

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.  
Метидології програмування»

«Перевантаження операторів»

Варіант 24

Виконав студент ПІ-13 Нещерет Віталій Олександрович  
(шифр, прізвище, ім'я, по батькові)

Перевірів Вечерковська Анастасія Сергіївна  
(прізвище, ім'я, по батькові)

## Лабораторна робота 4

### Перевантаження операторів

#### Варіант 24

Визначити клас "Булева матриця" (BoolMatrix) розмірності  $n \times m$ . Реалізувати для нього декілька конструкторів, геттери, метод підрахунку числа одиниць у матриці. Перевантажити оператори диз'юнкції ( $|$ ) та інверсії ( $\sim$ ) компонент матриць. Створити три булеві матриці ( $M1$ ,  $M2$ ,  $M3$ ), використовуючи різні конструктори. Визначити матрицю  $M3$  як диз'юнкцію булевих матриць  $M1$  та  $M2$  ( $M3 = M1 \vee M2$ ). Знайти інверсію матриці  $M3$ . У отриманій матриці  $M3$  підрахувати число одиниць.

#### Код програми

C++

#### lab\_cpp\_4.cpp

```
#include "BoolMatrix.h"

int main()
{
    srand(time(NULL));
    int rows;
    int cols;
    cout << "Enter the number of rows: ";
    cin >> rows;
    cout << "Enter the number of columns: ";
    cin >> cols;

    BoolMatrix matrix1(rows, cols);
    cout << "First matrix: " << endl;
    matrix1.printMatrix();

    bool** mtr = initMatrix(rows, cols);
    BoolMatrix matrix2(rows, cols, mtr);
    cout << "Second matrix: " << endl;
    matrix2.printMatrix();

    BoolMatrix matrix3 = matrix1 | matrix2;
    cout << "Third matrix (matrix1 | matrix2): " << endl;
    matrix3.printMatrix();

    matrix3 = ~matrix3;
    cout << "Third matrix (~matrix3): " << endl;
    matrix3.printMatrix();

    cout << "Number of 1 in matrix is " << matrix3.countTrue() << endl;

    deleteMatrix(mtr, rows);
    return 0;
}
```

## BoolMatrix.h

```
#pragma once
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

class BoolMatrix {
private:
    int rows;
    int cols;
    bool** matrix;
public:
    BoolMatrix(int, int);
    BoolMatrix(int, int, bool**);
    BoolMatrix(BoolMatrix&);
    ~BoolMatrix();
    int getRows();
    int getCols();
    bool** getMatrix();
    int countTrue();
    void printMatrix();
    BoolMatrix& operator~();
    BoolMatrix& operator|(BoolMatrix&);
};

bool** initMatrix(int, int);
void deleteMatrix(bool**, int);
```

## BoolMatrix.cpp

```
#include "BoolMatrix.h"

BoolMatrix::BoolMatrix(int rws, int cls) {
    rows = rws;
    cols = cls;
    matrix = initMatrix(rws, cls);
}

BoolMatrix::BoolMatrix(int rws, int cls, bool** matr) {
    rows = rws;
    cols = cls;
    matrix = new bool* [rows];
    for (int i = 0; i < rows; i++)
    {
        matrix[i] = new bool[cols];
        for (int j = 0; j < cols; j++)
        {
            matrix[i][j] = matr[i][j];
        }
    }
}

BoolMatrix::BoolMatrix(BoolMatrix& other) {
    rows = other.rows;
    cols = other.cols;
    matrix = new bool* [rows];
    for (int i = 0; i < rows; i++)
    {
        matrix[i] = new bool[cols];
        for (int j = 0; j < cols; j++)
```

```

        {
            matrix[i][j] = other.matrix[i][j];
        }
    }
}

BoolMatrix::~~BoolMatrix() {
    deleteMatrix(matrix, rows);
}

int BoolMatrix::getRows() { return rows; }

int BoolMatrix::getCols() { return cols; }

bool** BoolMatrix::getMatrix() { return matrix; }

int BoolMatrix::countTrue() {
    int res = 0;
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            if (matrix[i][j])
            {
                res++;
            }
        }
    }
    return res;
}

void BoolMatrix::printMatrix() {
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            cout << matrix[i][j] << '\t';
        }
        cout << endl;
    }
    cout << endl;
}

BoolMatrix& BoolMatrix::operator~() {
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            matrix[i][j] = !matrix[i][j];
        }
    }
    return *this;
}

BoolMatrix& BoolMatrix::operator|(BoolMatrix& other) {
    BoolMatrix& res = *this;
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            res.matrix[i][j] = matrix[i][j] | other.matrix[i][j];
        }
    }
    return res;
}

```

```

bool** initMatrix(int rows, int cols) {
    bool** matrix = new bool* [rows];
    for (int i = 0; i < rows; i++)
    {
        matrix[i] = new bool[cols];
    }

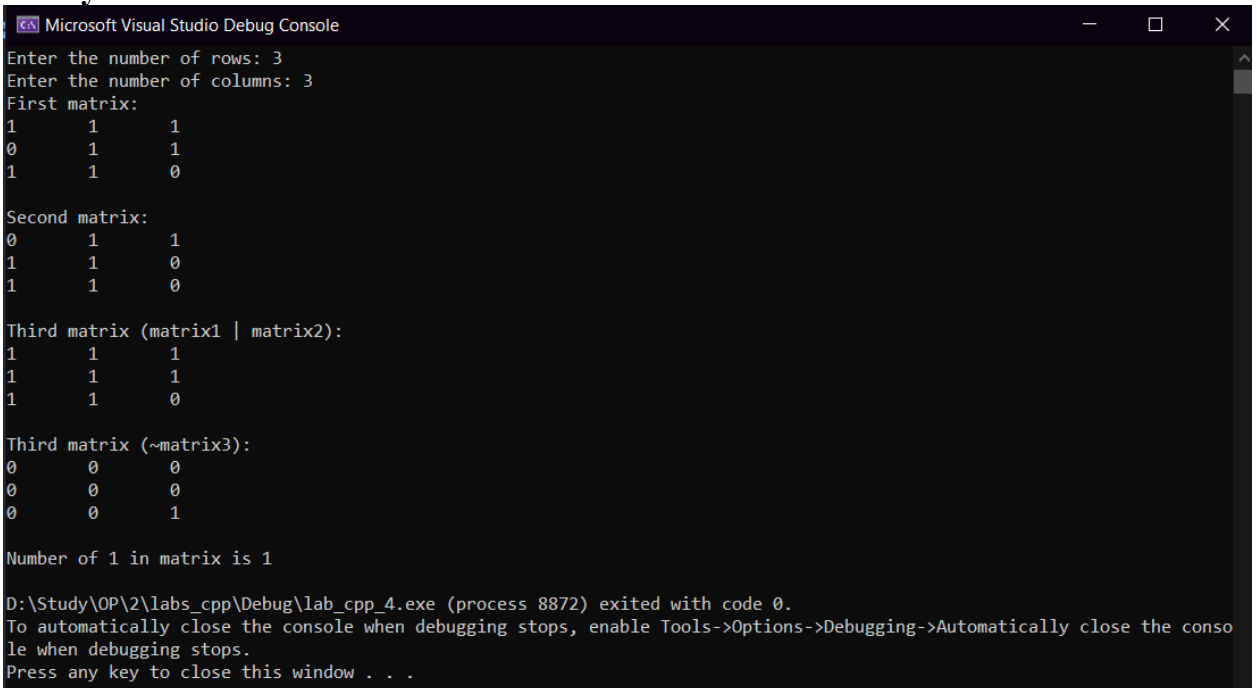
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            matrix[i][j] = rand() % 2;
        }
    }

    return matrix;
}

void deleteMatrix(bool** mtr, int rows) {
    for (int i = 0; i < rows; i++)
    {
        delete[] mtr[i];
    }
    delete[] mtr;
}

```

## Тестування:



```

Microsoft Visual Studio Debug Console
Enter the number of rows: 3
Enter the number of columns: 3
First matrix:
1      1      1
0      1      1
1      1      0

Second matrix:
0      1      1
1      1      0
1      1      0

Third matrix (matrix1 | matrix2):
1      1      1
1      1      1
1      1      0

Third matrix (~matrix3):
0      0      0
0      0      0
0      0      1

Number of 1 in matrix is 1

D:\Study\OP\2\labs_cpp\Debug\lab_cpp_4.exe (process 8872) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

## Висновки:

Я вивчив механізми створення і використання класів з використанням перевантажених операторів. Застосував ці навички на практиці.