НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Кафедра прикладної математики

РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА

з кредитного модуля

"Програмування 1. Основи програмування"

на тему:

«Обернення матриці прямими методами (метод Крамера)»

Виконала Гриб Вікторія Олексіївна

група КМ-02 факультет прикладної математики

N залікової книжки КМ-0205

Керівник Олефір О.С. ( )

"16" грудня 2020р.

Захищена з оцінкою\_\_\_\_\_\_\_\_\_\_\_

Київ 2020

ЗМІСТ

[1. ПОСТАНОВКА ЗАДАЧІ 3](#_Toc58777457)

[2. ПРОЕКТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ 5](#_Toc58777458)

[2.1 Методи вирішення задач 5](#_Toc58777459)

[2.2 Проектування алгоритмів 7](#_Toc58777460)

[2.3 Структура програмного забезпечення 9](#_Toc58777461)

[2.4 Опис розроблених алгоритмів 11](#_Toc58777462)

[2.5 Засоби керування програмами 12](#_Toc58777463)

[3. РЕАЛІЗАЦІЯ ПРОГРАМИ 15](#_Toc58777464)

[3.1 Опис вхідних даних 15](#_Toc58777465)

[3.2 Опис результатів 16](#_Toc58777466)

[3.3 Опис контрольних прикладів 16](#_Toc58777467)

[3.4 Експериментальні розрахунки 18](#_Toc58777468)

[ВИСНОВКИ 20](#_Toc58777469)

[ЛІТЕРАТУРА 21](#_Toc58777470)

[ДОДАТОК А 22](#_Toc58777471)

[ДОДАТОК Б 24](#_Toc58777472)

[ДОДАТОК В 32](#_Toc58777473)

1. ПОСТАНОВКА ЗАДАЧІ

Отже обернення матриці методом Крамера. Для виконання цього завдання слід розібрати його на кроки.

В загальному, мету роботи можна розділити на декілька підпунктів.

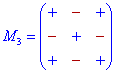
* Вивчення математичної складової задачі;
* Отримання базових знань мови програмування Python, необхідних для реалізації завдання;
* Створення схеми дії програми та її взаємодії з користувачем;
* Написання допоміжних функцій, що будуть потрібними для виконання програмою поставленої задачі;
* Передбачення коректного вводу та виводу даних з метою отримання чітких та правильних результатів та недопуску помилок в ході виконання програми;
* Створення ескізів інтерфейсу програми (за можливості – його реалізація)

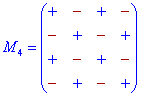
Для використання програми необхідною та достатньою умовою є встановлення на робочому девайсі користувача мови програмування Python (версії 3.+) та бібліотек NumPy та Itertools, котрі використовуються у програмі для полегшення проміжних завдань та покращення візуального сприйняття виведених даних.

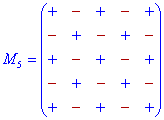
1. ПРОЕКТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ
2. Методи вирішення задач

Перш за все для реалізації цього методу, треба знайти алгебраїчне доповнення для кожного елемента матриці.

**Алгебраїчне доповнення** *А[j,k]* – це мінор *M[j,k]*, взятий зі знаком "плюс" , якщо *j+k* – парне число і зі знаком "мінус" – якщо непарне  
алгебраїчне доповнення, формула  
**Матриця алгебраїчнихбдоповнень** - це матрия складена з визначників *А[j,k],j,k=1..n.*

Знаки мінорів спрощено можна подати у вигляді схем



Визначник будь-якого порядку n, згідно правила Лапласа, можна записати у вигляді суми по парних добутків елементів рядків (стовпців) на їх алгебраїчні доповнення.  
формула Лапласа  
**Алгебраїчне доповнення***А[j,k]*, як і мінор, це визначник на одиницю меншого порядку ніж головний визначник. Тому для обчислення визначника *n* порядку потрібно обчислити *n* визначників *n-1*порядку.  
На практиці визначники матриць через алгебраїчні доповнення розписують до тих пір, поки не отримають мінори *3* порядку, які знаходять за правилом Саррюса або трикутників.

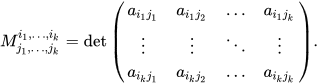
Практична реалізація для матриць більших *4* порядку складна, але реалізація таких алгоритмів на мові програмування через рекурентні формули значно спрощує обчислення.  
В навчанні переважно оперують з матрицями максимум *4,5* порядку. Якщо маємо розріджені матриці (багато елементів нульових) то визначник за рядком (стовпцем), який містить найбільшу кількість нульових елементів *k* зводиться до знаходження кількох *(n-k)* визначників на *1* меншого порядку від основного. Тому з допомогою елементарних перетворень спочатку перетворюють визначник, щоб отримати найбільше нульових елементів, а вже потім розписують його через алгебраїчні доповнення. Щоб Вас не навантажувати зайвою теорією перейдемо до практичної реалізації.

**Визна́чник** або **детерміна́нт** — це число; вираз складений за певним законом з n² елементів квадратної матриці. Одна з найважливіших характеристик квадратних матриць.

Для квадратної матриці розміру визна́чник є многочленом степеня *n* від елементів матриці, і є сумою добутків елементів матриці зі всіма можливими комбінаціями різних номерів рядків і стовпців (в кожному із добутків є рівно по одному елементу з кожного рядка і кожного стовпця). Кожному добутку приписується знак плюс чи мінус, в залежності від парності перестановки номерів.

Маючи означення визначника, можна перейти до розбору **мінорів**.

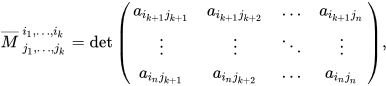
*Визначник* матриці, яка одержується з даної викреслюванням всіх рядків та стовпців, окрім вибраних, називається **мінором** *k*-го порядку, розташованим в рядках з номерами , ,..та стовпцях з номерами ,,… .



Якщо номери зазначених рядків збігаються з номерами зазначених стовпців, то мінор називається головним, а якщо відзначені перші *k* рядків і перші *k* стовпців - кутовим або провідним головним.

**Мінором** **елемента**  [квадратної матриці](https://uk.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BD%D0%B0_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8F) *A* порядку *n* називається визначник (*n-1*) порядку, який одержуємо з визначника *n*-го порядку шляхом викреслювання *і*-го рядка та *j*-го стовпця, на перетині яких знаходиться елемент .

Визначник матриці, яка одержується викреслюванням тільки вибраних рядків та стовпців з матриці *A* у випадку, коли отримана матриця буде квадратною, називається **доповнювальним мінором** до мінору :

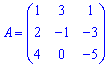
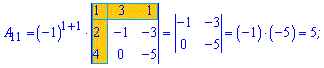
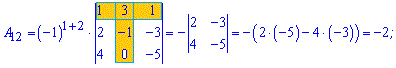
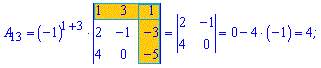
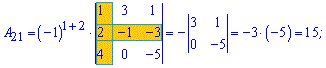
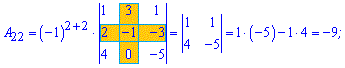
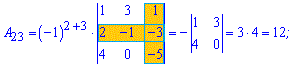
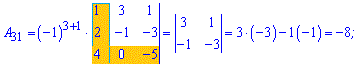
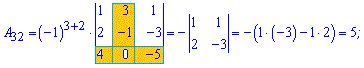
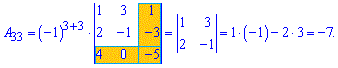


де <…< та <…< — номери не вибраних рядків і стовпців.

Нехай — деякий мінор порядку матриці *A*. Мінор порядку *k+1* матриці називається оточуючим для мінора , якщо його матриця містить в собі матрицю мінору . Таким чином, оточуючий мінор для мінора можна одержати дописуючи до нього один рядок і один стовпчик.

**Приклади:**

Знайти алгебраїчні доповнення матриці *3\*3*

1)   
Розв'язок. За наведеною вгорі схемою обчислюємо алгебраїчні доповнення, для цього знаходимо мінори з відповідними знаками  
  
  
  
  
  
  
  
  


Далі ми маємо транспонувати матрицю, отриману з алгебраїчних допонень.

Транспонування матриці називають таку операцію, коли міняють місцями рядки та стовпці матриці, зберігаючи порядок їх слідування.

Кожен елемент отриманої матриці ділимо на визначник початкової матриці.

1. Проектування алгоритмів

У ході розробки алгоритму роботи програми виникали нові завдання, вирішення яких варто було забезпечити у ході розробки програми. Далі наведені ці проблеми та рішення, прийняті для їх розв’язання:

1. Подача матриці у програмі у вигляді, зручному для використання та зрозумілому при виведенні. Цю проблему допомагає вирішити модуль NumPy, у котрому передбачені методи роботи з двохвимірними масивами, котрими і є матриці.
2. Забезпечення пошуку основної математичної основи для визначення мінору – визначника певної квадратної матриці довільної розмірності ( для більшої універсальності програми) без використання готових модулів. З цією метою була розроблена функція пошуку визначника з використанням циклів та методу перестановок із модуля Itertools.
3. Доступність створення матриці, пошуку її перестановок та їх кількості, добутку певних елементів, їх сума. Для цього було створено 5підпрограми, котрі для зручності було подано через функції. Одна з них дає можливсі обрати створення рандомної матриці: користувачу необхідно ввести лише її розмірність.
4. Перевірка введених даних. Для вирішення даної проблеми використовувались різні методи, такі як цикли while та допоміжні функції.
5. Забезпечення легкості взаємодії. З цією метою для кожного вводу користувача забезпечено повідомлення, яким має бути ввід для того чи іншого кроку.
6. Зручність використання. Для вирішення цього питання було застосовано цикли while, котрі дозволяють використовувати програму заново без повторного запуску.
7. Структура програмного забезпечення

Виконання загальної програми починається із підпрограми «Початок виконання». У ній виводиться інформація про виконавця, період створення програми та її призначення.

Наступна підпрограма, «Ввід розмірності матриці і обрання вводу», дає можливість користувачу обрати спосіб наповнення матриці: ручний або рандомний. Після неї можливі два шляхи подальшого ходу виконання програми.

Спільною для них обох є підпрограма «Введення розмірності матриці».

Наступна підпрограма – «Наповнення матриці»- теж є необхідною для обох варіантів ходу виконання програми. Якщо обрано «Ручне наповнення матриці», користувач вводить елементи матриці відповідно до їх індексу. Якщо обрано «Рандомне наповнення матриці» програма сама виконує це завдання.

Подальший хід підпрограм не відрізняється.

Підпрограма «Створення матриці» утворює матрицю із введених чи заданих комп’ютером чисел.

«Обчислення визначника» - програма обчислює визначник заданої вище матриці.

«Обчислення алгебраїчних доповнень і транспонування нової матриці» - обчислює алгебраїчне доповнення кожного елемента, створює з них нову матрицю і транспонує її.

«Ділення матриці на визначник – програма бере вже транспоновану матрицю і ділить кожен її елемент на визначник, що був порахований на етапі «Обчислення визначника» і заповнює результатами нову матрицю.

На етапі «Виведення результату» ця нова матриця виводиться як наслідок дії програми.

Підпункт «Пропозиція про повторне виконання» дає змогу користувачу скористатись програмою ще раз без перезапуску програми.

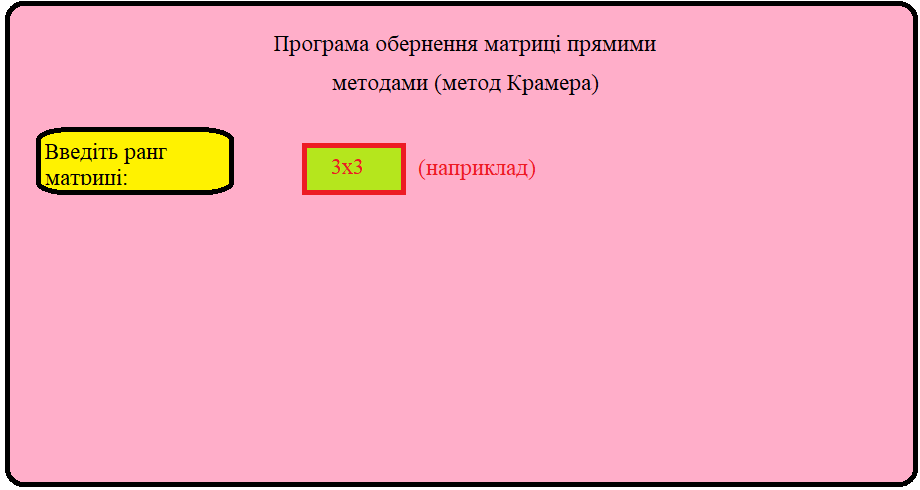
У разі відмови, користувач потрапляє на пункт «Завершення роботи», котрий закінчує виконання програми.

1. Опис розроблених алгоритмів

Загальний хід програми подано у блок-схемі в додатку А. Варто описати декілька функції, що застосовуються у програмі.

1. Random\_matrix(dim) – функція для створення рандомної матриці.
2. Permutations (n) – рекурсивна функція, що генерує усі можливі перестановки
3. Number\_permutations(n) – функція, що приймає один аргумент – розмірність квадратної матриці, і повертає список, до якого входять кортежі, склад яких – всі можливі перестановки чисел від 0 до n-1. Вони надалі використовуються як ітератори для пошуку визначника
4. Multiply(n, Matrix) – функція для множення певних елементів матриці. Усі такі добутки збираються в окремий список, далі елементи цього списку сумуються у функції sum(n, Matrix).
5. Minor(matrix, n, x, y) – тут ми обираємо спосіб наповнення матриці (якщо необхідно, вводимо її елементи), обчислюємо визначник, шукаємо алгебраїчні доповнення, транспонуємо матрицю та ділимо кожен її елемент на визначник.
6. Засоби керування програмами

**Ескізи інтерфейсу програми обернення матриці прямими методами (метод Крамера)**



**Рис. 1 Ескіз головного меню**



**Рис. 2 Ескіз меню заповнення матриці**

Завдяки тому, що у спеціалізованому інтерфейсі елементи, окрім введення даних, використовуються для вибору подальшого ходу виконання програми, її реалізація може бути доступною через командний рядок (події вибору в цьому випадку реалізуються через введення відповідей на запитання щодо подальшого ходу виконання (з указанням потрібних відповідей для кожного варіанту розвитку подій).

1. РЕАЛІЗАЦІЯ ПРОГРАМИ
2. Опис вхідних даних

У програмі передбачено 3 типи введених даних:

1. Розмірність матриці – ціле число, не менше 1. Обмежень зверху немає, проте вищою є продуктивність для невеликих значень.
2. Потрібні рядки та стовпці – цілі числа, не менші 1 і не більші кількості рядків/стовпців у матриці. Можуть бути застосовані як для опису елемента (що лежить на перетині вказаного рядка і стовпця), так і для опису мінору (на перетині вказаних рядків і стовпців) – у цьому випадку їх кількість обмежена розмірністю мінору.

Для взаємодії у командному рядку застосовується введення у діалоговому форматі. Приклад таких діалогів:

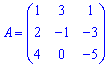
* «Enter matrix size» - у відповідь треба ввести розмірність матриці.
* «Еnter 1 if you want to create a random matrix\n Enter 0 if you want to create your own matrix» - введіть 1, якщо хочете прцювати з рандомною матрицею і 0, якщо з введеною.
* «Enter {k+1}, {n+1} matrix element:» потребують введення числа, що відповідає вказаному елементу.
* «Запустити програму заново(для підтвердження введіть "Так")?:» та схожі – запитання для забезпечення повторного виконання програми для певних значень без необхідності повного рестарту програми, приймають лише варіанти «Так» як вказівку до повторного виконання того чи іншого етапу та будь-який інший ввід як негативну відповідь.

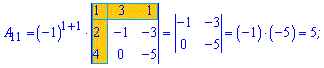
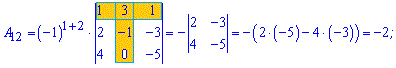
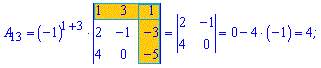
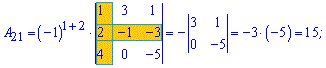
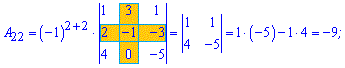
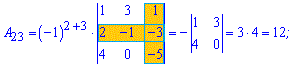
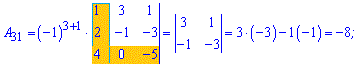
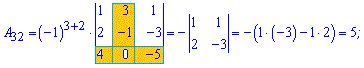
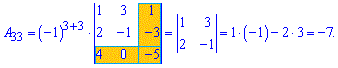
1. Опис результатів

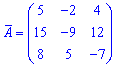
Результат дії програми:

* Введена (або рандомно створена) матриця та обернена матриця, котра виводяться на екран як двовимірний масив із модуля NumPy;

1. Опис контрольних прикладів



det **A** = 1·(-1)·(-5) + 3·(-3)·4 + 1·2·0 - 1·(-1)·4 - 1·(-3)·0 - 3·2·(-5) = 5 - 36 + 0 + 4 - 0 + 30 = 3  
Розв'язок. За наведеною вгорі схемою обчислюємо алгебраїчні доповнення, для цього знаходимо мінори з відповідними знаками  
  
  
  
  
  
  
  
  


Знайдені алгебраїчні доповнення записуємо у вигляді матриці  


Транспонуємо та ділимо

В результаті маємо:

[1.667, -0.667, 1.333]

[5.0, -3.0, 4.0]

[-2.667, 1.667, -2.333]

1. Експериментальні розрахунки

У ході тестування програми програми було перевірено працездатність програми, засвідчено недопуск переривання виконання програми в ході дій користувача (вводу хибних даних і т.д.), перевірено влучність підказок для юзерів, додано можливість повторного використання програми. Було перевірено виконання програми на контрольних прикладах і отримано результати, котрі співпали із отриманими вручну, що засвідчує правильність виконання програми. Детальніше із прикладами виконання програми можна ознайомитись у додатку В.

ВИСНОВКИ

Отже, у результаті виконання розрахунково-графічної роботи було вивчено обернення матриці, визначники та алгебраїчні доповнення як явища та методи їх розрахунку на основі математичних знань. Було отримано навички розробки алгоритму програми, написання схем взаємодії та блок-схем, вивчено можливості мови програмування Python для реалізації поставлених цілей. Було отримано навчики застосування мови програмування Python для втілення потрібних логічних та математичних операцій, досліджено застосування циклів та створення функцій для виконання тих чи інших підзавдань. На основі отриманих знань було розроблено програму обчислення обернення матриці методом Крамера, неодноразово перевірено її працездатність та звірено результати із тими, що були отримані в ході ручних розрахунків.

На основі тестування було розроблено та покращену схему зручної взаємодії між користувачем та програмою, створено ескізи інтерфейсу майбутньої програми та передбачено можливість застосування програми без додаткового інтерфейсу (зокрема через командний рядок).

ЛІТЕРАТУРА

1. Визначники // Вища математика в прикладах і задачах / Клепко В.Ю., Голець В.Л.. — 2-ге видання. — К. : Центр учбової літератури, 2009. — С. 13-15. — 594 с.
2. *Гантмахер Ф. Р.* Теория матриц. — 2 изд. — Москва : [Наука](https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%83%D0%BA%D0%B0_(%D0%B2%D0%B8%D0%B4%D0%B0%D0%B2%D0%BD%D0%B8%D1%86%D1%82%D0%B2%D0%BE)), 1967. — 576 с. — ISBN 5-9221-0524-8.
3. Алгебраїчне доповнення матриці. https://yukhym.com/uk/matritsi-ta-viznachniki/algebrajichni-dopovnennya-ta-minori.html

ДОДАТОК Б

Текст програми

import sys

import numpy as np

import itertools

print('Програма обернення матриці прямими методами(метод Крамера\n Виконала Гриб Вікторія')

def prog():

def number\_permutations(n):

if n == 0:

return 1

else:

return n\*number\_permutations(n - 1)

def random\_matrix(dim):

"""

The function generates dim x dim array of integers

between 0 and 10.

"""

matrix = np.random.randint(10, size = (dim, dim))

return matrix

def permutations(n):

"""

The function generates all permutations

"""

num = ""

for i in range(n):

num += str(i+1)

t = list(itertools.permutations(num, n))

return t

def multiply(n, Matrix):

"""

The function count result of multiplication

"""

res = list(n)

a = 0

while res != [str(i + 1) for i in range(len(res))]:

for i in range(len(res) - 1):

if int(res[i]) > int(res[i + 1]):

temp = res[i + 1]

res[i + 1] = res[i]

res[i] = temp

a += 1

result = 1

for j in range(len(res)):

result \*= Matrix[j][int(n[j]) - 1]

result \*= ((-1) \*\* a)

return result

def sum(n, Matrix):

"""

The function count the sum

"""

result = 0

for i in n:

result += multiply(i, Matrix)

return result

def minor(matrix, n, x, y):

matrix1 = []

a = 0

for k in range(n):

if k == x:

continue

else:

matrix1.append([])

for m in range(n):

if m == y:

continue

else:

matrix1[a].append(matrix[k][m])

a += 1

return sum(permutations(n-1), matrix1)

matrix = []

j = int(input('Enter matrix size '))

q = input('Enter 1 if you want to create a random matrix\n Enter 0 if you want to create your own matrix ')

if q == '1':

matrix = random\_matrix(j)

else:

for k in range(j):

matrix.append([])

for n in range(j):

matrix[k].append(int(input(f'Enter {k+1}, {n+1} matrix element:')))

for k in matrix:

print(k)

det = sum(permutations(j), matrix)

res\_matrix = []

for k in range(j):

res\_matrix.append([])

for n in range(j):

if (k % 2 == 1 and n % 2 == 0) or (k % 2 == 0 and n % 2 == 1):

res\_matrix[k].append((-1)\*minor(matrix, j, k, n))

else:

res\_matrix[k].append(minor(matrix, j, k, n))

res\_matrix[k][n] /= det

res\_matrix[k][n] = round(res\_matrix[k][n], 3)

for el in res\_matrix:

print(el)

while True:

res = input('Запустити програму заново? ')

if res == 'Так':

prog()

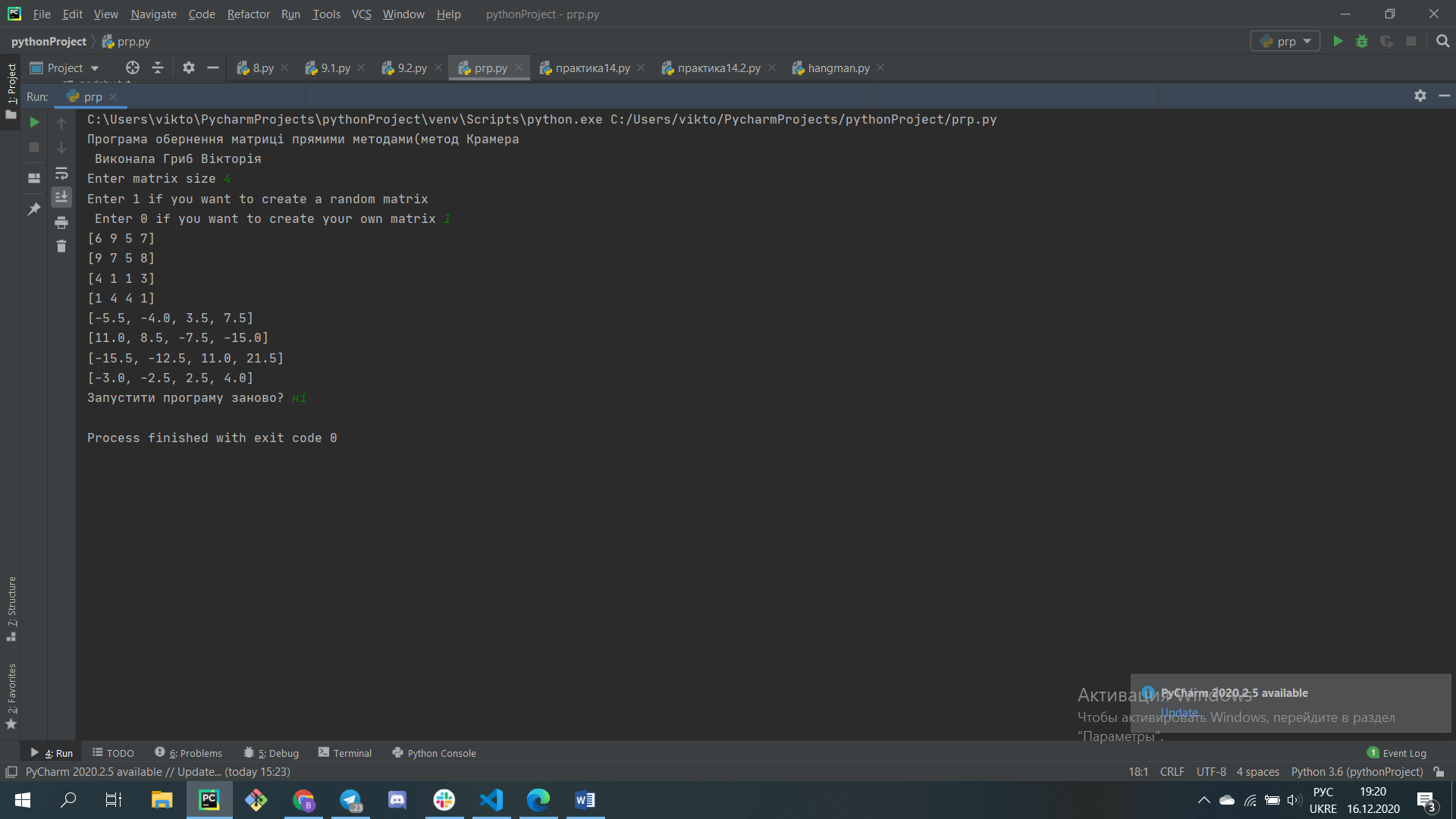
else:

sys.exit(0)

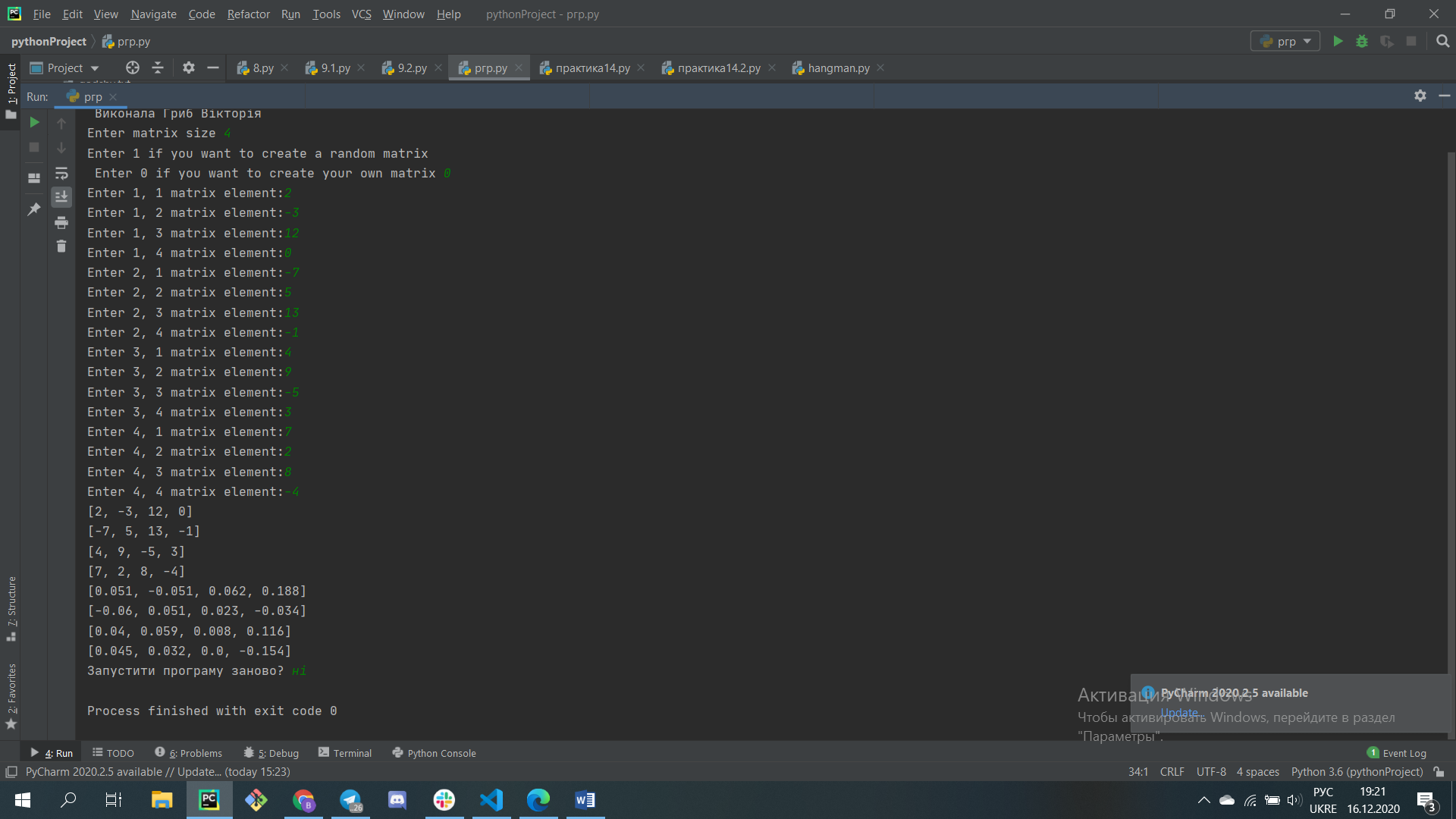
prog()

ДОДАТОК В

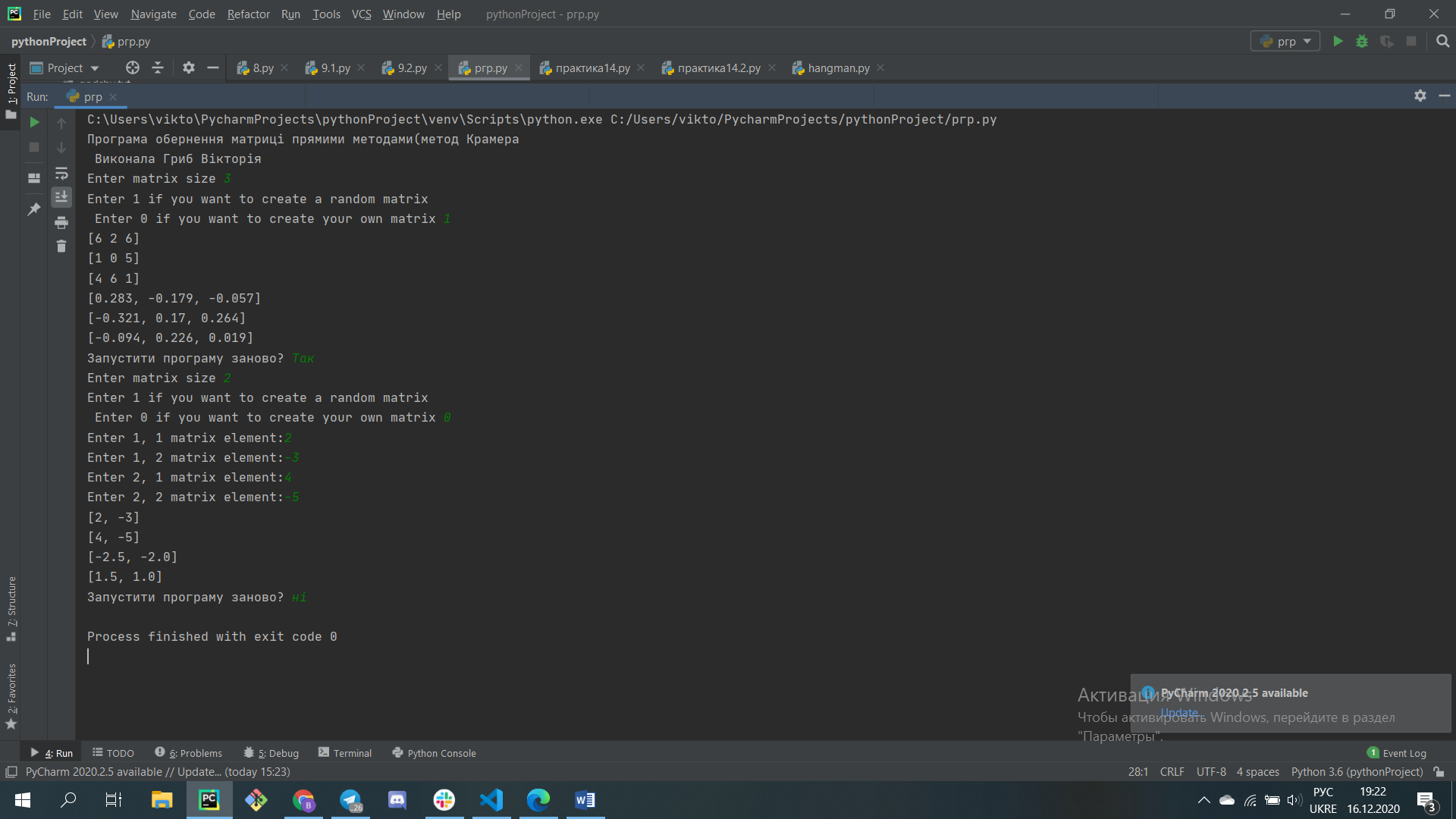
Експериментальні розрахунки



**Рис. 1 Тестування на прикладі 1**



**Рис. 2 Тестування на прикладі 2**



**Рис. 3 Тестування на прикладі 3**