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 24

Виконав студент ІП-11 Печковський Олександр Костянтинович

(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна

(прізвище, ім'я, по батькові)

Київ 2022

**Умова задачі:**

Визначити клас "Булева матриця" (BoolMatrix) розмірностi nxm. Реалізувати для нього декілька конструкторів, геттери, метод підрахунку числа одиниць у матриці. Перевантажити оператори диз'юнкцiï ("|") та інверсії ("~") компонент матриць. Створити три булеві матриці (М1, М2, М3), використовуючи різні конструктори. Визначити матрицю М3 як диз'юнкцію булевих матриць М1 та М2 (М3 = М1 V М2). Знайти інверсію матриці М3. У отриманій матриці М3 підрахувати число одиниць.

**Код на С++:**

**Lab4C++.cpp**

#include "header.h"

int main()

{

int m, n;

cout << "Enter m: ";

cin >> m;

cout << "Enter n: ";

cin >> n;

cout << "Matrix 1:" << endl;

BoolMatrix M1 = BoolMatrix(m, n);

output(M1, m, n);

std::this\_thread::sleep\_for(std::chrono::nanoseconds(1000000000));

cout << "Matrix 2:" << endl;

BoolMatrix M2 = BoolMatrix(m, n);

output(M2, m, n);

BoolMatrix M3;

M1.set\_m(m);

M1.set\_n(n);

M2.set\_m(m);

M2.set\_n(n);

M3.set\_m(m);

M3.set\_n(n);

cout << "Matrix 3:" << endl;

M3 = M1 | M2;

output(M3, m, n);

cout << "Matrix 3 inversion:" << endl;

M3 = ~M3;

output(M3, m, n);

M3.count1();

cout << "Matrix 3 inversion contains " << M3.getCounter() << " numbers 1" << endl;

}

**Header.h**

#pragma once

#include <iostream>

#include <thread>

using namespace std;

class BoolMatrix

{

private:

int m, n, \*\* M, \*\*temp, counter=0;

public:

BoolMatrix();

BoolMatrix(int m, int n);

BoolMatrix operator|(BoolMatrix M2);

BoolMatrix operator~();

void set\_m(int m\_);

void set\_n(int n\_);

void count1();

int\*\* getM();

int getCounter();

};

void output(BoolMatrix, int, int);

**functions.cpp**

#include "header.h"

BoolMatrix::BoolMatrix()

{

}

BoolMatrix::BoolMatrix(int m, int n)

{

M = new int\* [m];

srand(time(NULL));

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

{

M[i] = new int[n];

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

{

M[i][j] = rand() % 2;

}

}

}

BoolMatrix BoolMatrix::operator|(BoolMatrix M2)

{

temp = new int\* [m];

srand(time(NULL));

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

{

temp[i] = new int[n];

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

{

if (M[i][j] == 0 && M2.getM()[i][j] == 0)

{

temp[i][j] = 0;

}

else

{

temp[i][j] = 1;

}

}

}

M = temp;

return \*this;

}

BoolMatrix BoolMatrix::operator~()

{

temp = new int\* [m];

srand(time(NULL));

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

{

temp[i] = new int[n];

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

{

if (M[i][j] == 0)

{

temp[i][j] = 1;

}

else

{

temp[i][j] = 0;

}

}

}

M = temp;

return \*this;

}

int\*\* BoolMatrix::getM()

{

return M;

}

int BoolMatrix::getCounter()

{

return counter;

}

void BoolMatrix::set\_m(int m\_)

{

m = m\_;

}

void BoolMatrix::set\_n(int n\_)

{

n = n\_;

}

void BoolMatrix::count1()

{

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

{

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

{

if (M[i][j] == 1)

{

counter++;

}

}

cout << endl;

}

}

void output(BoolMatrix M, int m, int n)

{

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

{

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

{

cout.width(6);

cout << M.getM()[i][j];

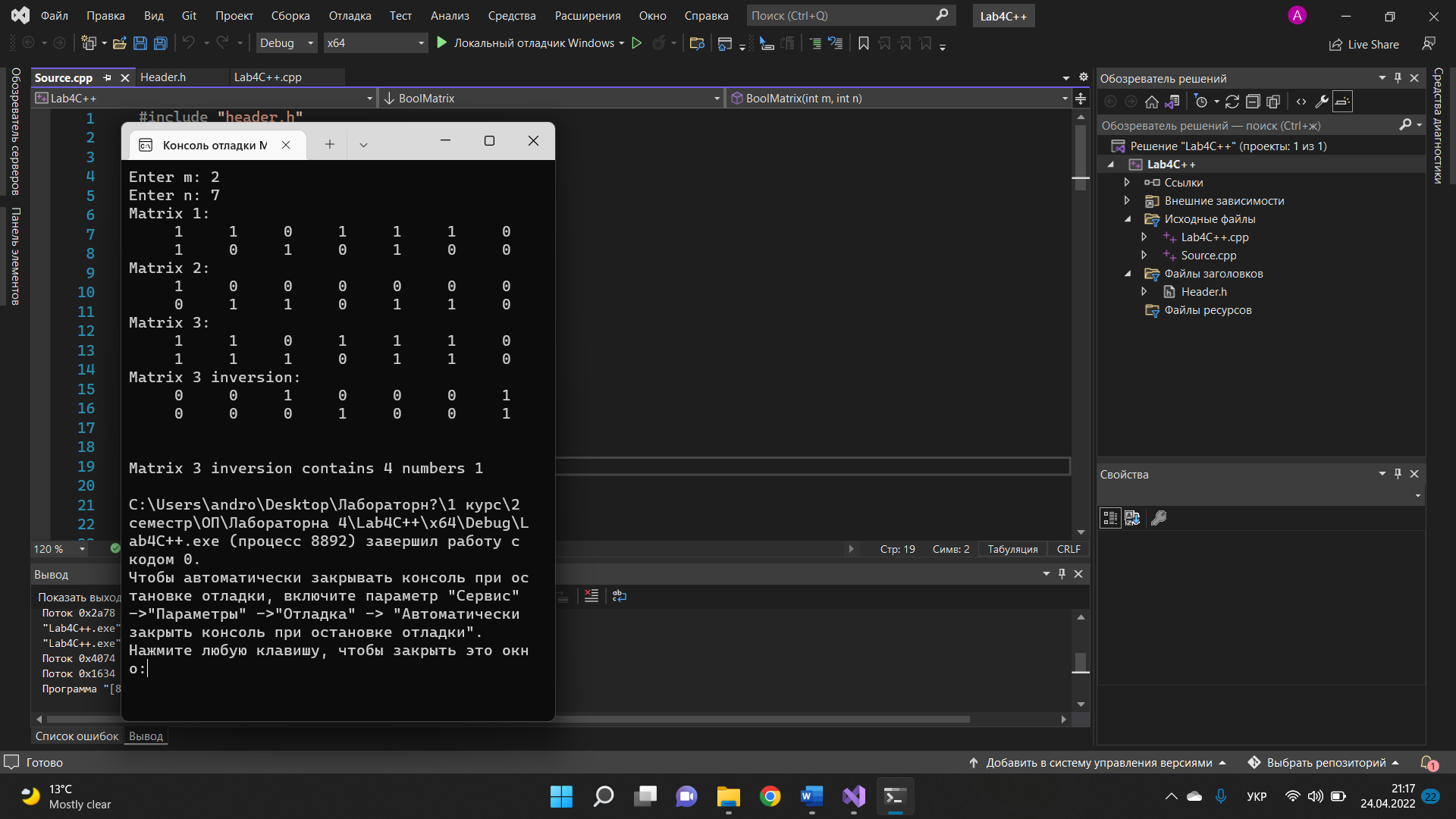
}

cout << endl;

}

}

**Скріншот роботи програми на C++:**

****

**Висновок:** під час виконання лабораторної роботи я вивчив механізми створення класів з використанням перевантажених операторів (операцій).