UNIVERSITY OF MILA

Faculty of Science and Technology

Department of Process Engineering

Level: 1st year ST - ENG & LMD

Course : Introduction to Programming

Lectures 08:

Matrix Operations

---

by:

Dr. Farouk KECITA

Academic Year 2025/2026

1 Matrices and Linear Algebra in C

1.1 The Determinant of a Matrix in C

The determinant of a matrix is a special number that combines important information about the matrix into a single real number. It is defined only for square matrices, i.e., matrices with the same number of rows and columns.

The determinant is used in many places:

- To solve systems of linear equations
- To record how a linear transformation changes area or volume
- To find the inverse of a matrix

1.1.1 Determinant of a 2×2 Matrix

The determinant of the 2×2 matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is denoted by $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ and is defined as:

$$\det(A) = |A| = ad - bc$$

```c
#include <stdio.h>
int main() {
    int A[2][2], i, j;
    long det_A;
    printf("Enter the 4 elements of the matrix A:\n");
    for(i = 0; i < 2; i++)
        for(j = 0; j < 2; j++)
            scanf("%d", &A[i][j]);
    printf("The entered matrix is: \n");
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            printf("%d\t", A[i][j]);
        }
        printf("\n");
    }
    // finding the determinant of a 2x2 matrix
    det_A = A[0][0] * A[1][1] - A[1][0] * A[0][1];

    printf("The Determinant of the matrix A is : %ld\n", det_A);
    return 0;}
```

Listing 1: Determinant of a 2×2 Matrix

```
Enter the 4 elements of the matrix A:
1 2 3 4
The entered matrix is:
1 2
3 4
The Determinant of the matrix A is : -2
```

1.1.2 Determinant of a 3×3 Matrix

For a 3×3 matrix:

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

The determinant is given by:

$$\det(A) = a(ei - fh) - b(di - fg) + c(dh - eg)$$

```c
#include <stdio.h>
int main() {
    int a, b, c, d, e, f, g, h, i;
    int det_A;

    printf("Enter the elements of the 3x3 matrix:\n");
    scanf("%d%d%d%d%d%d%d%d%d", &a, &b, &c, &d, &e, &f, &g, &h,
       &i);

    det_A = (a * (e * i - f * h)) - (b * (d * i - f * g)) + (c
       * (d * h - e * g));

    printf("Determinant of the matrix is: %d\n", det_A);
    return 0;
}
```

Listing 2: Determinant of a 3×3 Matrix - Method 1

```
Enter the elements of the 3x3 matrix:
1 2 3 4 5 6 7 8 9
Determinant of the matrix is: 0
```

```c
#include <stdio.h>
int main() {
    int i, j, det_A = 0, A[3][3];

    printf("Input elements in the matrix A :\n");
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            printf("A[%d][%d]: ", i, j);
            scanf("%d", &A[i][j]);
```

```c
        }
    }
    // Display the matrix A
    printf("The matrix A is :\n");
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }
    // Calculate the determinant of the matrix A
    for (j = 0; j < 3; j++) {
        det_A = det_A + (A[0][j]*(A[1][(j+1)%3] * A[2][(j+2)%3]
            - A[1][(j + 2) % 3] * A[2][(j + 1) % 3]));
    }
    printf("The Determinant of A is: %d\n", det_A);
    return 0;
}
```

Listing 3: Determinant of a 3×3 Matrix - Method 2 (Using Array)

```
Input elements in the matrix A :
A[0][0]: 1
A[0][1]: 2
A[0][2]: 3
A[1][0]: 4
A[1][1]: 5
A[1][2]: 6
A[2][0]: 7
A[2][1]: 8
A[2][2]: 9
The matrix A is :
1 2 3
4 5 6
7 8 9
The Determinant of A is: 0
```

1.2 Trace of a Matrix

In linear algebra, the trace of a matrix is defined as the sum of the principal diagonal elements of a square matrix. It is usually represented as tr(**A**), where A is any square matrix of order $n \times n$.

For a 3×3 matrix:

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

The trace is:

$$\mathrm{tr}(A) = a + e + i$$

Algorithm to find Trace:

1. Initialize `trace = 0`
2. For `i = 0` to `n-1`
3. For `j = 0` to `n-1`
4. If `i == j`, then `trace += A[i][j]`
5. Print the result

```c
#include <stdio.h>
int main() {
    int A[10][10], n, i, j, trace;
    printf("Enter order of the square matrix : ");
    scanf("%d", &n);
    printf("\nEnter the elements of the matrix A: \n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &A[i][j]);
        }
    }
    // Display the matrix A
    printf("The matrix A is :\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }
    // find trace of the matrix
    trace = 0;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i == j) {
                trace = trace + A[i][j];
            }
        }
    }
    printf("Trace of the matrix A = %d \n", trace);
    return 0;  }
```

Listing 4: Program to Find the Trace of a Matrix

```
Enter order of the square matrix : 3

Enter the elements of the matrix A:
1 2 3 4 5 6 7 8 9
The matrix A is :
1 2 3
4 5 6
7 8 9
Trace of the matrix A = 15
```

1.3 Transpose of a Matrix

The transpose of a matrix is a new matrix that is obtained by exchanging its rows and columns. If we have a matrix A of size $m \times n$, then its transpose A^T will have size $n \times m$.

For a matrix:

$$A_{3\times 2} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

its transpose is:

$$A^T_{2\times 3} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

In C programming, the transpose of a matrix `A[n][m]` of size $n \times m$ is obtained by changing `A[i][j]` to `A[j][i]`.

```c
#include <stdio.h>
int main() {
    int A[10][10], tr_A[10][10], m, n, i, j;

    printf("Enter number of rows and columns:\n");
    scanf("%d %d", &m, &n);

    // assigning elements to the matrix A
    printf("\nEnter matrix elements:\n");
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            printf("Enter element A[%d][%d] = ", i, j);
            scanf("%d", &A[i][j]);
        }
    }

    // printing the matrix A
    printf("\nThe entered matrix is: \n");
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", A[i][j]);
        }
```

```c
        printf("\n");
    }

    // computing the transpose of A
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            tr_A[j][i] = A[i][j];
        }
    }

    // printing the transpose matrix
    printf("\nThe transpose of the matrix A is: \n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            printf("%d ", tr_A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Listing 5: Program to Find the Transpose of a Matrix

```
Enter number of rows and columns:
3 2

Enter matrix elements:
Enter element A[0][0] = 1
Enter element A[0][1] = 2
Enter element A[1][0] = 3
Enter element A[1][1] = 4
Enter element A[2][0] = 5
Enter element A[2][1] = 6

The entered matrix is:
1 2
3 4
5 6

The transpose of the matrix A is:
1 3 5
2 4 6
```

Summary of Matrix Operations

Operation	Description	Formula/Example
Dett (2×2)	Special number for square matrices	$\det = ad - bc$
Det (3×3)	Special number for square matrices	$\det = a(ei - fh) - b(di - fg) + c(dh - eg)$
Trace	Sum of diagonal elements	$\text{tr}(A) = \sum_{i=1}^{n} A[i][i]$
Transpose	Exchange rows and columns	$A^T[i][j] = A[j][i]$

Table 1: Summary of Matrix Operations in C

Key Takeaways:

- Multi-dimensional arrays allow storage of tabular data
- 2D arrays are stored in row-major order in memory
- Matrix operations are fundamental in scientific computing
- Understanding these concepts is essential for numerical methods and linear algebra applications