# **Звіт про виконання лабораторної роботи № 2.4**

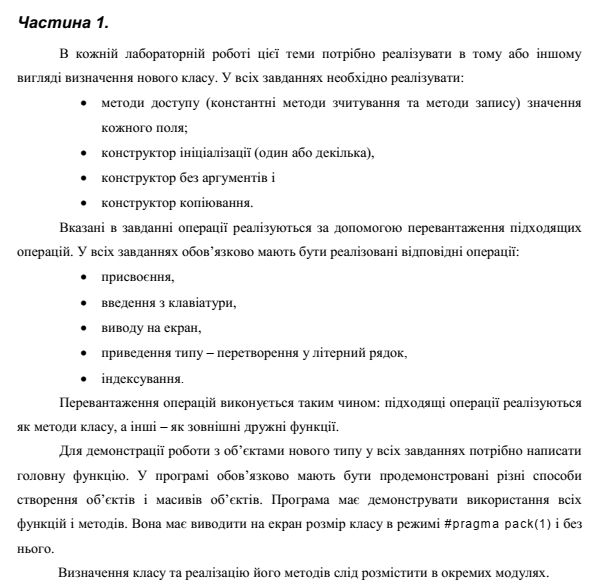
## «Масиви та константи в класі» з дисципліни «Об’єктно-орієнтоване програмування»

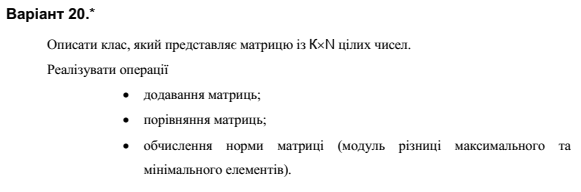
## Студента групи «**ІТ-12**» - **Степанчука Сергія**

## Мета роботи

Освоїти використання масивів та констант в класі.

## Умова завдання

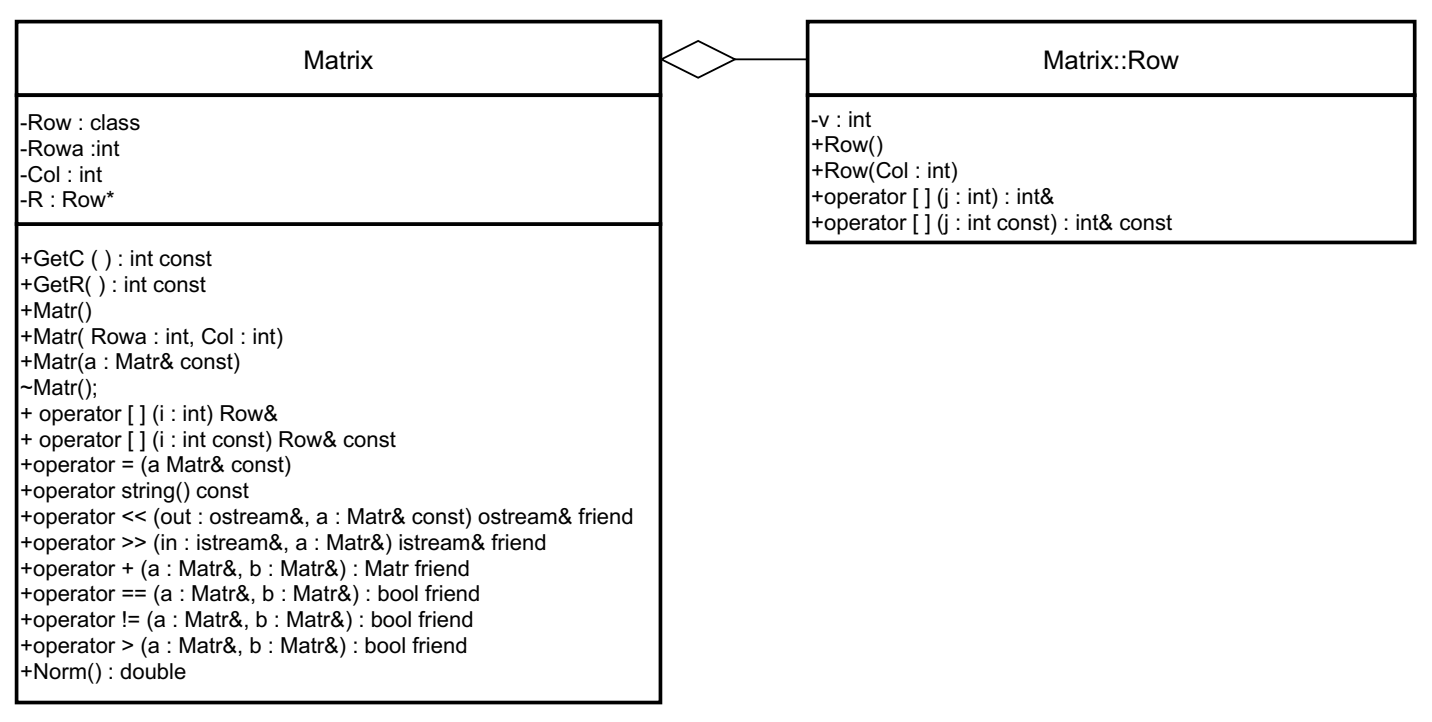




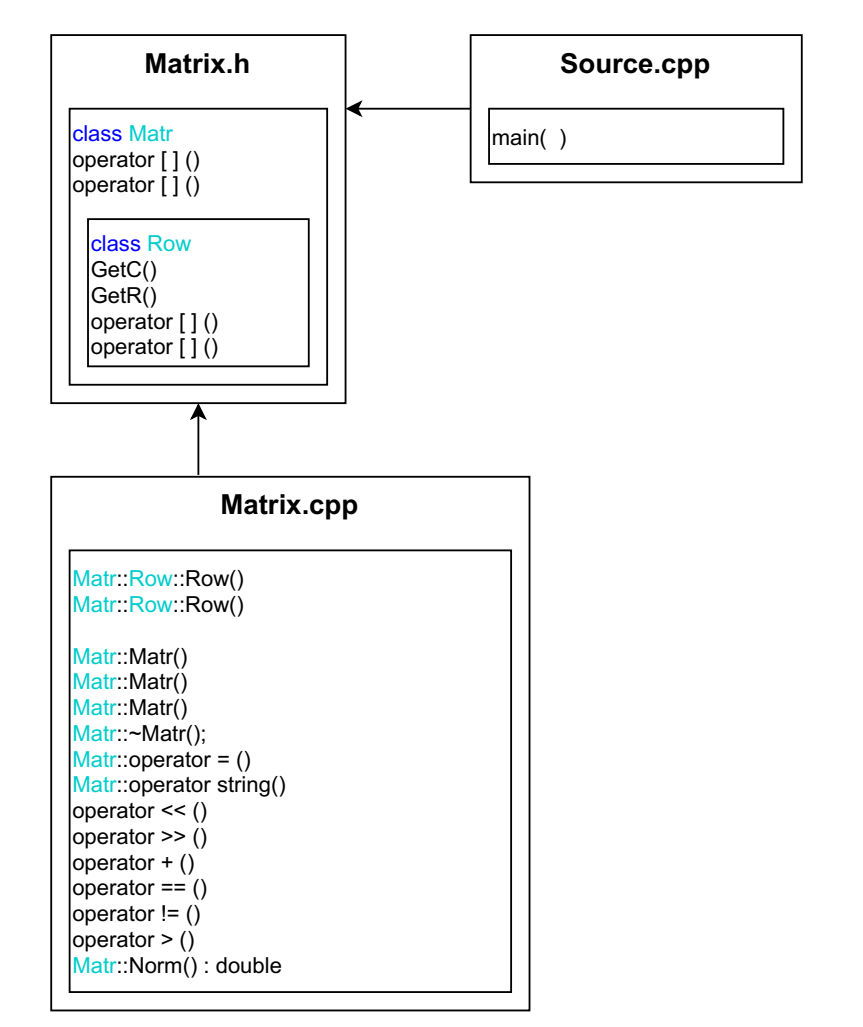
## Посилання на git-репозиторій з проектом:

<https://github.com/SergiyStepanchuk/OOP_Lab_2.4>

## UML-діаграма класів



## Структурна схема



## Текст програми

### // Matrix.h

#pragma once

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

class Matr

{

class Row;

int Rowa, Col;

Row\* r;

class Row

{

public:

int\* v;

Row();

Row(int Col);

int& operator [] (int j) { return v[j]; }

const int& operator [] (const int i) const { return v[i]; }

};

public:

int GetC() const { return Col; }

int GetR() const { return Rowa; }

Matr();

Matr(int Rowa, int Col);

Matr(const Matr& a);

~Matr();

Row& operator [] (int i) { return r[i]; }

const Row& operator [] (const int i) const { return r[i]; }

Matr& operator = (const Matr& a);

operator string() const;

friend ostream& operator << (ostream& out, const Matr& a);

friend istream& operator >> (istream& in, Matr& a);

friend Matr operator + (Matr& a, Matr& b);//Matrix1 + Matrix2

friend bool operator ==(Matr& a, Matr& b);//equal

friend bool operator !=(Matr& a, Matr& b);//size

friend bool operator >(Matr& a, Matr& b);//bigger/smaller

double Norm();

};

### // Matrix.cpp

#include "Matrix.h"

#include <iomanip>

using namespace std;

Matr::Row::Row()

{

v = new int[1];

v[0] = 0;

}

Matr::Row::Row(int Col)

{

v = new int[Col];

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

v[j] = 0;

}

Matr::Matr()

{

Col = 1;

r = new Row(Col);

r[0] = Row(Col);

}

Matr::Matr(int Rowa, int Col)

{

if (Rowa < 1)

this->Rowa = 1;

else

this->Rowa = Rowa;

if (Col < 1)

this->Col = 1;

else

this->Col = Col;

r = new Row[this->Rowa];

for (int i = 0; i < this->Rowa; i++)

{

r[i] = Row(this->Col);

}

}

Matr::Matr(const Matr& a)

{

Rowa = a.Rowa;

Col = a.Col;

r = new Row[this->Rowa];

for (int i = 0; i < this->Rowa; i++)

{

r[i] = Row(this->Col);

}

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

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

r[i][j] = a.r[i][j];

}

Matr:: ~Matr()

{

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

{

if (r[i].v != nullptr)

delete[] r[i].v;

}

if (r != nullptr)

delete[] r;

}

Matr& Matr::operator = (const Matr& a)

{

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

if (r[i].v != nullptr)

{

delete[] r[i].v;

}

if (r != nullptr)

{

delete[] r;

}

Rowa = a.Rowa;

Col = a.Col;

r = new Row[this->Rowa];

for (int i = 0; i < this->Rowa; i++)

{

r[i] = Row(this->Col);

}

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

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

r[i][j] = a.r[i][j];

return \*this;

}

Matr::operator string() const

{

stringstream sout;

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

{

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

{

sout << setw(4) << r[i][k];

}

sout << endl;

}

return sout.str();

}

Matr operator + (Matr& a, Matr& b)

{

Matr tmp(a);

for (int i = 0; i < a.GetR(); i++)

{

for (int k = 0; k < a.GetC(); k++)

{

tmp[i][k] = a[i][k] + b[i][k];

}

}

return tmp;

}

bool operator == (Matr& a, Matr& b)

{

if (a.Col != b.Col || a.Rowa != b.Rowa)

return false;

for (int i = 0; i < a.GetR(); i++)

{

for (int k = 0; k < a.GetC(); k++)

{

if (a[i][k] != b[i][k])

return false;

}

cout << endl;

}

return true;

}

bool operator != (Matr& a, Matr& b)

{

if (a.Col != b.Col || a.Rowa != b.Rowa)

return true;

else

return false;

}

bool operator > (Matr& a, Matr& b)

{

if (a.Col < b.Col || a.Rowa < b.Rowa)

return false;

for (int i = 0; i < a.GetR(); i++)

{

for (int k = 0; k < a.GetC(); k++)

{

if (a[i][k] < b[i][k])

return false;

}

}

return true;

}

ostream& operator << (ostream& out, const Matr& a)

{

out << string(a);

return out;

}

istream& operator >> (istream& in, Matr& a)

{

for (int i = 0; i < a.GetR(); i++)

{

for (int j = 0; j < a.GetC(); j++)

{

cout << "Mas[" << i << "][" << j << "] = "; in >> a[i][j];

}

cout << endl;

}

return in;

}

double Matr::Norm()

{

int b = 0;

for (int i = 0; i < this->GetR(); i++)

{

for (int k = 0; k < this->GetC(); k++)

{

int f = 0;

int e = 0;

if (f > r[i][k])//max

f = r[i][k];

if (e < r[i][k])//min

e = r[i][k];

b = (f - e);

}

cout << endl;

}

return abs(b);

}

### // Source.cpp

#include <iostream>

#include "Matrix.h"

using namespace std;

int main()

{

int Row1, Col1, Row2, Col2;

cout << "Matrix A:" << endl;

cout << "Row = "; cin >> Row1;

cout << "Col = "; cin >> Col1;

Matr a(Row1, Col1);

cin >> a;

cout << "Matrix B:" << endl;

cout << "Row = "; cin >> Row2;

cout << "Col = "; cin >> Col2;

Matr b(Row2, Col2);

cin >> b;

cout << "Matrix A:" << endl;

cout << a << endl;

cout << "Matrix B:" << endl;

cout << b << endl;

if (a != b)

{

cout << "Matrix C(Sum):" << endl;

cout << "Matrix sizes are not the same, so we can not search their sum" << endl;

}

else

{

cout << "Matrix C(Sum):" << endl;

Matr c = a + b;

cout << c << endl;

}

cout << endl;

if (a != b)

{

cout << "Matrix sizes are not the same" << endl;

if (a > b)

cout << "Matrix A is bigger than Matrix B" << endl;

else

cout << "Matrix A is smaller than Matrix B" << endl;

}

else

{

cout << "Matrix sizes are the same" << endl;

if (a == b)

cout << "Matrix are equal" << endl;

else

cout << "Matrix are not equal" << endl;

}

cout << endl;

cout << "Norm (Matrix A) = " << a.Norm() << endl;

cout << "Size of class = " << sizeof(a) << endl;

return 0;

}

## UnitTest

### Код

#include "pch.h"

#include "CppUnitTest.h"

#include "../Lab\_2.4\_/Matrix.h"

#include "../Lab\_2.4\_/Matrix.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest24

{

TEST\_CLASS(UnitTest24)

{

public:

TEST\_METHOD(TestMethod1)

{

Matr A(3,4);

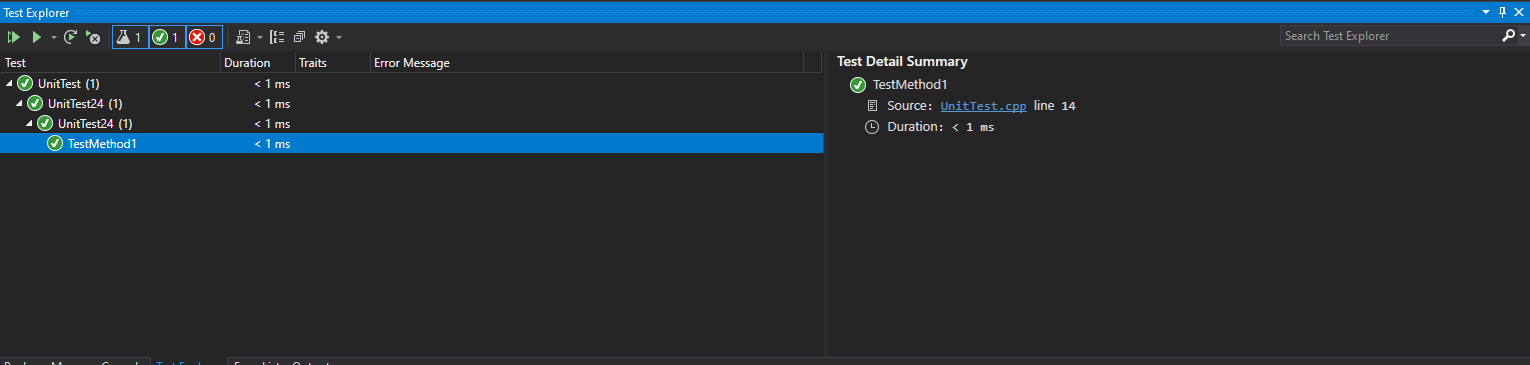
Assert::AreEqual(A.GetC(), 4);

}

};

}

### Результат



## Висновок

Освоїв використання масивів та констант в класі.