Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №4 з дисципліни

«Основи програмування-2.

Модульне програмування»

«Перевантаження операторів»

Варіант 19

Виконав студент: ІП-11 Лисенко Андрій Юрійович

Перевірила: Вітковська Ірина Іванівна

Київ 2021

**Лабораторна робота №4**

**Перевантаження операторів**

**Мета роботи**

Вивчити механізми створення класів з використанням перевантажених операторів (операцій)

**Індивідуальне завдання**

**Варіант 19**

Визначити клас "Квадратна матриця 3х3". Реалізувати для нього декілька конструкторів, геттери, метод обчислення визначника матриці. Перевантажити оператори множення "\*" матриць та інкрементації її елементів "++". Створити три матриці (M1, M2, M3), використовуючи різні конструктори. Визначити матрицю М3 як добуток матриць М1 та М2. Інкрементувати елементи отриманої матриці М3. Знайти визначник зміненої матриці М3.

**Постановка задачі**

Створимо клас Matrix3х3 для роботи з матрицями заданої розмірності. Клас матиме константні публічні статичні атрибути SIZE, MIN та MAX, що зберігатимуть розмірність, мінімальний та максимальний елемент під час генерації матриці. Також цей клас матиме декілька конструктів, а саме конструктор за замовчуваням, який генеруватиме матрицю випадковим чином, параметричний, що отримує як параметр масив чисел для заповнення, та конструктор копії. Клас матиме єдиний геттер для елементів матриці. З перевантажених операторів матимемо оператор множення та інкрементації. Також визначимо метод для знаходження визначика 3х3 за допомогою допоміжного методу determinate\_2x2(), цей метод буде викликатися лише членами цього класу або його наслідників, тож визначимо його під специфікатором доступу protected.

**Побудова математичної моделі**

Складемо таблицю змінних

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Розмірність | *int* | *SIZE* | Зберігати розмірність матриць |
| мінімальний, максимальний елемент | *int* | *MIN, MAX* | Зберігати мінімальний, максимальний елемент матриці |
| елементи матриці | *int[][]* | *elements* | зберігати елементи матриці |
| об'єкти матриці 3х3 | *Matrix3x3* | *M1, M2, M3* | Об'єкт класу Matrix3x3, зберігатиме атрибути та методи цього класу |
| визначник | *int* | *det* | Зберігати визначник матриці |
| вектор чисел | *int[]* | *vec* | масив чисел для заповнення матриці для параметричного конструктору |

**Псевдокод**

**Основна програма**

get SIZE, MIN, MAX static attributes of class Matrix3x3

create Matrix3x3 instance: M1

M1.show()

input vector

create Matrix3x3 instance: M2(vector)

M2.show()

create Matrix3x3 instance: M3(M2)

M3 = M1 \* M2

M3.show()

M3++

det = M3.determinate\_3x3()

output M1, det

**Конструктор *Matrix3х3 default***

for row from 0 to size:

for col from 0 to size

elements[row][col] = random from MIN to MAX

**Конструктор *Matrix3х3 parametric (vec)***

***val = 0***

for row from 0 to size:

for col from 0 to size

elements[row][col] = vec(val)

val++

**Конструктор *Matrix3х3 copy (other)***

for row in matrices:

for element in row:

elements[row][col] = other. elements[row][col]

**Оператор \* для *Matrix3x3 (other)***

create Matrix3x3 instance: result

for row from 0 to size:

for col from 0 to size:

result\_value = 0

for k from 0 to size:

result\_value += elements[row][k] \* other.element[k][col]

result.elements[row][col] = result\_value

return result

**Оператор ++ для *Matrix3x3 (notused)***

Create Matrix3x3 instance: temp

for row from 0 to size

for col from 0 to size

++elements[row][col]

return temp

**Метод *Matri3x3 show***

i = 0

for row from 0 to size:

for col from 0 to size:

output elements[row][col]

**Метод *Matrix3x3: get\_element***

return elements[row][col]

**Метод *Matrix3x3: determinant\_3x3***

if size == 3:

det = 0

for col from 0 to 3

for term = elements[0][col] \* determinant\_2x2(0, col)

if col % 2:

det -= term

else:

det += term

return det

else:

output "Cannot find determinant"

**Метод *Matrix3x3: determinant\_2x2 (crossed\_row, crossed\_col)***

minor\_row = 0

for row from 0 to 3:

minor\_col = 0

if row != crossed\_row:

for col from 0 to 3:

if col != crossed\_col:

minor[minor\_row][minor\_col] = elements[row][col]

minor\_row++;

return minor[0][0] \* minor[1][1] - minor[0][1] \* minor[1][0]

**Код на С++**

**main.cpp**

#include <ctime>

#include "matrix\_3x3.h"

int main()

{

srand((unsigned int)time(NULL));

// Get all Matrix3x3 static attributes

const int SIZE = Matrix3x3::SIZE;

const int MIN = Matrix3x3::MIN;

const int MAX = Matrix3x3::MAX;

std::cout << "<======================================================================================>\n";

// Default constructor

std::cout << "<Matrix M1: Initialization by filling matrix with random values in range ["

<< MIN << ", " << MAX << "]>\n\n";

Matrix3x3 M1;

M1.show("Default constructor matrix M1:\n");

std::cout << "<-------------------------------------------------------------------------->\n";

// Parametric constructor

int vec[SIZE \* SIZE];

std::cout << "<Matrix M2: Initialization by filling matrix with a vector>\n\n"

<< "Enter initialization vector (format: [v1] [v2] [v3] ... [v" << SIZE \* SIZE << "])\n";

for (int i = 0; i < SIZE \* SIZE; i++)

std::cin >> vec[i];

Matrix3x3 M2(vec);

M2.show("Parametric constructor matrix M2:\n");

std::cout << "<-------------------------------------------------------------------------->\n";

// Copy constructor

std::cout << "<Matrix M3: Initialization by copying another matrix>\n\n";

Matrix3x3 M3 = M2;

M3.show("Copy constructor matrix M3 = M2:\n");

std::cout << "<======================================================================================>\n";

// Using operator\* overloading

M3 = M1 \* M2;

M3.show("Matrix M3 after multiplication with M1 and M2 using operator\* overloading:\n");

std::cout << "<-------------------------------------------------------------------------->\n";

// Using operator++ overloading

M3++.show("Matrix M3 after incrementation using operator++ overloading:\n");

std::cout << "<-------------------------------------------------------------------------->\n";

// Finding determinate with class method if possible

int det = M3.determinant\_3x3();

std::cout << "Determinate of matrix M3: " << det << "\n\n";

std::cout << "<======================================================================================>\n";

system("pause");

return 0;

}

**matrix.h**

#pragma once

#include <iostream>

#include <iomanip>

#include <string>

#include <cstdlib>

class Matrix3x3

{

public:

// Matrix size

static const int SIZE = 3;

// Matrix element min value

static const int MIN = 0;

// Matrix element max value

static const int MAX = 1000;

private:

int elements[SIZE][SIZE];

protected:

int determinant\_2x2(int row, int col);

public:

// Element getter

int get\_element(int row, int col);

// Constructors

Matrix3x3(); // Default

Matrix3x3(int vec[]); // Parametric

Matrix3x3(const Matrix3x3& other); // Copy

// Operator\* overloading for matrix multiplication

Matrix3x3 operator\*(const Matrix3x3& other);

// Operator++ overloading for incrementing all matrix elements

Matrix3x3 operator++(int notused);

// Find matrix determinate

int determinant\_3x3();

// Output matrix elements to screen

void show (std::string message) const;

};

**matrix.cpp**

#include "matrix\_3x3.h"

Matrix3x3::Matrix3x3()

{

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

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

elements[i][j] = rand() % (MAX - MIN + 1) + MIN;

}

Matrix3x3::Matrix3x3(int vec[])

{

int val = 0;

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

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

elements[i][j] = vec[val++];

}

Matrix3x3::Matrix3x3(const Matrix3x3& other)

{

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

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

elements[i][j] = other.elements[i][j];

}

Matrix3x3 Matrix3x3::operator\*(const Matrix3x3& other)

{

Matrix3x3 result;

int result\_value;

for (int row = 0; row < SIZE; row++)

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

{

result\_value = 0;

for (int k = 0; k < SIZE; k++)

result\_value += elements[row][k] \* other.elements[k][col];

result.elements[row][col] = result\_value;

}

return result;

}

Matrix3x3 Matrix3x3::operator++(int notused)

{

Matrix3x3 temp = \*this;

for (int row = 0; row < SIZE; row++)

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

++elements[row][col];

return temp;

}

int Matrix3x3::get\_element(int row, int col)

{

return elements[row][col];

}

void Matrix3x3::show(std::string message) const

{

unsigned int width\_min = std::to\_string(MIN).length();

unsigned int width\_max = std::to\_string(MAX).length();

unsigned int width = (width\_min > width\_max) ? width\_min + 1 : width\_max + 1;

std::cout << message << std::endl;

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

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

std::cout << std::setw(width) << elements[i][j];

std::cout << std::endl;

}

std::cout << std::endl;

}

int Matrix3x3::determinant\_3x3()

{

if (SIZE == 3) {

int det = 0;

int term;

for (int col = 0; col < 3; col++) {

term = elements[0][col] \* determinant\_2x2(0, col);

if (col % 2)

det -= term;

else

det += term;

}

return det;

}

else

std::cout << "Cannot find 3x3 determinate, SIZE != 3" << "\n\n";

}

int Matrix3x3::determinant\_2x2(int crossed\_row, int crossed\_col)

{

int minor[2][2];

int minor\_row = 0, minor\_col;

for (int row = 0; row < 3; row++) {

minor\_col = 0;

if (row != crossed\_row) {

for (int col = 0; col < 3; col++) {

if (col != crossed\_col) {

minor[minor\_row][minor\_col] = elements[row][col];

minor\_col++;

}

}

minor\_row++;

}

}

return minor[0][0] \* minor[1][1] - minor[0][1] \* minor[1][0];

}

**Висновок**

Підчас виконання цієї роботи, я розробив практичні навички роботи з класами та перевантаженими операторами за допомогою методів класу, а саме створення перевантажених операторів ++ та \* та інкрементації та множення матриць. Я використав статичні атрибути для декількох полів класу, тому що ці атрибути спільні для всіх об'єктів матриці. Також я навчився створювати декілька різних конструкторів класів та визначив геттери для елементів матриць*.* Отже, я вивчив механізми створення класів з використанням перевантажених операторів.