Hàm inputMatrix(n, A):

Hiển thị "Nhập các phần tử ma trận:"

Lặp qua từng phần tử của ma trận A (kích thước n x n):

Nhập giá trị phần tử vào A[i][j]

Hàm outputMatrix(n, A):

Thiết lập định dạng hiển thị số với 2 chữ số thập phân

Lặp qua từng phần tử của ma trận A (kích thước n x n):

Hiển thị phần tử A[i][j]

Hiển thị dòng mới sau mỗi hàng

Hàm multiMatrix(A, B, C, cola, rowa, rowb):

Lặp qua tất cả các phần tử của ma trận C:

Đặt giá trị ban đầu của C[i][j] là 0

Lặp qua các chỉ số k từ 0 đến rowa (số cột của A):

C[i][j] += A[i][k] \* B[k][j]

Hàm Danhilepski(A, M, M1, B, n):

Lặp từ k = n-2 đến k = 0:

Lặp qua tất cả các chỉ số i và j:

Nếu i ≠ k:

Nếu i == j:

M[i][j] = 1, M1[i][j] = 1

Khác:

M[i][j] = 0, M1[i][j] = 0

Ngược lại (i == k):

Gán giá trị M1[i][j] = A[k+1][j]

Nếu j == k:

M[i][j] = 1 / A[k+1][k]

Ngược lại:

M[i][j] = -A[k+1][j] / A[k+1][k]

Nhân ma trận A với M và lưu kết quả vào B

Nhân ma trận M1 với B và lưu kết quả vào A

Nhân ma trận B1 với M và lưu kết quả vào B

Sao chép kết quả B vào B1

Hàm determinant(A, n):

Nếu n == 1:

Trả về A[0][0]

Nếu n == 2:

Trả về A[0][0] \* A[1][1] - A[0][1] \* A[1][0]

Khác:

Khởi tạo giá trị det = 0

Lặp qua chỉ số f từ 0 đến n-1:

Khởi tạo ma trận con temp

Lặp qua chỉ số i từ 1 đến n-1:

Lặp qua chỉ số j từ 0 đến n-1:

Nếu j ≠ f, sao chép A[i][j] vào temp

Cộng vào det giá trị (-1)^f \* A[0][f] \* determinant(temp, n-1)

Trả về det

Hàm invertMatrix(A, inverse, n):

Tính định thức của A gọi là det

Nếu det == 0:

In ra "Matrix is singular and cannot be inverted."

Trả về false

Khởi tạo ma trận adjoint

Lặp qua từng chỉ số i và j:

Khởi tạo ma trận con temp

Lặp qua các chỉ số x và y từ 0 đến n-1:

Nếu x ≠ i và y ≠ j, sao chép A[x][y] vào temp

Tính adjoint[j][i] = (pow(-1, i + j) \* determinant(temp, n - 1)) / det

Chuyển ma trận adjoint vào ma trận inverse

Trả về true

Hàm solution(a, b, c, d, x[]):

Tính delta = b^2 - 3ac

Tính k = (9abc - 2b^3 - 27ad) / (2 \* sqrt(fabs(delta^3)))

Nếu delta > 0:

Nếu -1 <= k <= 1:

Tính các nghiệm x[0], x[1], x[2] bằng công thức bậc 3

Ngược lại:

Điều chỉnh giá trị k và tính x[0]

Nếu delta == 0:

Tính nghiệm x[0] dựa vào giá trị b^3 - 27ad

Nếu delta < 0:

Tính nghiệm x[0] dựa vào k và delta