Міністерство освіти і науки України

ВСП - Технічний коледж Національного університету "Львівська політехніка"

Відділення

інформаційних технологій

та комп’ютерної техніки

**Звіт**

До лабораторної роботи № 2

з предмету «Навчальна практика з об’єктно-орієнтованого програмування»

тема: «Перевантаження операторів С++»

*Виконала:*

*Студентка групи 31 ПЗ*

*Павлисько Анастасія*

*Перевірив: Чистяк В.І.*

Львів – 2021

**Мета:** Ознайомитись з поняттям та призначенням дружніх функцій та конструктора копіювання. Навчитися реалізовувати перевантаження операторів.

**Хід роботи**

Описати клас, що реалізовує вказаний нижче тип даних. Клас повинен містити множину конструкторів для створення об'єктів певного типу (конструктор по замочуванню та з параметрами, конструктор копії) та подані у таблиці операції над об’єктами класу з використанням механізму перевантаження операцій:

**Варіант 5**

Тип даних: “матриця”

Операції: додавання, частка, обчислення транспонованої матриці

Написати програму, яка демонструє роботу з об’єктами цього класу. Програма повинна містити меню для перевірки усіх методів класу і операцій.

**Лістинг програми**

Matrix.h

#pragma once

#include <vector>

class Matrix

{

int n, m;

std::vector<std::vector<double>> matrix;

public:

Matrix();

Matrix(int a, int b);

Matrix(Matrix& mat);

void set\_size(int a, int b);

void set\_matrix();

void Print();

friend Matrix& operator+(const Matrix& mat1, const Matrix& mat2);//додавання двох матриць

Matrix& operator/(Matrix& mat);//частка двох матриць - множення на обернену матрицю дільника

Matrix& Trans();//транспонування матриці

friend Matrix& operator/(Matrix& mat,const double divider);//ділення матриці на число

Matrix& algadditions();//повертає матрицю алгебраїчних доповнень

double minor(int index1, int index2);//повертає мінор відносно елемента матриці

double det();//визначник матриці

friend Matrix& operator\*(const Matrix mat1, const Matrix mat2);//множення матриць

Matrix& operator=(const Matrix& mat);

};

Matrix.cpp

#include "Matrix.h"

#include <iostream>

#include <iomanip>

Matrix::Matrix()

{

n = 0; m = 0;

}

Matrix::Matrix(int a, int b)

{

set\_size(a, b);

set\_matrix();

}

Matrix::Matrix(Matrix& mat)

{

set\_size(mat.n, mat.m);

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

matrix.push\_back(mat.matrix[i]);

}

}

void Matrix::set\_size(int a, int b)

{

this->n = a;

this->m = b;

}

void Matrix::set\_matrix()

{

double element;

std::cout << "Enter your matrix " << n << "x" << m << " : \n";

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

std::vector<double> vec;

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

std::cin >> element;

vec.push\_back(element);

}

matrix.push\_back(vec);

}

}

void Matrix::Print()

{

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

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

std::cout << std::setw(5) <<round(matrix[i][j] \* 100) / 100 << ' ';

}

std::cout << '\n';

}

}

Matrix& operator+(const Matrix& mat1, const Matrix& mat2)

{

Matrix\* result = new Matrix();

if (mat1.n == mat2.n && mat1.m == mat2.m) {

result->set\_size(mat1.n, mat2.m);

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

std::vector<double> vec;

for (int j = 0; j <result->m; j++) {

vec.push\_back(mat1.matrix[i][j] + mat2.matrix[i][j]);

}

result->matrix.push\_back(vec);

}

}

else std::cout << "\nYou can not add this two matrix with different sizes!\n";

return \*result;

}

Matrix& Matrix::operator/(Matrix& mat)

{

Matrix\* result = new Matrix();

result->set\_size(n, m);

if (mat.det() == 0.0) {

std::cout << "You can not divide this matrix - determin is zero\n";

}

else {//якщо det!=0 знаходимо обернену матрицю до матриці-дільника

Matrix divider(mat.algadditions());//складаємо матрицю алгебраїчних доповнень (союзну) та протранспонувати її

divider.Trans();

divider = divider / mat.det();//ділимо союзну матрицю на детермінант

\*result = (\*this) \* divider; //виконуємо множення матриць

}

return \*result;

}

Matrix& Matrix::Trans()

{

double temp;

int k = (m <= n) ? m : n;

//для квадратної матриці чи частини прямокутної

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

for (int j = i + 1; j < k; j++) {

temp = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = temp;

}

}

if (n > m) {//коли рядків більше ніж стовпців

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

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

matrix[j].push\_back(matrix[i][j]);

}

}

matrix.erase(matrix.begin()+m, matrix.end());//видаляє рядки транспоновані в стовпці

int tmp = n; n = m; m = tmp;//змінює розміри матриці

}

else if (n < m) {//коли стовпців більше ніж рядків

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

std::vector<double> vec;

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

vec.push\_back(matrix[j][i]);

}

matrix.push\_back(vec);

}

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

matrix[i].erase(matrix[i].begin() + n, matrix[i].end());//видаляє стовпці транспоновані в рядки

}

int tmp = n; n = m; m = tmp;//змінює розміри матриці

}

return \*this;

}

Matrix& Matrix::algadditions()

{

Matrix \*A = new Matrix(); A->set\_size(n, m);

if (n != m) {

std::cout << "\nYou can not find minor for this matrix (not square matrix)!\n";

}

else {

int sign = -1;

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

std::vector<double> vec;

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

sign \*= -1;

vec.push\_back(sign\*minor(i, j));

}

A->matrix.push\_back(vec);

}

}

return \*A;

}

double Matrix::minor(int index1, int index2)

{

double min = 0;

if (n != m) {

std::cout << "\nYou can not find minor for this matrix (not square matrix)!\n";

}

else {

Matrix b; b.set\_size(n - 1, m - 1);

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

std::vector<double> vec;

for (int k = 0; k < m; k++) {

if (j != index1 && k != index2) vec.push\_back(matrix[j][k]);

}

if(!vec.empty())b.matrix.push\_back(vec);

}

if (b.n == 1 && b.m == 1) min = b.matrix[0][0];

else min = b.det();

}

return min;

}

double Matrix::det()

{

double determin = 0;

if (n == m) {

if (n == 2)

determin = matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];

else {

int sign = -1;

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

sign \*= -1;

Matrix b; b.set\_size(n - 1, m - 1);

for (int j = 1; j < n; j++) {

std::vector<double> vec;

for (int k = 0; k < m; k++) {

if (k != i) vec.push\_back(matrix[j][k]);

}

b.matrix.push\_back(vec);

}

determin += sign \* matrix[0][i] \* b.det();

}

}

}

else

std::cout << "\nYou can not find determin for this matrix (not square matrix)!\n";

return determin;

}

Matrix& operator/(Matrix& mat, const double divider)

{

Matrix\* result = new Matrix();

if (divider == 0.0) {

std::cout << "Divider can not be equal 0\n";

}

else {

result->n = mat.n;

result->m = mat.m;

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

std::vector<double> vec;

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

vec.push\_back(mat.matrix[i][j] / divider);

}

result->matrix.push\_back(vec);

}

}

return \*result;

}

Matrix& Matrix::operator=(const Matrix& mat)

{

(\*this).set\_size(mat.n, mat.m);

if (!matrix.empty())matrix.clear();

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

(\*this).matrix.push\_back(mat.matrix[i]);

}

return (\*this);

}

Matrix& operator\*(const Matrix mat1, const Matrix mat2)

{

Matrix\* result = new Matrix();

if (mat1.m == mat2.n) {

result->n = mat1.n;

result->m = mat2.m;

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

std::vector<double> vec;

for (int j = 0; j < result->m; j++) {

double elem = 0;

for (int k = 0; k < mat1.m; k++) {

elem += mat1.matrix[i][k] \* mat2.matrix[k][j];

}

vec.push\_back(elem);

}

result->matrix.push\_back(vec);

}

}

else

std::cout << "\nYou can not multiply this two matrixes!\n";

return \*result;

}

Source.cpp

#include "Matrix.h"

#include <iostream>

void menu() {

std::cout << "\nEnter character for to do something:\n"

<< "+ to add matrices\n"

<< "/ to divide matrices\n"

<< "% to divide matrix by number\n"

<< "t to transpose matrix\n"

<< "\* to multiply matrices\n"

<< "d to find determinant of matrix\n"

<< "m to find minor for element of matrix\n"

<< "a return matrix of algebras additions\n"

<< "e to exit\n";

}

int main() {

int n1, m1, n2, m2;

std::cout << "Enter size of first matrix (nxm):\n";

std::cin >> n1 >> m1;

Matrix mat1(n1, m1);

std::cout << "Enter size of second matrix (nxm):\n";

std::cin >> n2 >> m2;

Matrix mat2(n2, m2);

char flag = 's';

while (flag != 'e') {

std::cin.ignore();

system("cls");

std::cout << "First matrix: \n";

mat1.Print();

std::cout << "Second matrix: \n";

mat2.Print();

menu();

std::cin >> flag;

switch (flag)

{

case '+': {

std::cout << "Sum of this matrices is:\n";

Matrix\* result=new Matrix(mat1 + mat2);

result->Print();

delete result;

}break;

case '/': {

std::cout << "Result of dividing these matrices is:\n";

Matrix\* result = new Matrix(mat1 / mat2);

result->Print();

delete result;

}break;

case '%': {

int num;

std::cout << "Choose matrix 1 or 2: "; std::cin >> num;

if (num != 1 && num != 2) {

std::cout << "Incorect input\n";

}

else {

double divider;

std::cout << "Enter divider: "; std::cin >> divider;

std::cout << "Result of dividing matrix " << num << " by " << divider << " is:\n";

Matrix result;

if (num == 1) result = mat1 / divider;

else result = mat2 / divider;

result.Print();

}

}break;

case 't': {

int num;

std::cout << "Choose matrix 1 or 2: "; std::cin >> num;

if (num != 1 && num != 2) {

std::cout << "Incorect input\n";

}

else {

Matrix result;

std::cout << "Result of transposing matrix " << num << " is:\n";

if (num == 1) result = mat1;

else result = mat2;

result.Trans().Print();

}

}break;

case '\*': {

std::cout << "Product of this matrices is:\n";

Matrix result = mat1 \* mat2;

result.Print();

}break;

case 'd': {

int num;

std::cout << "Choose matrix 1 or 2: "; std::cin >> num;

if (num != 1 && num != 2) {

std::cout << "Incorect input\n";

}

else {

std::cout << "Determinant of matrix " << num << " is: ";

if (num == 1) std::cout << mat1.det() << '\n';

else std::cout << mat2.det() << '\n';

}

}break;

case 'm': {

int num;

std::cout << "Choose matrix 1 or 2: "; std::cin >> num;

if (num != 1 && num != 2) {

std::cout << "Incorect input\n";

}

else {

int i, j;

std::cout << "Enter index (i, j) of element: ";

std::cin >> i >> j;

if(num == 1){

if ((i < 0 || i > n1) || (j < 0 || j > m1)) {

std::cout << "Incorect input\n";

}

else {

std::cout << "Minor element["<< i <<"]["<< j <<"] of matrix " << num << " is: " << mat1.minor(i, j) << '\n';

}

}

else {

if ((i < 0 || i > n2) || (j < 0 || j > m2)) {

std::cout << "Incorect input\n";

}

else {

std::cout << "Minor element[" << i << "][" << j << "] of matrix " << num << " is: " << mat2.minor(i, j) << '\n';

}

}

}

}break;

case 'a': {

int num;

std::cout << "Choose matrix 1 or 2: "; std::cin >> num;

if (num != 1 && num != 2) {

std::cout << "Incorect input\n";

}

else {

std::cout << "Matrix of algebras additions for matrix " << num << " is:\n";

if (num == 1) mat1.algadditions().Print();

else mat2.algadditions().Print();

}

}break;

default:

break;

}

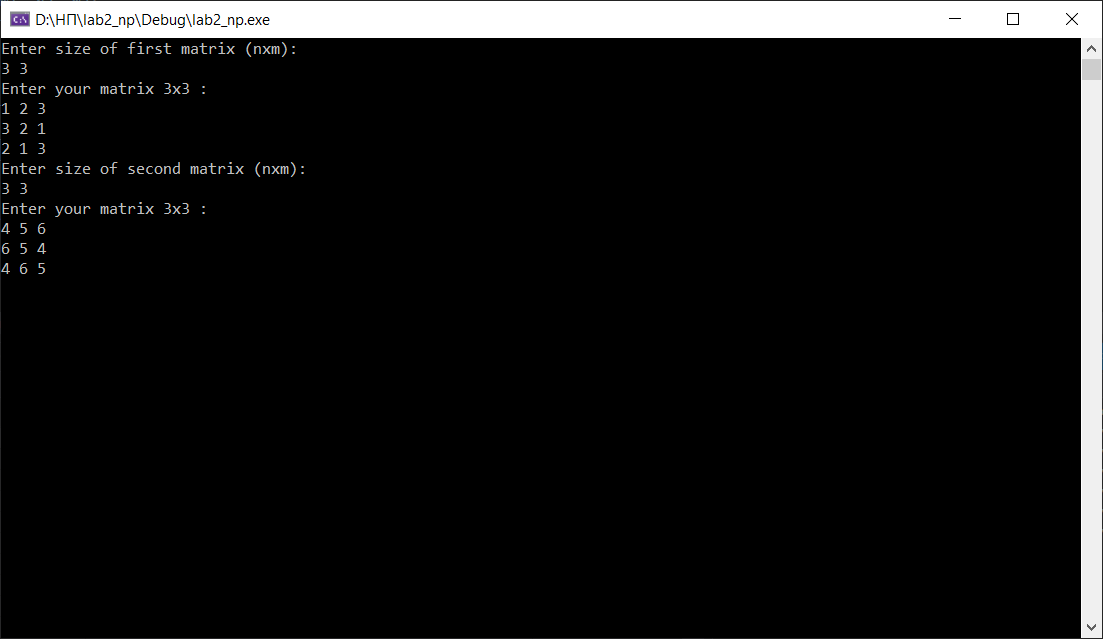
system("pause");

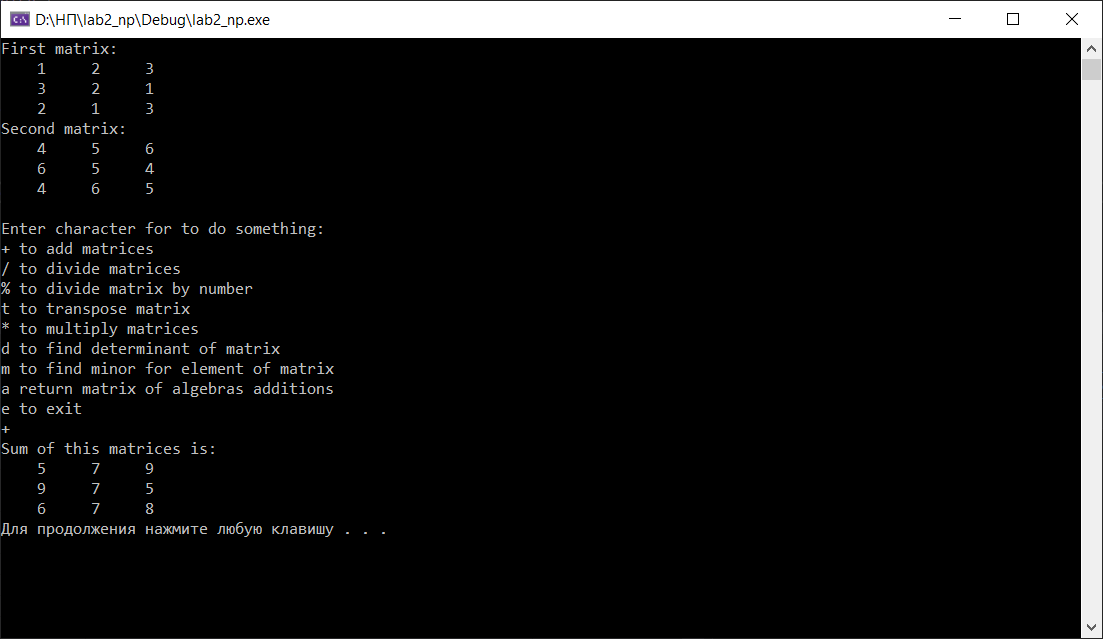
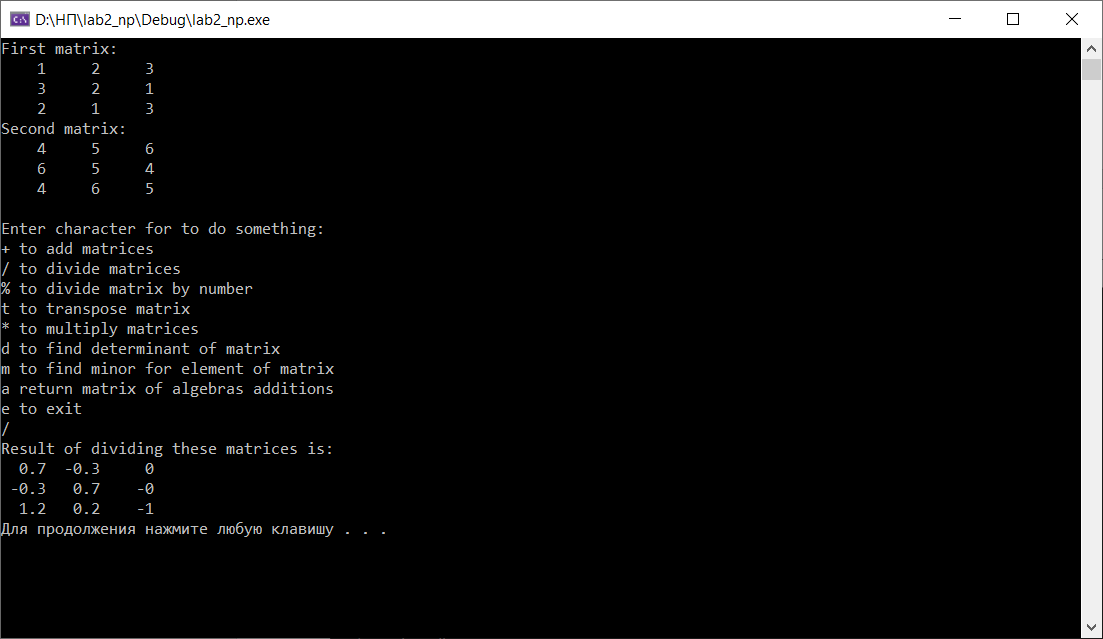
}

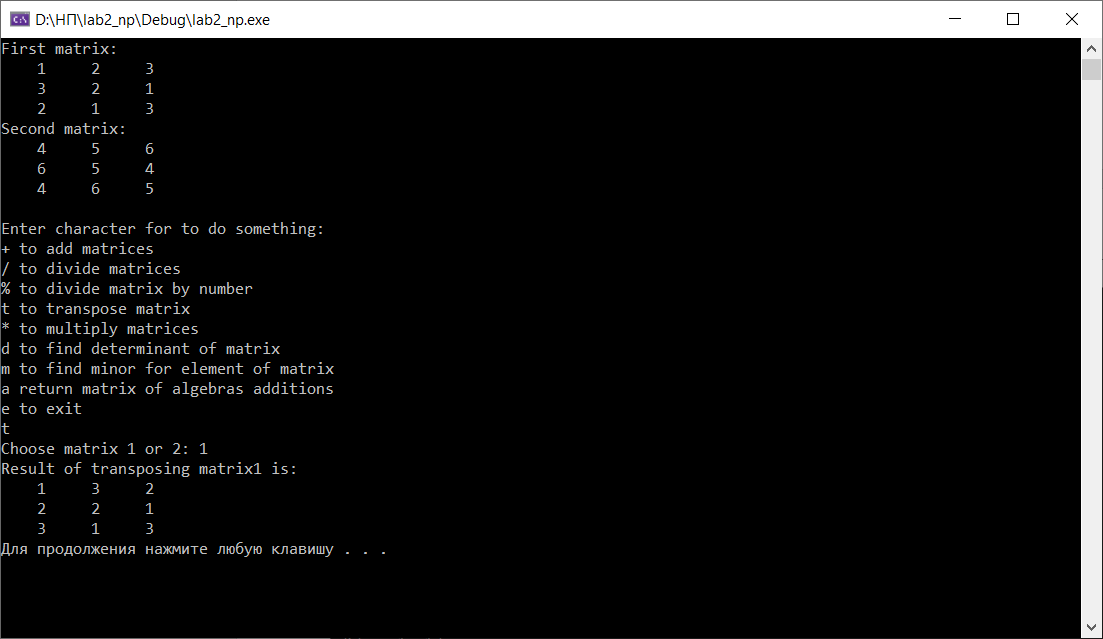
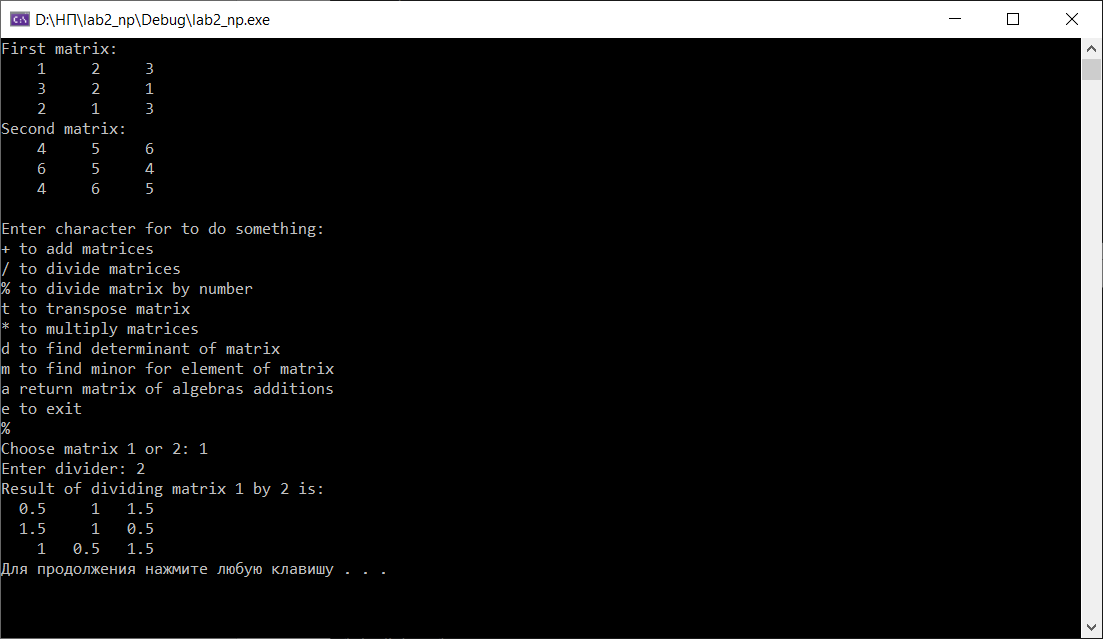
return 0;

}

**Результати роботи програми**





**Висновок:**

Виконуючи цю лабораторну роботу, я ознайомилась з поняттям та призначенням дружніх функцій та конструктора копіювання; навчилася реалізовувати перевантаження операторів.