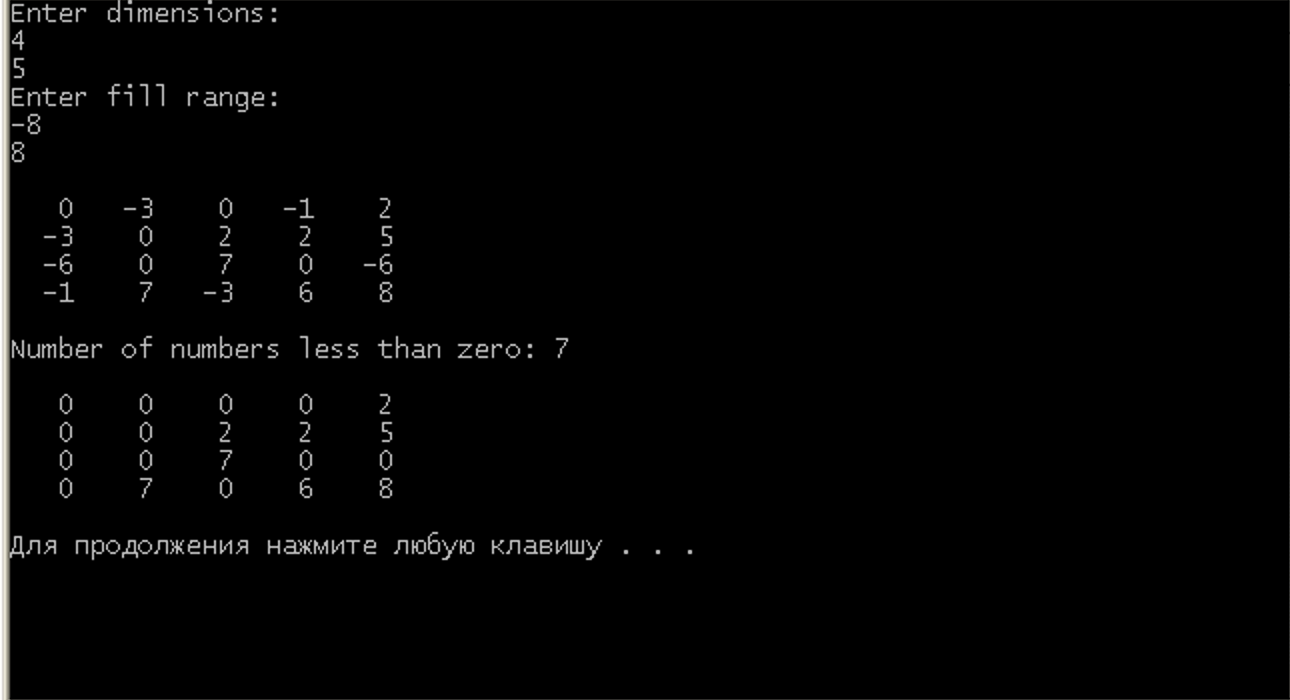
|  |
| --- |
| // MatrixExec.cpp |
|  |  |
|  | #include "stdafx.h" |
|  | #include "MatrixDll.h" |
|  |  |
|  | // A factory of IKlass-implementing objects looks thus |
|  | typedef Matrix\* (\_\_cdecl \*iklass\_factory)(); |
|  |  |
|  | int main() |
|  | { |
|  | // Load the DLL |
|  | HINSTANCE dll\_handle = ::LoadLibrary(TEXT("MatrixDll.dll")); |
|  | if (!dll\_handle) { |
|  | cerr << "Unable to load DLL!\n"; |
|  | return 1; |
|  | } |
|  |  |
|  | // Get the function from the DLL |
|  | iklass\_factory factory\_func = reinterpret\_cast<iklass\_factory>( |
|  | ::GetProcAddress(dll\_handle, "create\_klass")); |
|  | if (!factory\_func) { |
|  | cerr << "Unable to load create\_klass from DLL!\n"; |
|  | ::FreeLibrary(dll\_handle); |
|  | return 1; |
|  | } |
|  |  |
|  | // Ask the factory for a new object implementing the IKlass |
|  | // interface |
|  | Matrix\* instance = factory\_func(); |
|  |  |
|  | string n\_str, m\_str; |
|  | int n, m; |
|  | while (true) { |
|  | try { |
|  | cout << "Please enter size of the matrix" << endl; |
|  | cout << "n: "; |
|  | cin >> n\_str; |
|  | cout << "m: "; |
|  | cin >> m\_str; |
|  |  |
|  | n = stoi(n\_str); |
|  | m = stoi(m\_str); |
|  | if (n < 1 || m < 1) { |
|  | throw invalid\_argument("N and M nust be greater than zero"); |
|  | } |
|  | break; |
|  | } catch (const invalid\_argument &e) { |
|  | cerr << e.what() << endl; |
|  | } catch(...) { |
|  | cerr << "N and M must be integers, not string" << endl; |
|  | } |
|  | } |
|  |  |
|  | // Main part |
|  | instance->set\_size(n, m); |
|  | instance->create\_matrix(); |
|  | instance->fill\_matrix(); |
|  | instance->show\_matrix(); |
|  | instance->change\_matrix(); |
|  | instance->show\_matrix(); |
|  |  |
|  | // Destroy it explicitly |
|  | ::FreeLibrary(dll\_handle); |
|  |  |
|  | return 0; |
|  | } |

## stdafx.h

|  |
| --- |
| // stdafx.h: включаемый файл для стандартных системных включаемых файлов |
|  | // или включаемых файлов для конкретного проекта, которые часто используются, но |
|  | // не часто изменяются |
|  | // |
|  |  |
|  | #pragma once |
|  |  |
|  | #include "targetver.h" |
|  | #include <Windows.h> |
|  | #include <stdio.h> |
|  | #include <iostream> |
|  | #include <string> |
|  | #include <tchar.h> |
|  |  |
|  | using namespace std; |

# Приклади роботи програми





# Висновки

У даній лабораторній роботі я навчився створювати та використовувати власні dll-файли, ознайомився з особливостями їхнього оголошення та реалізації. Розробив власну бібліотеку динамічної компоновки та використав її у своїй програмі.