Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання № 2

“ Побудова оберненої матриці методом Гауса-Жордана”

Варіант № 6

Виконав студент 2-го курсу

Групи ІПС-22

Каширець Роман Віталійович

Київ - 2024

**Завдання**

Побудова оберненої матриці методом Гауса-Жордана.

**Теорія**

**Метод Гаусса — Йордана** використовується для розв'язання [систем лінійних алгебраїчних рівнянь](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BB%D1%96%D0%BD%D1%96%D0%B9%D0%BD%D0%B8%D1%85_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D1%97%D1%87%D0%BD%D0%B8%D1%85_%D1%80%D1%96%D0%B2%D0%BD%D1%8F%D0%BD%D1%8C), знаходження [оберненої матриці](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D1%80%D0%BD%D0%B5%D0%BD%D0%B0_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8F), знаходження координат [вектора](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o "Вектор (математика)) у заданому базисі, відшукання [рангу матриці](https://uk.wikipedia.org/wiki/%D0%A0%D0%B0%D0%BD%D0%B3_(%D0%BB%D1%96%D0%BD%D1%96%D0%B9%D0%BD%D0%B0_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0)). Метод є модифікацією [методу Гаусса](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%93%D0%B0%D1%83%D1%81%D0%B0). Названий на честь [Карла Фрідріха Гаусса](https://uk.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BB_%D0%A4%D1%80%D1%96%D0%B4%D1%80%D1%96%D1%85_%D0%93%D0%B0%D1%83%D1%81%D1%81) та німецького математика та геодезиста [Вільгельма Йордана](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%B3%D0%B5%D0%BB%D1%8C%D0%BC_%D0%99%D0%BE%D1%80%D0%B4%D0%B0%D0%BD).

**Алгоритм**

1. Обирається перша зліва колонка, що містить хоч одне ненульове значення.
2. Якщо верхнє число у цій колонці — нуль, то обмінюється увесь перший рядок [матриці](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) з іншим рядком матриці, де у цій колонці нема нуля.
3. Усі елементи першого рядка діляться на верхній елемент обраної колонки.
4. Від рядків, що залишились, віднімається перший рядок, помножений на перший елемент відповідного рядка, з метою отримання нуля в першому елементі кожного рядка (крім першого).
5. Далі, повторюємо ці операції із матрицею, отриманою з початкової матриці після викреслювання першого рядка та першого стовпчика.
6. Після повторення операцій *n* − 1 разів отримаємо верхню трикутну матрицю.
7. Віднімаємо від передостаннього рядка останній рядок, помножений на відповідний коефіцієнт, щоб у передостанньому рядку залишилась лише 1 на головній діагоналі.
8. Повторюємо попередній крок для наступних рядків. У результаті отримуємо одиничну матрицю і рішення на місці [вільного вектора](https://uk.wikipedia.org/w/index.php?title=%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80&action=edit&redlink=1) (над ним необхідно виконувати ті самі перетворення).

**Складність алгоритму**

* **Часова складність**: O(n³), де n — це кількість змінних або розмір матриці (n × n).

Ця кубічна складність походить від необхідності виконання операцій на кожному етапі виключення для кожної з n рядків і стовпців.

Ось розбивка:

1. Для приведення матриці до трикутної форми (метод Гаусса) потрібно приблизно O(n³/3) операцій.
2. На етапі перетворення до одиничної матриці (Жордан) також потрібно близько O(n³/3) операцій.

В результаті загальна кількість операцій становить O(n³). Це робить метод Гаусса — Жордана не дуже ефективним для дуже великих матриць, оскільки його складність зростає швидко зі збільшенням розміру матриці.

**Мова реалізації алгоритму**

С++

**Модулі програми**

* + **Complex\*\* allocateMatrix(int n)**

Ця функція створює нову динамічну матрицю розміру n×nn \times nn×n та повертає вказівник на неї. Матриця складається з елементів типу Complex (комплексні числа).

* + **void deallocateMatrix(Complex\*\* matrix, int n)**

Функція очищає пам'ять, яка була виділена для матриці. Це важливо для запобігання витоків пам'яті.

* + **void printMatrix(Complex\*\* matrix, int n, string name)**

Ця функція виводить матрицю на екран, де name — це ім'я матриці для відображення.

* + **void gaussJordanInverse(Complex\*\* A, Complex\*\* I, int n)**

Ця функція виконує основну роботу зі знаходження оберненої матриці.

* **Приведення головного елемента до одиниці**: Для кожного рядка матриці AAA, починаючи з діагонального елемента A[i][i]A[i][i]A[i][i], всі елементи цього рядка діляться на діагональний елемент, щоб зробити його рівним 1.
* **Приведення інших елементів стовпця до нуля**: Для всіх рядків, крім поточного iii, необхідно занулити елементи в тому ж стовпці, щоб зробити всі елементи, окрім діагонального, рівними нулю. Це робиться шляхом віднімання кратного рядка iii з інших рядків.
* В процесі всіх цих операцій одночасно змінюється як початкова матриця AAA, так і матриця III, яка є одиничною на початку. Після виконання всіх операцій III стане оберненою матрицею до AAA.
* **Складність:** .

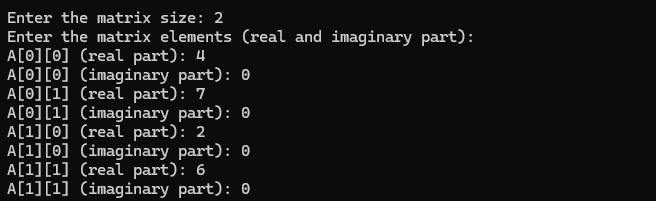
**Інтерфейс користувача**

Введення інформації відбувається через консоль спочатку програма просить користувача ввести розмір матриці(число) розміру n\*n.

Після чого користувач вводить реальну та уявну частини матриці.

**Тестові приклади**

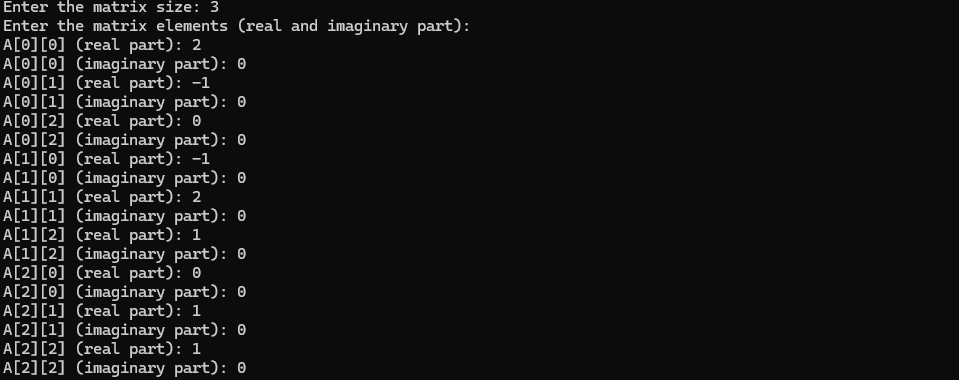
Приклад 1:

****

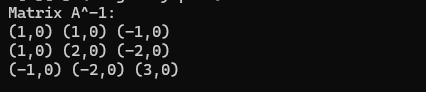
Відповідь до першого прикладу:

****

Приклад 2:



Відповідь до 2 прикладу:



**Висновки**Отже, у даній роботі, було описано побудову оберненої матриці методом Гауса – Жордана. Пояснен, що це таке, а також написано алгоритм побудови оберненої матриці даним методом. Написано код, що є прикріпленим до листа, а також пояснено модулі програми. Описано як працює інтерфейс користувача.  
Наведено 2 прикллади роботи програми.  
В даній роботі я освоїв, що таке метод Гауса – Жордана, а також навчився втілювати це за допомогою кода. Дізнався складність алгоритму.  
  
**Використані літературні джерела**Лекції і практичні заняття

<https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0_%E2%80%94_%D0%99%D0%BE%D1%80%D0%B4%D0%B0%D0%BD%D0%B0>