**Лабораторна робота №2**

## Тема: Перевантаження операцій.

**Мета***.* ***Одержати практичні навички створення абстрактних типів даних і перевантаження операцій у мові С++.***

Завдання 1.

#pragma once

#include <iostream>

#include <string>

using namespace std;

class IntMatrix

{

private:

int\*\* IntArray;

int n;

public:

IntMatrix(int size);

IntMatrix(const IntMatrix& other);

IntMatrix& operator=(const IntMatrix& other);

~IntMatrix();

void input();

void output() const;

int sumOfColumn(int col) const;

int getZeroCount() const;

void setMainDiagonal(int value);

IntMatrix& operator++();

IntMatrix operator++(int);

IntMatrix& operator--();

IntMatrix operator--(int);

operator bool() const;

IntMatrix operator+(const IntMatrix& other) const;

operator string() const;

static IntMatrix fromString(const string& str);

};

#include "IntMatrix.h"

#include <iomanip>

#include <sstream>

IntMatrix::IntMatrix(int size) : n(size)

{

IntArray = new int\* [n];

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

IntArray[i] = new int[n] {};

}

IntMatrix::IntMatrix(const IntMatrix& other) : n(other.n)

{

IntArray = new int\* [n];

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

IntArray[i] = new int[n];

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

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

}

}

IntMatrix& IntMatrix::operator=(const IntMatrix& other)

{

if (this == &other) return \*this;

for (int i = 0; i < n; ++i) delete[] IntArray[i];

delete[] IntArray;

n = other.n;

IntArray = new int\* [n];

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

IntArray[i] = new int[n];

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

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

}

return \*this;

}

IntMatrix::~IntMatrix()

{

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

delete[] IntArray[i];

delete[] IntArray;

}

void IntMatrix::input()

{

cout << "Enter elements of the matrix (" << n << "x" << n << "):\n";

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

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

cout << "Element [" << i << "][" << j << "]: ";

cin >> IntArray[i][j];

}

}

void IntMatrix::output() const

{

cout << "Matrix elements:\n";

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

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

cout << setw(5) << IntArray[i][j];

cout << endl;

}

}

int IntMatrix::sumOfColumn(int col) const

{

int sum = 0;

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

sum += IntArray[i][col];

return sum;

}

int IntMatrix::getZeroCount() const

{

int count = 0;

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

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

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

++count;

return count;

}

void IntMatrix::setMainDiagonal(int value)

{

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

IntArray[i][i] = value;

}

IntMatrix& IntMatrix::operator++()

{

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

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

++IntArray[i][j];

return \*this;

}

IntMatrix IntMatrix::operator++(int)

{

IntMatrix temp(\*this);

++(\*this);

return temp;

}

IntMatrix& IntMatrix::operator--()

{

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

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

--IntArray[i][j];

return \*this;

}

IntMatrix IntMatrix::operator--(int)

{

IntMatrix temp(\*this);

--(\*this);

return temp;

}

IntMatrix::operator bool() const

{

return n > 0;

}

IntMatrix IntMatrix::operator+(const IntMatrix& other) const

{

if (n != other.n) {

throw invalid\_argument("Matrix sizes do not match for addition");

}

IntMatrix result(n);

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

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

result.IntArray[i][j] = IntArray[i][j] + other.IntArray[i][j];

return result;

}

IntMatrix::operator string() const

{

ostringstream oss;

oss << n << "\n";

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

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

oss << IntArray[i][j] << " ";

oss << "\n";

}

return oss.str();

}

IntMatrix IntMatrix::fromString(const string& str)

{

istringstream iss(str);

int size;

iss >> size;

IntMatrix m(size);

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

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

iss >> m.IntArray[i][j];

return m;

}

#include <iostream>

#include "IntMatrix.h"

using namespace std;

int main()

{

IntMatrix m1(2), m2(2);

cout << "Fill matrix m1:\n";

m1.input();

cout << "Fill matrix m2:\n";

m2.input();

cout << "\nMatrix m1:\n"; m1.output();

cout << "\nMatrix m2:\n"; m2.output();

cout << "\nApplying ++ to m1:\n";

++m1;

m1.output();

cout << "\nApplying -- to m2:\n";

m2--;

m2.output();

cout << "\nMatrix m1 + m2:\n";

IntMatrix sum = m1 + m2;

sum.output();

if (m1) cout << "\nm1 is square!\n";

string str = string(m1);

cout << "\nMatrix m1 as string:\n" << str << endl;

IntMatrix m3 = IntMatrix::fromString(str);

cout << "\nMatrix m3 (restored from string):\n";

m3.output();

return 0;

}

**Висновки:** У процесі виконання роботи було набуте практичне вміння створювати абстрактні типи даних у мові C++ та застосовувати механізм перевантаження операторів.