Лабораторна робота 4

1.

#include <iostream>

#include <iomanip>

using namespace std;

class Matrix

{

public:

static int matrix\_count;

string error\_state = "";

Matrix(unsigned int \_n = 3)

{

matrix\_count++;

n = \_n;

m = \_n;

matrix = new int\* [n];

if (matrix == nullptr)

{

error\_state = "Недостатньо пам'яті";

return;

}

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

{

matrix[i] = new int[m];

if (matrix[i] == nullptr)

{

error\_state = "Недостатньо пам'яті";

return;

}

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

{

matrix[i][j] = 0;

}

}

}

Matrix(unsigned int \_n, unsigned int \_m, int value)

{

matrix\_count++;

n = \_n;

m = \_m;

matrix = new int\* [n];

if (matrix == nullptr)

{

error\_state = "Недостатньо пам'яті";

return;

}

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

{

matrix[i] = new int[m];

if (matrix[i] == nullptr)

{

error\_state = "Недостатньо пам'яті";

return;

}

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

{

matrix[i][j] = value;

}

}

}

Matrix(const Matrix& other)

{

matrix\_count++;

n = other.n;

m = other.m;

matrix = new int\* [n];

if (matrix == nullptr)

{

error\_state = "Недостатньо пам'яті";

return;

}

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

{

matrix[i] = new int[m];

if (matrix[i] == nullptr)

{

error\_state = "Недостатньо пам'яті";

return;

}

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

{

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

}

}

}

Matrix operator=(const Matrix& other)

{

n = other.n;

m = other.m;

if (matrix)

{

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

delete[]matrix[i];

delete[]matrix;

}

matrix = new int\* [n];

if (matrix == nullptr)

{

error\_state = "Недостатньо пам'яті";

return Matrix();

}

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

{

matrix[i] = new int[m];

if (matrix[i] == nullptr)

{

error\_state = "Недостатньо пам'яті";

return Matrix();

}

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

{

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

}

}

}

~Matrix()

{

matrix\_count--;

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

{

delete[]matrix[i];

}

delete[]matrix;

}

void set(unsigned int i, unsigned int j, int value)

{

if (i >= n || j >= m)

{

error\_state = "Індекс поза діапазоном";

return;

}

matrix[i][j] = value;

}

int get(unsigned int i, unsigned int j)

{

if (i >= n || j >= m)

{

error\_state = "Індекс поза діапазоном";

return 1;

}

return matrix[i][j];

}

unsigned int rows()

{

return n;

}

unsigned int columns()

{

return m;

}

void randomize(int a, int b)

{

a \*= 100; b \*= 100;

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

{

for (unsigned int j = 0; j < m; j++)

{

matrix[i][j] = ((rand() \* rand()) % (b - a) + a) / 100;

}

}

}

Matrix operator+(Matrix other)

{

Matrix result(n, m, 0);

if (n != other.n || m != other.m)

{

result.error\_state = "Розмір матриці недійсний";

return result;

}

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

{

for (unsigned int j = 0; j < m; j++)

{

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

}

}

return result;

}

Matrix operator-(Matrix other)

{

Matrix result(n, m, 0);

if (n != other.n || m != other.m)

{

result.error\_state = "Розмір матриці недійсний";

return result;

}

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

{

for (unsigned int j = 0; j < m; j++)

{

result.matrix[i][j] = matrix[i][j] - other.matrix[i][j];

}

}

return result;

}

Matrix operator\*(int x)

{

Matrix result(n, m, 0);

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

{

for (unsigned int j = 0; j < m; j++)

{

result.matrix[i][j] = matrix[i][j] \* x;

}

}

return result;

}

bool operator==(Matrix other)

{

if (n != other.n || m != other.m)

return false;

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

{

for (unsigned int j = 0; j < m; j++)

{

if (matrix[i][j] != other.matrix[i][j])

return false;

}

}

return true;

}

bool operator!=(Matrix other)

{

return !(\*this == other);

}

bool operator>(Matrix other)

{

if (n != other.n || m != other.m)

return false;

int count = 0;

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

{

for (unsigned int j = 0; j < m; j++)

{

if (matrix[i][j] > other.matrix[i][j])

count++;

}

}

return (count == n \* m);

}

bool operator<(Matrix other)

{

if (n != other.n || m != other.m)

return false;

int count = 0;

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

{

for (unsigned int j = 0; j < m; j++)

{

if (matrix[i][j] < other.matrix[i][j])

count++;

}

}

return (count == n \* m);

}

friend ostream& operator<<(ostream& out, Matrix other)

{

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

{

for (int j = 0; j < other.m; j++)

{

out << fixed;

out << setprecision(2) << setw(8) << other.matrix[i][j];

}

out << endl;

}

return out;

}

private:

unsigned int n, m;

int\*\* matrix;

};

int Matrix::matrix\_count = 0;

int main()

{

srand(time(nullptr));

Matrix m1(3), m2(3);

Matrix\* m3 = new Matrix();

m1.randomize(1, 10);

m2.randomize(1, 10);

m3->randomize(1, 10);

cout << "Matrix1:\n" << m1 << endl;

cout << "Matrix2:\n" << m2 << endl;

cout << "Matrix3:\n" << \*m3 << endl;

cout << "Сума матриць Matrix1 і Matrix2:\n" << m1 + m2 << endl;

cout << "Різниця матриць Matrix1 і Matrix2:\n" << m1 - m2 << endl;

cout << "Matrix1 \* 2:\n" << m1 \* 2 << endl;

cout << "Matrix1 < (Matrix1 \* 2): " << boolalpha << (m1 < (m1 \* 2)) << endl;

cout << "Matrix1 > (Matrix1 \* 2): " << boolalpha << (m1 > (m1 \* 2)) << endl;

cout << "Matrix1 == Matrix1: " << boolalpha << (m1 == m1) << endl;

cout << "Matrix1 == Matrix2: " << boolalpha << (m1 == m2) << endl;

cout << "Matrix1 != Matrix1: " << boolalpha << (m1 != m1) << endl;

cout << "Matrix1 != Matrix2: " << boolalpha << (m1 != m2) << endl;

m1.set(0, 0, 100);

cout << "Встановіть Matrix1[0][0] як 100 і отримайте його: " << m1.get(0, 0) << endl;

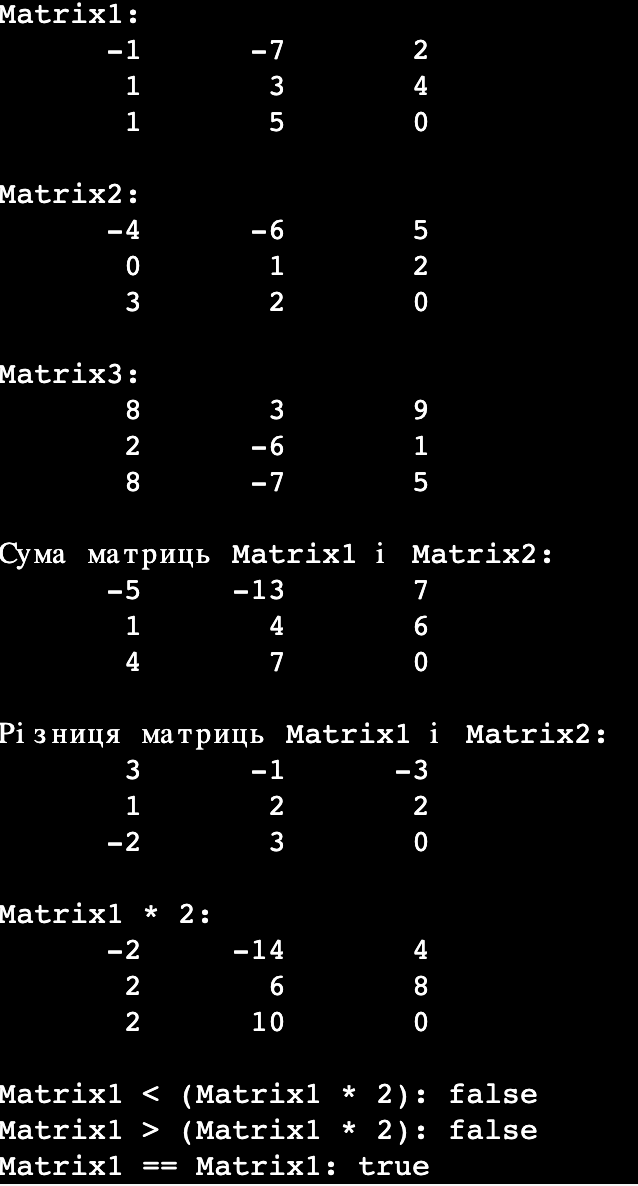
cout << "Кількість створених матриць: " << Matrix::matrix\_count << endl;

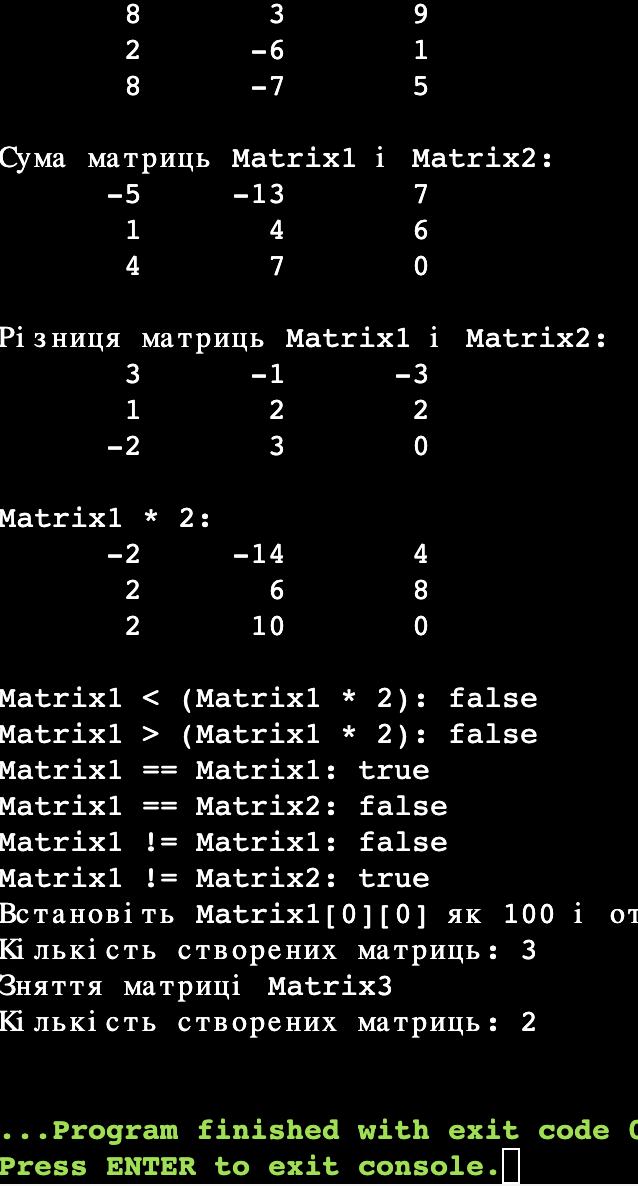
cout << "Зняття матриці Matrix3" << endl;

delete m3;

cout << "Кількість створених матриць: " << Matrix::matrix\_count << endl;

}





2.

#include <iostream>

using namespace std;

template <typename T>

class MATRIX

{

private:

T\*\* M;

int m;

int n;

public:

MATRIX()

{

n = m = 0;

M = nullptr;

}

MATRIX(int \_m, int \_n)

{

m = \_m;

n = \_n;

M = (T\*\*) new T\*[m];

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

M[i] = (T\*)new T[n];

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

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

M[i][j] = 0;

}

MATRIX(const MATRIX& \_M)

{

m = \_M.m;

n = \_M.n;

M = (T\*\*) new T\*[m];

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

M[i] = (T\*) new T[n];

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

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

M[i][j] = \_M.M[i][j];

}

T GetMij(int i, int j)

{

if ((m > 0) && (n > 0))

return M[i][j];

else

return 0;

}

void SetMij(int i, int j, T value)

{

if ((i < 0) || (i >= m))

return;

if ((j < 0) || (j >= n))

return;

M[i][j] = value;

}

void Print(const char\* ObjName)

{

cout << "Object: " << ObjName << endl;

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

{

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

cout << M[i][j] << "\t";

cout << endl;

}

cout << "---------------------" << endl << endl;

}

MATRIX operator=(const MATRIX& \_M)

{

if (n > 0)

{

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

delete[] M[i];

}

if (m > 0)

{

delete[] M;

}

m = \_M.m;

n = \_M.n;

M = (T\*\*) new T\*[m];

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

M[i] = (T\*) new T[n];

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

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

M[i][j] = \_M.M[i][j];

return \*this;

}

~MATRIX()

{

if (n > 0)

{

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

delete[] M[i];

}

if (m > 0)

delete[] M;

}

};

int main()

{

MATRIX<int> M(2, 3);

M.Print("M");

int i, j;

for (i = 0; i < 2; i++)

for (j = 0; j < 3; j++)

M.SetMij(i, j, i + j);

M.Print("M");

MATRIX<int> M2 = M;

M2.Print("M2");

MATRIX<int> M3;

M3 = M;

M3.Print("M3");

MATRIX<int> M4;

M4 = M3 = M2 = M;

M4.Print("M4");

}

