# Опис алгоритмів

Перелік основних змінних та їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| dimension | Розмірність матриці |
| method | Номер методу обернення матриці |
| matrix | Початкова квадратна матриця |
| determinant | Визначник матриці |
| markerInv | Маркер перевірки результату обернення. Якщо отримано NaN (внаслідок, наприклад, ділення на 0 або інших неправильних операцій), то маркер = false |
| tempMatrix | Тимчасова матриця |
| inversedmatrix | Обернена матриця |
| factor | Коефіцієнт для перетворення матриці |
| L | Нижньотрикутна матриця |
| U | Верхньотрикутна матриця |
| P | Масив перестановок |
| pivotRow | Опорний ряд |
| pivot | Опорний елемент |
| e | Вектор з одиничним елементом |
| x | Вектор рішень |
| y | Вектор проміжних даних |

## 3.1.Загальний алгоритм

1. ПОЧАТОК
2. Зчитати розмірність системи *dimension*.
3. Зчитати номер обраного методу *method*.
4. Створити квадратну матрицю *matrix* розмірності *dimension*
5. Зчитати матрицю системи з інтерфейсу:
   1. Цикл проходу по всіх рядках матриці інтерфейсу від 1 до *dimension* (ai – поточна строка):
      1. Цикл проходу по всіх стовпцях матриці інтерфейсу від 1 до *dimension* (aij – поточний елемент):
         1. ЯКЩО поточний елемент матриці – правильно записане число, ТО записати його в відповідну комірку *matrix*. ІНАКШЕ записати в комірку елемент NaN.
6. Знайти визначник *determinant* методом Гауса (підрозділ 3.2)
7. ЯКЩО *determinant* 0 і *method =* 1, ТО отримати обернену матрицю *invmatrix* згідно алгоритму методу LUP-декомпозиції (підрозділ 3.3). ІНАКШЕ вивести повідомлення помилки.
8. ЯКЩО *determinant* 0 і *method =* 2, ТО отримати обернену матрицю *invmatrix* згідно алгоритму методу Жордана-Гауса (підрозділ 3.4) ІНАКШЕ вивести повідомлення помилки.
9. Перевірити результат обернення:
   1. *markerInv* := true
   2. Цикл проходу по всіх рядках матриці інтерфейсу від 1 до *dimension* (ai – поточна строка):
      1. Цикл проходу по всіх стовпцях матриці інтерфейсу від 1 до *dimension* (aij – поточний елемент):
         1. ЯКЩО поточний елемент матриці = NaN (Not a Number), ТО *markerInv* := false
10. ЯКЩО *markerInv* = true, ТО вивести обернену матрицю *inversedmatrix* на головне вікно. ІНАКШЕ вивести повідомлення помилки.
11. КІНЕЦЬ

## 3.2. Алгоритм знаходження визначника методом Гауса

1. ПОЧАТОК
2. Створити копію матриці matrix - матрицю *tempMatrix* розмірності *dimension*
3. Процес виключення Гауса:
   1. Цикл *i* від 1 до *dimension-1*:
      1. Цикл *j* від 1 до *dimension*:
         1. *factor* := *tempMatrixji* / *tempMatrixii*
         2. Цикл *k* від 1 до dimension :
            1. *tempMatrixjk* = *tempMatrixjk*– *factor* \* *tempMatrixii*
4. Цикл *i* від 1 до *dimension-1*:
   1. *determinant* = *determinant* \* *tempMatrixii*
5. КІНЕЦЬ

## 3.3. Алгоритм LUP-декомпозиції

1. ПОЧАТОК
2. Створити матрицю *L* розмірності *dimension*
3. Створити матрицю *U* розмірності *dimension*
4. Створити матрицю *invmatrix* розмірності *dimension*
5. Створити масив *P* розмірності *dimension*
6. Заповнити матрицю *L* 0
7. Заповнити матрицю *invmatrix* 0
8. Скопіювати матрицю *matrix* у матрицю *U*
9. Цикл *і* від 1 до *dimension*:

9.1. *Pi* := *i* - 1

1. Цикл *k* від 1 до *dimension:*

10.1. *pivotRow* := *k*

10.2. *pivot* :=

10.3. Цикл *i* від *k* до *dimension*:

10.3.1. ЯКЩО :

10.3.1.1. *pivotRow* := i;

10.3.1.2. *pivot* :=

10.4. Поміняти місцями *Pk* і *PpivotRow*

10.5. Поміняти місцями *Uk* і *UpivotRow*

10.6. Поміняти місцями *Lk* і *LpivotRow*

10.7. Цикл *i* від *k+1* до *dimension*:

10.7.1. *Lik* := *Uik*/*Ukk*

10.7.2. Цикл *j* від *k*+1 до *dimension*:

10.7.2.1. *Uij := Uij - Lik\* Uki*

1. Цикл *i* від 0 до *dimension:*
   1. Створити масив *e* розмірності *dimension* та заповнити його 0
   2. ei = 0
   3. Створити масив *x* розмірності *dimension*
   4. Створити масив *y* розмірності *dimension*
   5. Цикл *j* від 0 до *dimension*:

11.5.1. *yj* :=

11.5.2. Цикл *k* від 0 до *dimension*:

11.5.2.1. *yj* := *yj – Ljk\*yk*

* 1. Цикл *j* від *dimension*-1 до 0:

11.6.1. *xj := yj*

11.6.2. Цикл *k* від *j+1* до *dimension*:

11.6.2.1. *xj := xj – Ujk\*xk*

11.6.3. *xj := xj / Ujj*

* 1. Цикл *j* від 0 до *dimension*:

11.7.1. *invmatrixji* := *xj*

12. КІНЕЦЬ

## 3.4. Алгоритм обернення матриці Жордана-Гауса

1. ПОЧАТОК
2. Створити матрицю *invmatrix* розмірності *dimension* і заповнити діагональні елементи 1, а недіагональні – 0
3. Процес виключення Гауса:
   1. Цикл *i* від 1 до *dimension-1*:
      1. Цикл *j* від 1 до *dimension*:
         1. *factor* := *invmatrix ji* / *invmatrix ii*
         2. Цикл *k* від 1 до *dimension*:
            1. *invmatrix jk* = *invmatrix jk*– *factor* \* *invmatrix ii*
            2. *matrix jk* = *matrix jk*– *factor* \* *matrix ii*
4. Цикл *i* від 1 до *dimension*:
   1. Цикл i від 1 до *dimension*:
      1. *invmatrix ij* = *invmatrix ij*– *matrixii*
5. КІНЕЦЬ