**Анотація**

Описано програму, яка виконує основну роботу з матрицями, використаний об’єктно-орієнтований підхід розробки програмних продуктів мовою програмування С++. Застосовано мову моделювання (UML) для відображення акторами та прецедентами в потребі до системі. В результаті роботи програма надає користувачеві можливість легко і швидко виконувати математичні дії над матрицями.

**Вступ**

[Курсова](http://ua-referat.com/%D0%9A%D1%83%D1%80%D1%81%D0%BE%D0%B2%D0%B0) [робота](http://ua-referat.com/%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0) виконана на мові [програмування](http://ua-referat.com/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) високого рівня С + + з використанням компілятора Microsoft Visual Studio 2010. Ця [мова](http://ua-referat.com/%D0%9C%D0%BE%D0%B2%D0%B0) була розроблена на початку 80-х років в Bell LaboratoriesMicrosoft Visual C++ містить безліч інтегрованих засобів візуального програмування. Компілятор Visual C++ містить багато нових інструментальних засобів і поліпшених можливостей, надає величезні можливості в плані оптимізації додатків, внаслідок чого можна отримати виграш як відносно розміру програми, так і відносно швидкості її виконання, незалежно від того, що являє собою ваш додаток.

Система Microsoft Visual C++ дозволяє створювати як маленькі програми і утиліти для персонального використання, так і корпоративні системи, що працюють з базами даних на різних платформах.

Дана програма створена для роботи з матрицями. Завдання звучить, як : «Робота над матрицями (+,-,\*,det)». В процесі роботи над програмою окрім основного завдання також розроблено алгоритм який виконує такі як, транспонування матриці і множення її на скаляр.

Для написання даної курсової роботи використовувалося середовище MicrosoftVisualStudio 2010 , яке на даний час є найбільш високотехнічним і практичним для написання і компілювання коду. Написана програма працює на консолі операційних систем як Windows та Linux .

Мета курсової роботи полягає за допомогою об’єктно - орієнтованого підходу спростити роботу над матриця до автоматизації. Програма може використовуватися для визначення детермінанта матриці , множення двох матриць, додавання та віднімання двох матриць, множення на скаляр та знаходження транспонованої матриці.

Для написання курсової робити не використовувалися аналоги інших матричних калькуляторів. В даній курсовій роботі порівняння та недоліки з іншими аналогічними програмами не вказано.

**1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ**

Розробляємо програму для роботи з матрицями(+, -, \*, det).

На базі ТЗ було побудовано діаграму прецедентів, яка відображає вимоги замовника до системих. (на рис. 1.1).



Рисунок 1.1 Діаграма варіантів використання **2 ОБГРУНТУВАННЯ АЛГОРИТМУ І СТРУКТУРИ ПРОГРАМИ**

Множення на скаляр

Транспонування

Детермінант

Добуток

Віднімання

Додавання

Множення на скаляр

Транспонування

Детермінант

Добуток

Віднімання

Додавання

Математичні операції над матрицями

Теоретичні відомості

Головне меню

Рис. 2.1 Алгоритм програми

На Рисунку 2.1 наведений алгоритм роботи програми. При запуску програми появляється діалогове вікно (Головне меню) де вказані основні дії і можливості програми. Щоб вибрати певну дію треба нажати цифру , яка відповідає цій дії. Перший вибір полягає чи користувач хоче отримати теоретичні знання чи зробити певну математичну операцію з матрицею.

Розглянемо коли при введені цифри 1. Появиться нове діалогове вікно з 7 можливими варіантами. Шість з яких виведуть на екран теорію про певну операцію над матрицею. Кожна цифра виводить одну операцію. При нажаті q відбувається вихід з програми , при нажимані іншої клавіші появляється знову вибір операції.

Розглянемо коли в Головному меню ввести цифру 2. Появиться нове діалогове вікно з 7 можливими варіантами. При введені цифри 1- додавання матриць , користувач повинен ввести кількість рядків і стовпців матриць , як при додаванні вони мають бути рівні тому вводиться тільки один раз. Після чого вводяться дві матриці , після їх введення на екран виводиться вже сума. При введені цифри 2- віднімання матриць все йде по аналогії до першої дії , тільки замість додавання відбувається віднімання. При введені цифри 3 – множення матриць , користувач повинен вказати кількість рядків першої матриці , кількість стовпців першої матриці і кількість стовпців другої матриці , через те що множення матриць можливе коли кількість стовпців першої матриці дорівнює кількості рядків другої матриці , через те не йде запит за кількість рядків другої матриці. При введені цифри 4 – детермінант матриці, користувач повинен вказати кількість рядків матриці, матриця повинна бути квадратною другого або третього порядку, при введені кількості матриці не другого або третього порядку то видасть помилку. При введені цифри 5 – транспонування матриці, користувачу треба ввести кількість рядків і стовпців матриці, яку хочуть транспонувати. При введені цифри 6 – множення матриці на скаляр, користувачу треба ввести матрицю і скаляр, число на яке користувач хоче помножити матрицю. При введені q – користувач вийде з програми.

Після обчислення кожної дії появляється друге діалогове меню і обрахування можна повторити ще раз. При введені будь-якого іншого знака програма видасть друге діалогове меню. Вибір операції здійснюється за допомогою оператора switch. Інтерфейс є простим і логічним у користуванні, що призводить до легкого керування програмою.

**3 РОЗРОБКА ПРОГРАМИ**

Програма була розроблена на основі об’єктно - орієнтованого підходу. Програма складається з 4 файлів. Перший файл розширення \*.h , другий \*.cpp , третій \*.h і четвертий \*.txt . У файлі **matrix.h** знаходяться класи і методи які потрібні для роботи програми. У другому файлі **main.cpp** прописаний самий інтерфейс програми в діалогову вікні , за допомогою цього файлу вводиться матриці , розміри матриць і більшість чисел які потрібні для роботи програми. Третій файл називається **theory.h** в цьому файлі знаходиться клас відповідаючий за копірування теоретичних відомосте з файлу і ввід їх в консоль. Четвертий файл називається teoria.txt у файлі міститься теоретичні відомості про операції які є в даній програмі.

**3.1 Розробка системи класів**

В [об'єктно-орієнтованому програмуванні](http://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), класи використовуються для групування пов'язаних змінних та функцій. Клас описує набір [інкапсульованих](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F) [змінних екземпляра](http://uk.wikipedia.org/w/index.php?title=%D0%97%D0%BC%D1%96%D0%BD%D0%BD%D1%96_%D0%B5%D0%BA%D0%B7%D0%B5%D0%BC%D0%BF%D0%BB%D1%8F%D1%80%D0%B0&action=edit&redlink=1) та [методів](http://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29) (функцій), можливо, разом з

реалізацією цих типів разом з конструктором, який може використовуватись для створення екземплярів класа. Клас є [зв'язаним](http://uk.wikipedia.org/w/index.php?title=%D0%97%D0%B2%27%D1%8F%D0%B7%D0%B0%D0%BD%D0%BD%D1%96%D1%81%D1%82%D1%8C_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29&action=edit&redlink=1) пакетом, який складається із спеціальних [метаданих](http://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%B4%D0%B0%D0%BD%D1%96) часу [компіляції](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80). Він описує правила, за якими діють [об'єкти](http://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29). Ці об'єкти називаються "екземплярами" цього класу. Клас визначає структуру даних, що містить кожен екземпляр, і, також, [методи](http://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29) ([функції](http://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29)), які обробляють дані екземпляра та виконують завдання. Ці методи часто називають "[поведінкою](http://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D0%B5%D0%B4%D1%96%D0%BD%D0%BA%D0%B0)". Метод є функцією, що має доступ до даних об'єкта. Клас є найспецифічнішим [типом даних](http://uk.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) об'єкта по відношенню до окремого прошарку. Клас може мати представлення ([метаоб'єкт](http://uk.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D1%82%D0%B0%D0%BE%D0%B1%27%D1%94%D0%BA%D1%82&action=edit&redlink=1)) під час виконання програми, який надає допомогу в роботі з метаданими класу.

В даній програмі є три класи Theory , Matrix , NewMatrix. Перший клас відповідає за вивід інформації з файлу у консольне вікно , в ньому находяться 6 методів , які виводять теоретичну інформацію стосовно кожної операції.

Другий клас відповідає за виконання операцій над матрицями. В якому находяться шість методів.

Третій клас відповідає за створення динамічного двох вимірного масива.

**3.3 Опис файлу даних**

Всі теоретичні відомості що є в даній програмі записані у файл teoria.txt .

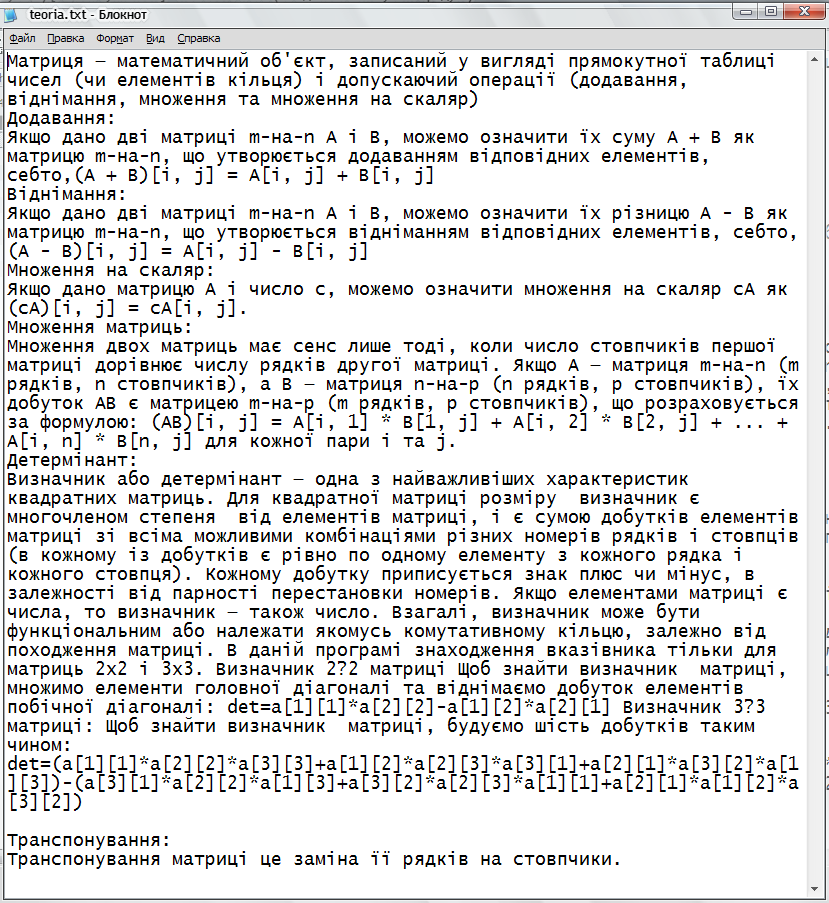


Рисунок 3.1 Теоретичні відомості

**4 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛЬТАТИ ЇЇ ВИКОНАННЯ**

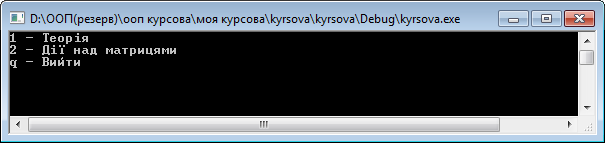


Рисунок 4.1 Головне меню програми

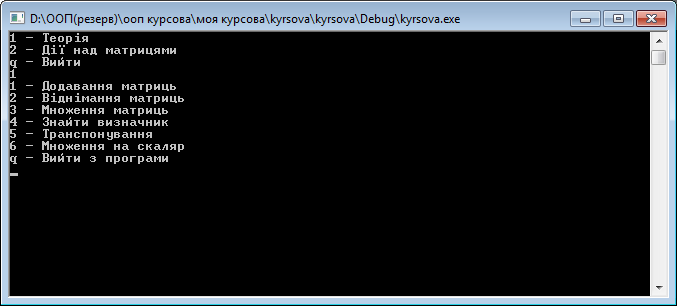


Рисунок 4.2 Підменю “Теорія”

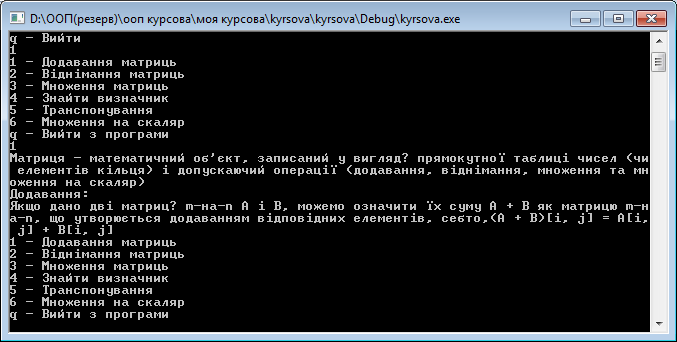


Рисунок 4.3 Теоретичні відомості про додавання матриць

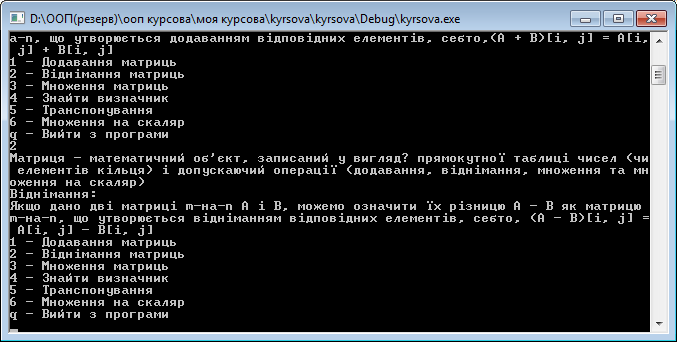


Рисунок 4.4 Теоретичні відомості про віднімання матриць

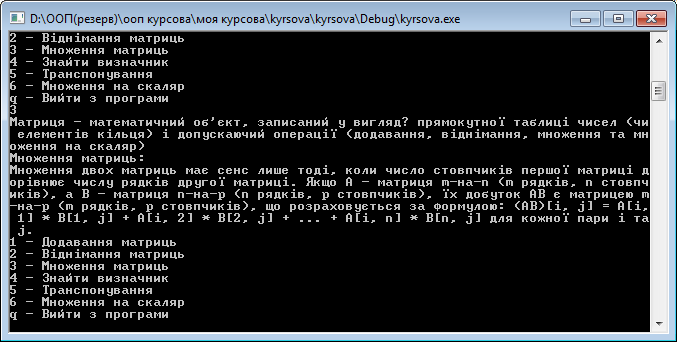


Рисунок 4.5 Теоретичні відомості про множення матриць

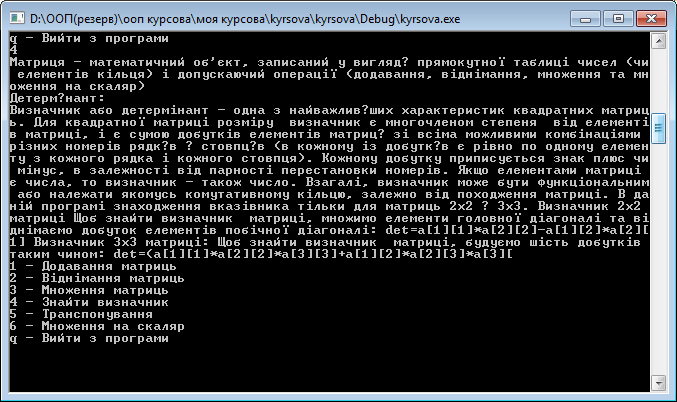


Рисунок 4.6 Теоретичні відомості про визначник матриці

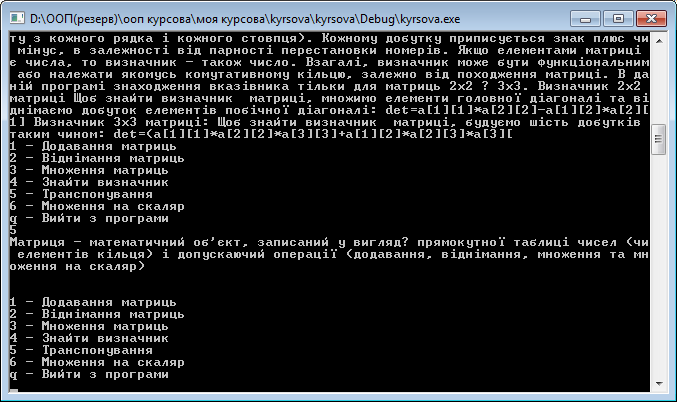


Рисунок 4.7 Теоретичні відомості про транспонування матриці

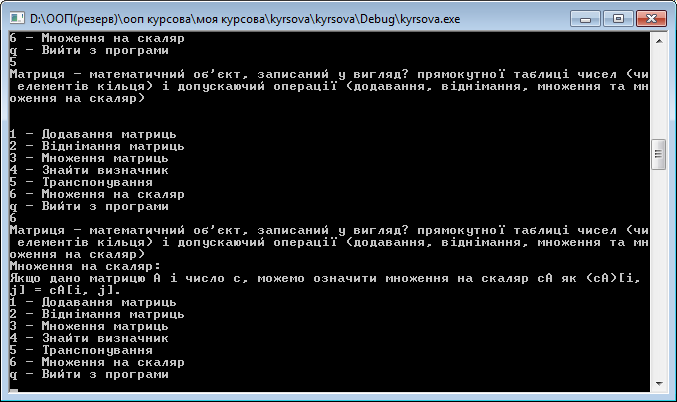


Рисунок 4.8 Теоретичні відомості про множення матриці на скаляр

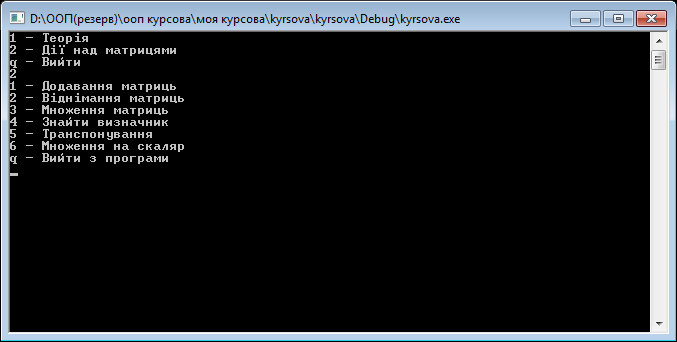


Рисунок 4.9 Дії над матрицями

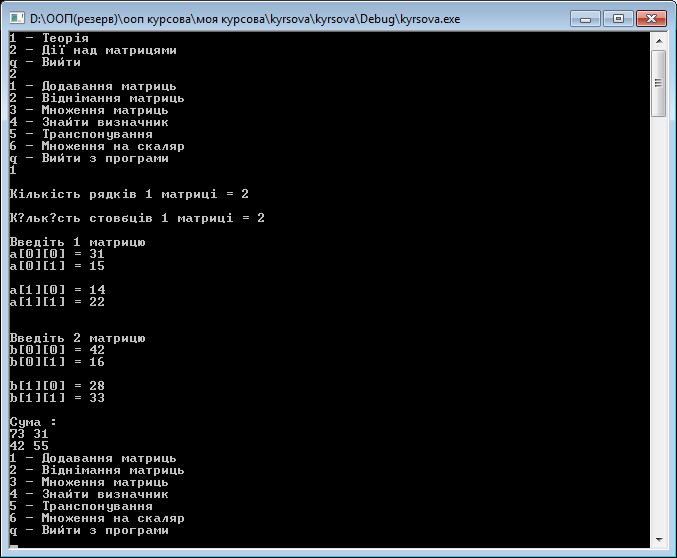
.

Рисунок 4.10 Додавання

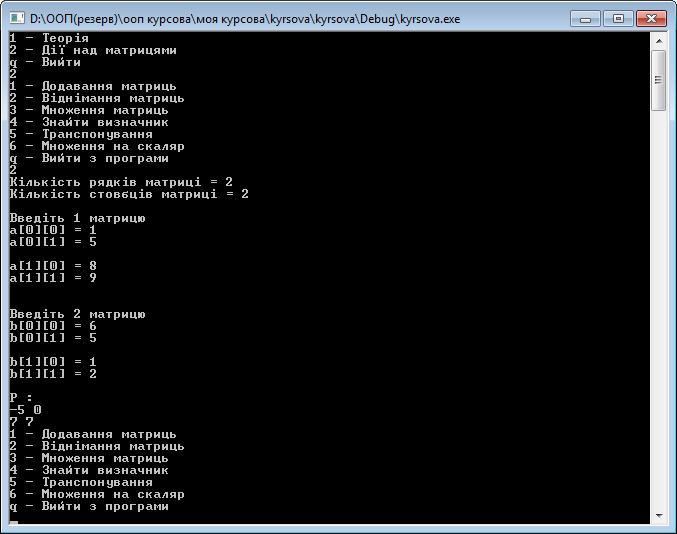


Рисунок 4.11 Віднімання

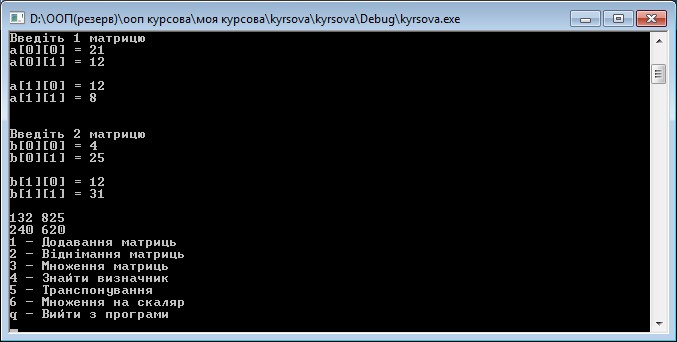


Рисунок 4.12 Множення

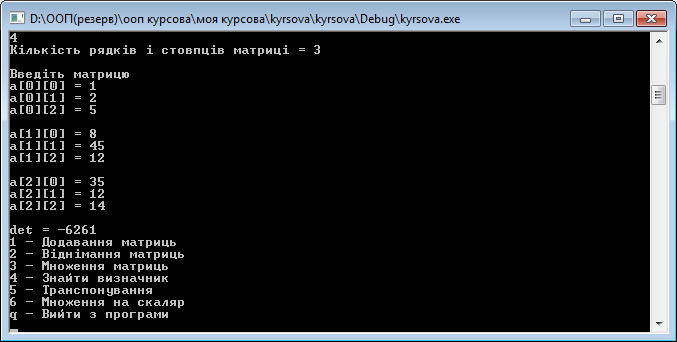


Рисунок 4.13 Визначник

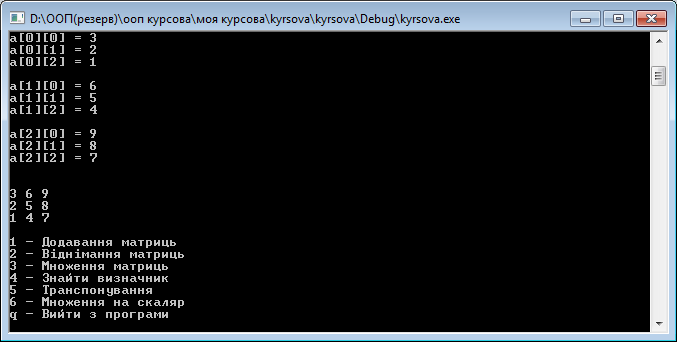


Рисунок 4.14 Транспонування

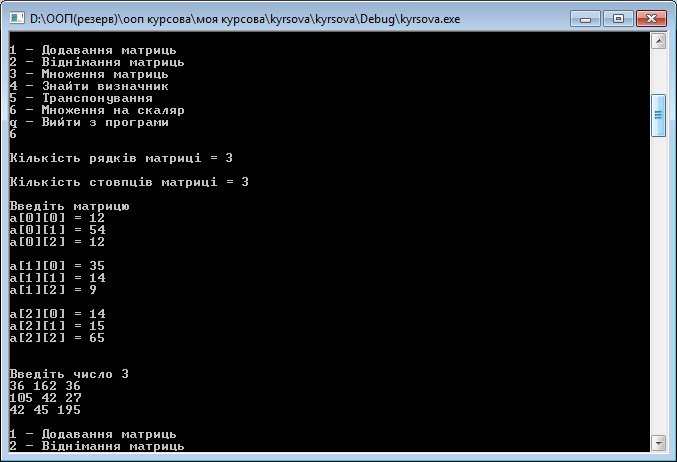


Рисунок 4.15 Множення на скаляр

**ВИСНОВКИ**

В результаті виконання курсової робити було створено програму, яка забезпечує різнопланову роботу з матрицями, вона створена з використанням компілятора Microsoft Visual Studio 2010, на мові програмування С++.

Дана програма надає користувачеві можливість легко і швидко виконувати математичні дії над матрицями, а саме: додавання, віднімання, множення матриць, знаходження детермінанта матриці, транспонування матриці та множення матриці на скаляр.

Створена програма «Робота над матрицями (+,-,\*,det)» в процесі доопрацювання може бути використана за основу для створення більш складної та великої програми по роботі в даній предметній області.

**Список використаної літератури**

1. Агуров Павел - Последовательные интерфейсы ПК. Практика программирования. — Санкт-Петербург: "БХВ - Петербург", 2004.
2. Герберт Шилдт - C# 4.0 полное руководство. — Чернівці: Рута, 1996.
3. Коваль А. П. Ділове спілкування. — Санкт-Петербург: "Вильямс", 2011.
4. Н. Культин - Microsoft Visual C# в задачах и примерах. — Санкт-Петербург: "БХВ - Петербург", 2009

**Додаток A**

**Matrix.h**

#include<iostream>

#include<math.h>

using namespace std;

class Matrix

{

public: void Plus(double\*\*a, int n, int m, double\*\*b, int k, int l)

{

double \*\*c = new double\*[n];

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

{

c[i] = new double [m];

}

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

{

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

{

c[i][j] = a[i][j] + b[i][j];

}

}

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

{

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

{

cout << c[i][j] << " ";

}

cout << endl;

}

}

void Minus(double\*\*a, int n, int m, double\*\*b, int k, int l)

{

double \*\*c = new double\*[n];

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

{

c[i] = new double [m];

}

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

{

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

{

c[i][j] = a[i][j] - b[i][j];

}

}

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

{

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

{

cout << c[i][j] << " ";

}

cout << endl;

}

}

void Dobutok(double\*\*a, int n, int m, double\*\*b, int k, int l)

{

double \*\*c = new double\*[n];

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

{

c[i] = new double [l];

}

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

{

for(int d = 0; d < l; d++)

{

c[j][d] = 0;

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

{

c[j][d] += a[i][j] \* b[j][d];

}

}

}

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

{

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

{

cout << c[i][j] << " ";

}

cout << endl;

}

}

void Det(double\*\*a, int n, int m)

{

double det;

if(n==2)

{

det=(a[0][0]\*a[1][1])-(a[0][1]\*a[1][0]);

cout<<"det = "<<det;

}

if(n==3)

{

det=(a[0][0]\*a[1][1]\*a[2][2]+a[0][1]\*a[1][2]\*a[2][0]+a[1][0]\*a[2][1]\*a[0][2])-(a[2][0]\*a[1][1]\*a[0][2]+a[2][1]\*a[1][2]\*a[0][0]+a[1][0]\*a[0][1]\*a[2][1]);

cout<<"det = "<<det;

}

if(n<2||n>3)

{

cout<<"error";

}

cout<<endl;

}

void Tran(double\*\*a, double\*\*b, int n, int m)

{

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

{

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

b[j][i] = a[i][j];

}

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

{

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

{

cout << b[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

void Skalar(double\*\*a, int n, int m, double sca)

{

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

{

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

a[i][j] \*= sca;

}

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

{

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

{

cout << a[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

};

class NewMatrix

{

public: double\*\* Create(int n, int m)

{

double \*\*a = new double\*[n];

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

{

a[i] = new double [m];

}

return a;

}

};

**Theory.h**

#include <iostream>

#include <math.h>

#include <fstream>

#include <string>

using namespace std;

class Theory

{

public: void Plus()

{

setlocale(LC\_ALL, ".1251");

ifstream ifile("teoria.txt", ios::in | ios::binary);

char str[1024];

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

cout << str << endl;

}

void Minus()

{

setlocale(LC\_ALL, ".1251");

ifstream ifile("teoria.txt", ios::in | ios::binary);

char str[1024];

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

cout << str << endl;

}

void Dobutok()

{

setlocale(LC\_ALL, ".1251");

ifstream ifile("teoria.txt", ios::in | ios::binary);

char str[1024];

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

cout << str << endl;

}

void Det()

{

setlocale(LC\_ALL, ".1251");

ifstream ifile("teoria.txt", ios::in | ios::binary);

char str[1024];

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

cout << str << endl;

}

void Tran()

{

setlocale(LC\_ALL, ".1251");

ifstream ifile("teoria.txt", ios::in | ios::binary);

char str[1024];

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

cout << str << endl;

}

void Skalar()

{

setlocale(LC\_ALL, ".1251");

ifstream ifile("teoria.txt", ios::in | ios::binary);

char str[1024];

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

ifile.getline(str, 1024);

cout << str << endl;

ifile.getline(str, 1024);

cout << str << endl;

}

};

**Main.cpp**

#include<iostream>

#include<math.h>

#include"matrix.h"

#include"Theory.h"

#include <locale.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Ukrainian");

int n, m, k, l;

Matrix obj;

NewMatrix obj2;

Theory obj3;

double \*\*a, \*\*b;

char tmp = ' ';

char tmp2 = ' ';

cout << "1 - Теорiя" << endl << "2 - Дiї над матрицями" << endl << "q - Вийти" << endl;

cin >> tmp2;

if(tmp2 == '2')

{

while (tmp != 'q')

{

cout << "1 - Додавання матриць" << endl;

cout << "2 - Вiднiмання матриць" << endl;

cout << "3 - Множення матриць" << endl;

cout << "4 - Знайти визначник" << endl;

cout << "5 - Транспонування" << endl;

cout << "6 - Множення на скаляр" << endl;

cout << "q - Вийти з програми" << endl;

cin >> tmp;

if (tmp == 'q')

break;

switch (tmp)

{

case '1':

{

cout << endl;

cout << "Кiлькiсть рядкiв 1 матрицi = ";

cin >> n;

cout << endl << "Кількість стовбцiв 1 матрицi = ";

cin >> m;

k = n;

l = m;

a = obj2.Create(n, m);

b = obj2.Create(k, l);

cout << endl << "Введiть 1 матрицю" << endl;

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

{

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

{

cout << "a[" << i << "][" << j << "] = ";

cin >> a[i][j];

}

cout << endl;

}

cout << endl << "Введiть 2 матрицю" << endl;

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

{

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

{

cout << "b[" << i << "][" << j << "] = ";

cin >> b[i][j];

}

cout << endl;

}

cout << "Сума :" << endl;

obj.Plus(a, n, m, b, k, l);

break;

}

case '2':

{

cout << "Кiлькiсть рядкiв матрицi = ";

cin >> n;

cout << "Кiлькiсть стовбцiв матрицi = ";

cin >> m;

k = n;

l = m;

a = obj2.Create(n, m);

b = obj2.Create(k, l);

cout << endl << "Введiть 1 матрицю " << endl;

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

{

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

{

cout << "a[" << i << "][" << j << "] = ";

cin >> a[i][j];

}

cout << endl;

}

cout << endl << "Введiть 2 матрицю" << endl;

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

{

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

{

cout << "b[" << i << "][" << j << "] = ";

cin >> b[i][j];

}

cout << endl;

}

cout << "Рiзниця :" << endl;

obj.Minus(a, n, m, b, k, l);

break;

}

case '3':

{

cout << "Кiлькiсть рядкiв 1 матрицi = ";

cin >> n;

cout << "Кiлькiсть стовпцiв 1 матрицi = ";

cin >> m;

k = m;

cout << "Кiлькiсть стовпцiв 2 матрицi = ";

cin >> l;

a = obj2.Create(n, m);

b = obj2.Create(k, l);

cout << endl << "Введiть 1 матрицю" << endl;

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

{

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

{

cout << "a[" << i << "][" << j << "] = ";

cin >> a[i][j];

}

cout << endl;

}

cout << endl << "Введiть 2 матрицю" << endl;

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

{

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

{

cout << "b[" << i << "][" << j << "] = ";

cin >> b[i][j];

}

cout << endl;

}

obj.Dobutok(a, n, m, b, k, l);

break;

}

case '4':

{

cout << "Кiлькiсть рядкiв i стовпцiв матрицi = ";

cin >> n;

m = n;

a = obj2.Create(n, m);

cout << endl << "Введiть матрицю" << endl;

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

{

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

{

cout << "a[" << i << "][" << j << "] = ";

cin >> a[i][j];

}

cout << endl;

}

obj.Det(a, n, m);

break;

}

case '5':

{

cout << endl;

cout << "Кiлькiсть рядкiв матрицi = ";

cin >> n;

cout << endl << "Кiлькiсть стовпцiв матрицi = ";

cin >> m;

a = obj2.Create(n, m);

b = obj2.Create(m, n);

cout << endl << "Введiть матрицю" << endl;

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

{

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

{

cout << "a[" << i << "][" << j << "] = ";

cin >> a[i][j];

}

cout << endl;

}

cout << endl;

obj.Tran(a, b, n, m);

break;

}

case '6':

{

double l;

cout << endl;

cout << "Кiлькiсть рядкiв матрицi = ";

cin >> n;

cout << endl << "Кiлькiсть стовпцiв матрицi = ";

cin >> m;

a = obj2.Create(n, m);

cout << endl << "Введiть матрицю" << endl;

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

{

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

{

cout << "a[" << i << "][" << j << "] = ";

cin >> a[i][j];

}

cout << endl;

}

cout << endl;

cout << "Введiть число ";

cin >> l;

obj.Skalar(a, n, m, l);

break;

}

default:

break;

}

}

}

if (tmp2 == '1')

{

while (tmp != 'q')

{

cout << "1 - Додавання матриць" << endl;

cout << "2 - Вiднiмання матриць" << endl;

cout << "3 - Множення матриць" << endl;

cout << "4 - Знайти визначник" << endl;

cout << "5 - Транспонування" << endl;

cout << "6 - Множення на скаляр" << endl;

cout << "q - Вийти з програми" << endl;

cin >> tmp;

if (tmp == 'q')

break;

switch (tmp)

{

case '1':

{

obj3.Plus();

break;

}

case '2':

{

obj3.Minus();

break;

}

case '3':

{

obj3.Dobutok();

break;

}

case '4':

{

obj3.Det();

break;

}

case '5':

{

obj3.Tran();

break;

}

case '6':

{

obj3.Skalar();

break;

}

default:

break;

}

}

}

else

return 0;

}

**ДОДАТОК Б**

**Діаграма класів**

