Matrix Calculator

Abdullah Khadim

Syed Mateen Ali

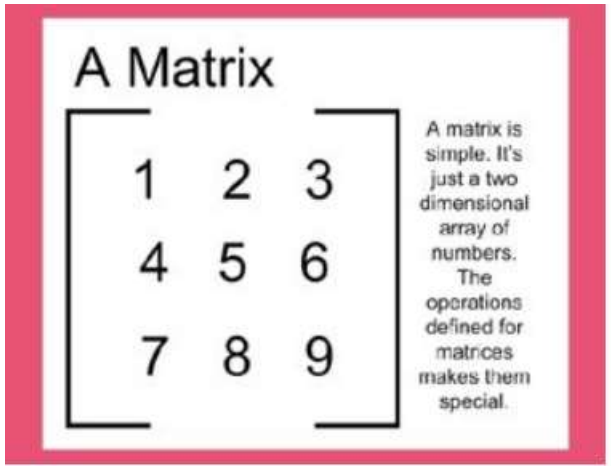
Bushra Abid

Contents

- What is matrix and what are operations performed on it
- How it helps in our daily lives
- Its use in Computer Science and real world
- Explaining Operations
- Algorithm and Flowchart
- Tools in C used for it
- How we make those operations in C language
- The Program

Matrix

- A matrix is a rectangular array of numbers, symbols, or expressions arranged in rows and columns, typically enclosed in square brackets. The numbers are called the elements or entries of the matrix, and the arrangement is used to represent data and solve systems of linear equations. A matrix's size is defined by its number of rows and columns, for example, an “m x n" matrix has “m” rows and “n” columns. for example



Why do we use matrices

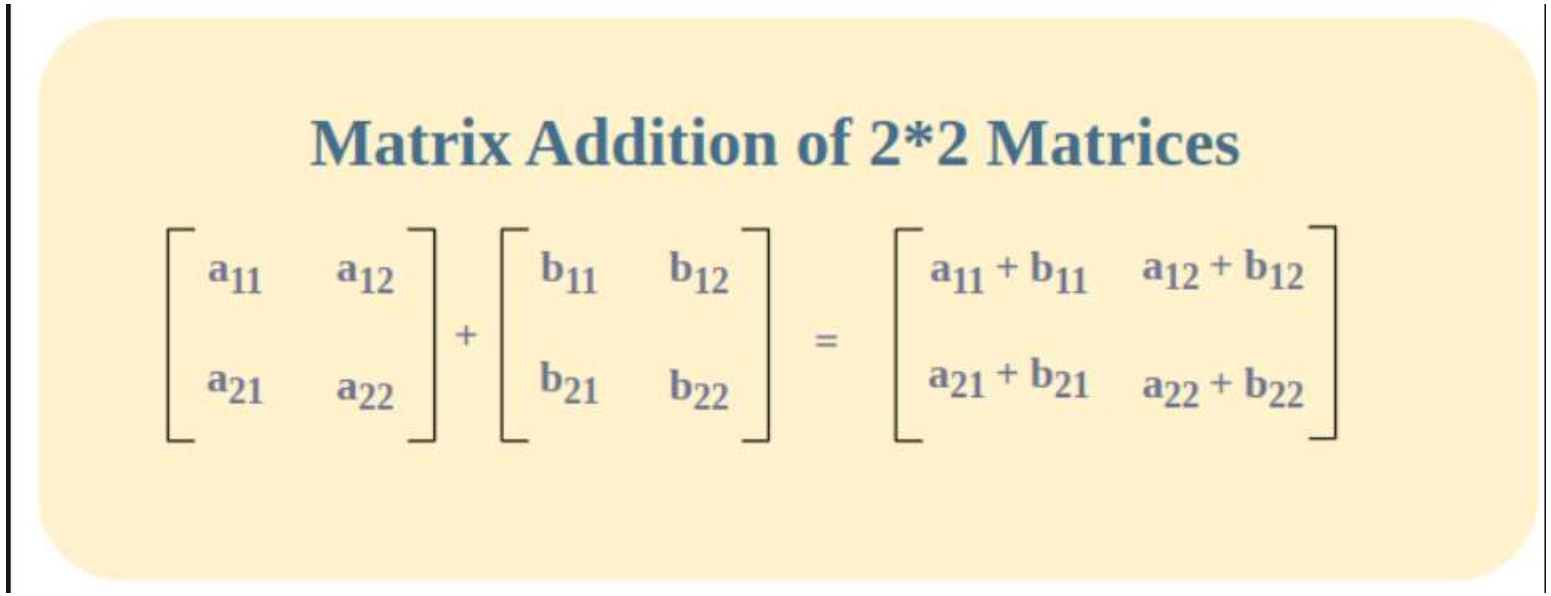
- Computer science and technology
- Computer Graphics: Matrices are used to represent and manipulate objects in 2D and 3D space, including transformations like scaling, rotation, and translation.
- Robotics: Robots use matrices to program their movements, calculate navigation, and perform tasks.
- Data Encryption: Matrices are used to encrypt and decrypt data, providing security for online transactions and communications.
- Page Ranking: Algorithms like the one used by Google use matrices to rank websites based on the links between them

Matrix Operations in CCP

- Additon
- Subtraction
- Multiplication
- Transpose
- Determinant

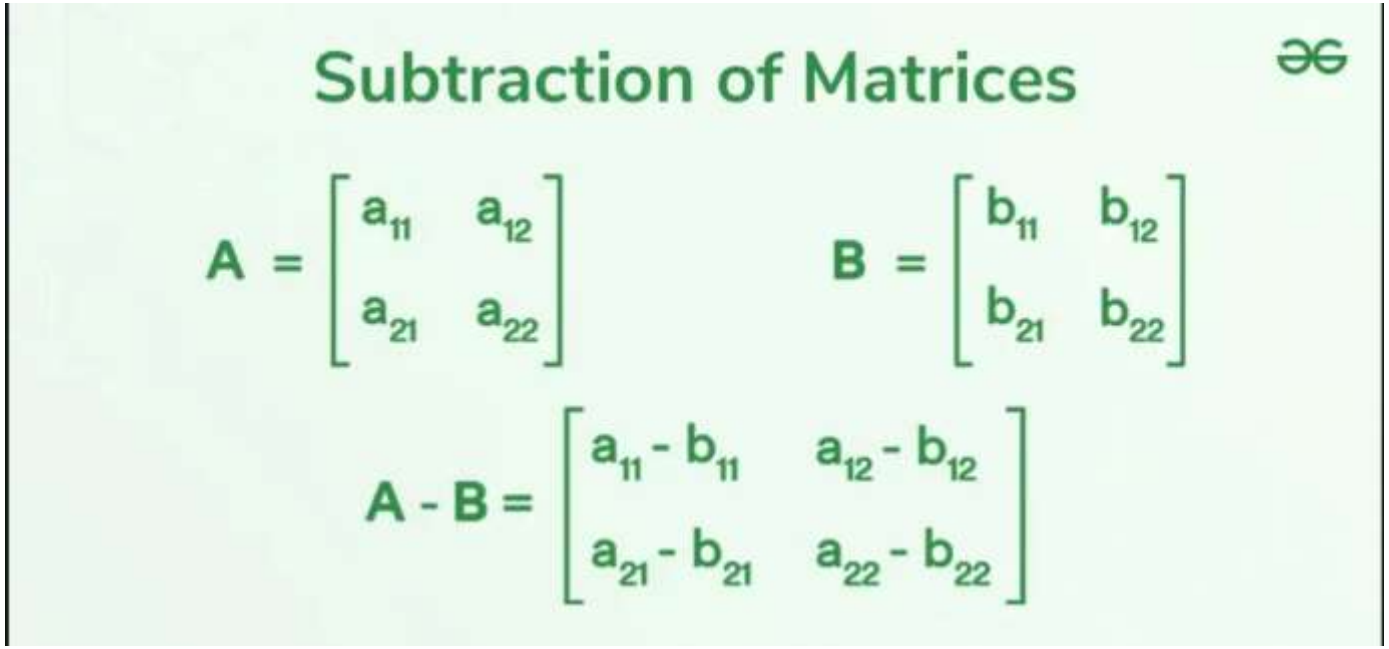
Matrix Addition

- The matrices that are to be added must have same order
- The elements of the same order are added with each other



Matrix Subtraction

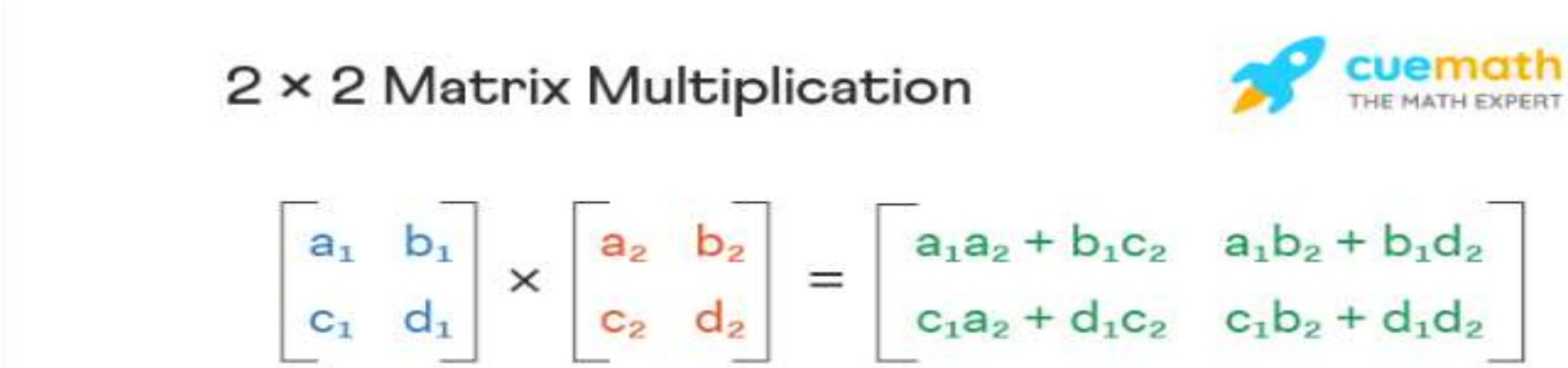
- Same order for both Matrices
- Element of same corresponding order will subtract



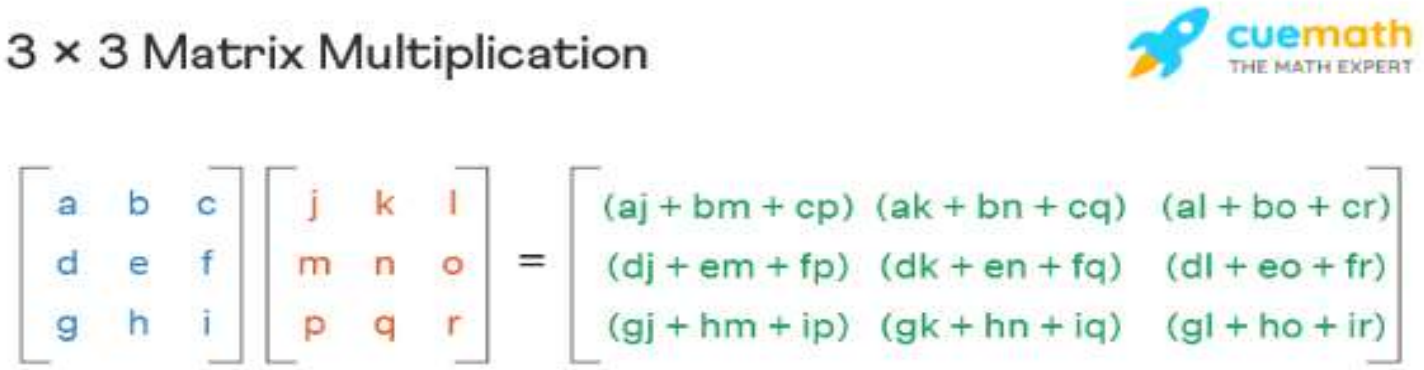
Matrix Multiplication

- The number of columns of first matrix should be same as the number of rows of second matrix.
- The multiplication will occur in order of elements of rows of first matrix to be multiplied by elements of columns of second matrix
- For example if A(m.n) x B(n.p) then then the product of A and B (matrix C) should be of order (m.p)

An 2 by 2 matrix multiplication



A 3 by 3 matrix multiplication



Transpose of a Matrix

- The matrix formed by swapping the order of elements in rows and columns ultimately swapping the order of matrix.
- For example if a matrix A has order of (2x3) then the order of its transpose At will be (3x2)
- The element at (2,1) will be at (1,2)

Transpose Operation

2	4	-1
-10	5	11
18	-7	6

2	-10	18
4	5	-7
-1	11	6

Transpose of a Matrix- examples

A	A^T
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$
$\lvert 5 \rvert$	$\lvert 5 \rvert$
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix}$	$\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$

Determinant of a Matrix

- The determinant is a scalar value computed for a square matrix that we use to find the inverse for matrix and at advance level it indicates properties of the linear transformation it represents, such as scaling factor and orientation.

A 2x2 matrix determinant

Determinant of a 2x2 Matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

$$\det \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} = (1)(4) - (2)(3) = -2$$

$$\det \begin{pmatrix} -5 & 2 \\ -2 & 0 \end{pmatrix} = \begin{vmatrix} -5 & 2 \\ -2 & 0 \end{vmatrix} = (-5)(0) - (2)(-2) = 4$$

$$\det \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} = \begin{vmatrix} 1 & 2 \\ 2 & 4 \end{vmatrix} = (1)(4) - (2)(2) = 0$$

A 3x3 matrix determinant

$$|A| = \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix}$$

$$= A_{11} \begin{vmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{vmatrix} - A_{21} \begin{vmatrix} A_{12} & A_{13} \\ A_{32} & A_{33} \end{vmatrix}$$

$$+ A_{31} \begin{vmatrix} A_{12} & A_{13} \\ A_{22} & A_{23} \end{vmatrix}$$

Calculating the 2x2 determinant in each term,

$$\begin{vmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{vmatrix} = A_{22}A_{33} - A_{23}A_{32}$$

$$\begin{vmatrix} A_{12} & A_{13} \\ A_{32} & A_{33} \end{vmatrix} = A_{12}A_{33} - A_{13}A_{32}$$

$$\begin{vmatrix} A_{12} & A_{13} \\ A_{22} & A_{23} \end{vmatrix} = A_{12}A_{23} - A_{13}A_{22}$$

We obtain

$$\begin{aligned} |A| &= A_{11}(A_{22}A_{33} - A_{23}A_{32}) \\ &\quad - A_{21}(A_{12}A_{33} - A_{13}A_{32}) \\ &\quad + A_{31}(A_{12}A_{23} - A_{13}A_{22}) \end{aligned}$$

$$\begin{aligned} &= A_{11}A_{22}A_{33} + A_{12}A_{23}A_{31} + A_{13}A_{21}A_{32} \\ &\quad - A_{13}A_{22}A_{31} - A_{12}A_{21}A_{33} - A_{11}A_{23}A_{32} \end{aligned}$$

Examples

Find the determinants of the following matrices.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 2 & 0 \\ 4 & 0 & 3 \end{bmatrix}$$

Sample answer

By the formula of 3×3 determinant,

$$\begin{aligned} |A| &= (1 \times 5 \times 9) + (2 \times 6 \times 7) + (3 \times 4 \times 8) \\ &\quad - (3 \times 5 \times 7) - (2 \times 4 \times 9) - (1 \times 6 \times 8) \\ &= 45 + 84 + 96 - 105 - 72 - 48 \\ &= 0 \end{aligned}$$

$$\begin{aligned} |B| &= (1 \times 2 \times 3) + (0 \times 0 \times 4) + (4 \times 0 \times 0) \\ &\quad - (4 \times 2 \times 4) - (0 \times 0 \times 3) - (1 \times 0 \times 0) \\ &= 6 + 0 + 0 - 32 - 0 - 0 \\ &= -26 \end{aligned}$$

Algorithm

Step 1: Start the program

Step 2: Display the title "Matrix Calculator"

Step 3: Ask the user to enter matrix size (2 or 3)

Step 4: If size is not 2 or 3 → display "Invalid size" and stop

Step 5: Input elements of the first matrix A

Step 6: Input elements of the second matrix B

Step 7: Display both matrices on the screen

Step 8: Show menu for operations:

- 1 → Addition
- 2 → Subtraction
- 3 → Multiplication
- 4 → Transpose
- 5 → Determinant

Step 9: Read the user’s choice

- **Case 1 — Addition**
- Compute res[i][j] = A[i][j] + B[i][j] for all elements
- Display result matrix
- **Case 2 — Subtraction**
- Compute res[i][j] = A[i][j] - B[i][j]
- Display result matrix
- **Case 3 — Multiplication**
- Set all elements of res to 0
- For each cell:
  res[i][j] = Σ(A[i][k] × B[k][j])
- Display result matrix
- **Case 4 — Transpose**
- Print transpose of matrix A: A[j][i]
- Print transpose of matrix B: B[j][i]
- **Case 5 — Determinant**
- If matrix size is **2×2**:
  det = a00*a11 – a01*a10 for both matrices
- If matrix size is **3×3**:
  Apply standard determinant formula
- Display the determinant values
- **Default**
- Display “Invalid Choice”
-

- **Step 10:** End the program

The Program

1. Setting the size

First we prompt the user to enter the size for the square matrix. The user will either enter 2 for 2x2 or 3 for 3x3 matrix.

In line 9 the compiler itself will set the size according to User input. If the user enter any other number it will Simply give the statement “Invalid matrix size”.

```
1  #include <stdio.h>
2
3  int main() {
4      int n, i, j, k, choice;
5      int det;
6      printf("=====MATRIX CALCULATOR=====\n");
7      printf("Enter matrix size (2 for 2 by 2 or 3 for 3 by 3): ");
8      scanf("%d", &n);
9      int a[n][n], b[n][n], res[n][n];
10
11     if (n != 2 && n != 3) {
12         printf("Invalid matrix size!\n");
13         return 0;
14     }
15
```

Declaring the elements

For declaring elements for matrix a and b we use two nested for loops the first loop is for the rows and the inner loop is for columns

i	j	a[i][j]
0	0	a[0][0]
0	1	a[0][1]
0	2	a[0][2]
1	0	a[1][0]
1	1	a[1][1]

```
16      printf("\nEnter elements of first matrix:\n");
17      for(i=0;i<n;i++){
18          for(j=0;j<n;j++){
19              printf("Enter element (%d,%d) : ",i+1,j+1);
20              scanf("%d",&a[i][j]);
21          }
22      }
23
24      printf("\nEnter elements of second matrix:\n");
25      for(i=0;i<n;i++){
26          for(j=0;j<n;j++){
27              printf("Enter element (%d,%d) : ",i+1,j+1);
28              scanf("%d",&b[i][j]);
29          }
30      }
```

Showing the Matrices

In these two nested for loops we display the Elements in matrix like format with only difference bieng the ‘\n’ operator after the inner loop ends. This is the trick for making that matrix format

A[i][j]	New line
A[0][0]	
A[0][1]	
A[0][2]	
	\n
A[1][0]	

```
31      printf("Matrix 1 : \n");
32      for(i=0;i<n;i++){
33          for(j=0;j<n;j++){
34              printf("%d ",a[i][j]);
35          }
36          printf("\n");
37      }
38      printf("\n");
39      printf("Matrix 2 : \n");
40      for(i=0;i<n;i++){
41          for(j=0;j<n;j++){
42              printf("%d ",b[i][j]);
43          }
44          printf("\n");
45      }
```

Operations

After displaying the matrices we ask the user to Enter number corresponding to the operations In the program user will enter 1 for addition or 2 For subtraction and so on. After the user make the choice we use the switch statement which contains the operations as cases

```
printf("Enter your choice of operation : \n");
printf("\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Transpose\n5.Determinant\n");

scanf("%d",&choice);
```

Addition

A[i][j]	B[i][j]	Res[i][j]
A[0][0]=1	B[0][0]=3	Res[0][0] = 4
A[0][1]=8	B[0][1]=6	Res[0][1] = 14
A[0][2]=4	B[0][2]=5	Res[0][2] = 9
A[1][0]=2	B[1][0]=4	Res[1][0] = 6
a[1][1]=8	B[1][1]=6	Res[1][1] = 14

```
52
53      switch(choice) {
54          case 1:
55              for(i=0;i<n;i++){
56                  for(j=0;j<n;j++){
57                      res[i][j] = a[i][j] + b[i][j];
58                  }
59              }
60              printf("\nAddition of the matrices:\n");
61              for(i=0;i<n;i++){
62                  for(j=0;j<n;j++){
63                      printf("%d ", res[i][j]);
64                  }
65                  printf("\n");
66              }
67              break;
68
```

Subtraction

A[i][j]	B[i][j]	Res[i][j]
A[0][0] = 1	B[0][0] = 3	Res[0][0] = -2
A[0][1] = 8	B[0][1] = 6	Res[0][1] = 2
A[0][2] = 4	B[0][2] = 5	Res[0][2] = -1
A[1][0] = 2	B[1][0] = 4	Res[1][0] = -2
A[1][1] = 8	B[1][1] = 6	Res[1][1] = 2

```
68
69          case 2:
70              for(i=0;i<n;i++){
71                  for(j=0;j<n;j++){
72                      res[i][j] = a[i][j] - b[i][j];
73                  }
74              }
75              printf("\nSubtraction of matrices :\n");
76              for(i=0;i<n;i++){
77                  for(j=0;j<n;j++){
78                      printf("%d ", res[i][j]);
79                  }
80                  printf("\n");
81              }
82              break;
83
```

Multiplication

For Multiplication we use 3 loops
First loop is for each row of matrix a the second loop is for the columns of matrix b the third loop is for multiplying a and b and saving it to the resultant matrix
We set resultant matrix to zero before multiplying to avoid saving garbage value from previous calculations
The multiplication will occur from multiplying the elements of Matrix a row wise to elements of matrix b column wise and add them.

Eg res[0][0] = 0 + a[0][0] x b[0][0];
res[0][0] = a[0][0] x b[0][0] + a[0][1] x b[1][0];
res[0][0] = a[0][0] x b[0][0] + a[0][1] x b[1][0] + a[0][2] x b[2][0];

After multiplying we use another nested for loop to display the Multiplied matrix

```
case 3:
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            res[i][j]=0;
            for(k=0;k<n;k++){
                res[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
    printf("\nMultiplication of matrices :\n");
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            printf("%d ", res[i][j]);
            }
        printf("\n");
    }
    break;
```

A[i][k]	B[k][j]	Res[i][j]
A[0][0]	B[0][0]	Res[0][0] = A[0][0]*B[0][0]
A[0][1]	B[1][0]	Res[0][0] = A[0][0]*B[0][0] +A[0][1]*B[1][0]
A[0][2]	B[2][0]	Res[0][0] = A[0][0]*B[0][0] +A[0][1]*B[1][0] + A[0][2]*B[2][0]
A[0][0]	B[0][1]	Res[0][1] = A[0][0]*B[0][1]
A[0][1]	B[1][1]	Res[0][1] = A[0][0]*B[0][1] +A[0][1]*B[1][1]

Transpose

For transposing we again use the two nested for displaying the transpose of both matrices the method to display the transpose is to simply change the order of matrix a[i][j] and b[i][j] to a[j][i] and b[j][i] .This will swapp The elements from row to columns and vice versa

A[i][j]	B[i][j]	Output 1	Output 2
A[0][0]	B[0][0]	A[0][0]	B[0][0]
A[0][1]	B[0][1]	A[1][0]	B[1][0]
A[0][2]	B[0][2]	A[2][0]	B[2][0]

```
case 4:
    printf("\nTranspose of first matrix:\n");
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            printf("%d ", a[j][i]);
        }
        printf("\n");
    }
    printf("\nTranspose of second matrix:\n");
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            printf("%d ", b[j][i]);
        }
        printf("\n");
    }
    break;
```

Determinant (2x2)

For finding determinant we use an if else statement if first if n == 2 then For 2 by 2 matrix we simply multiply the elements from 2 diagonals and subtract them

```
case 5:
    if(n==2){
        det = a[0][0]*a[1][1] - a[0][1]*a[1][0];
        printf("\nDeterminant of first matrix = %d\n", det);
        det = b[0][0]*b[1][1] - b[0][1]*b[1][0];
        printf("Determinant of second matrix = %d\n", det);
    }
```

$$\begin{pmatrix} 2 & 4 \\ 7 & 8 \end{pmatrix}$$

In this case a[0][0] = 2, a[1][1] = 8 , a[0][1] = 4 and a[1][0] = 7

So det = a[0][0]*a[1][1] – a[0][1]*a[1][0] = 16-28 = -12

Determinant (3x3)

If n is not 2 then it will probably be 3 so we calculate the 3x3 determinant in else statement. For 3x3 determinant we chose a number ‘a’ then find 2x2 matrix of only those elements which are not in the same row or column as ‘a’ and then we multiply the determinant by ‘a’

3	4	6
4	5	7
5	3	1

```
}
else {
    det = a[0][0]*(a[1][1]*a[2][2]-a[1][2]*a[2][1])
        - a[0][1]*(a[1][0]*a[2][2]-a[1][2]*a[2][0])
        + a[0][2]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]);
    printf("\nDeterminant of first matrix = %d\n", det);

    det = b[0][0]*(b[1][1]*b[2][2]-b[1][2]*b[2][1])
        - b[0][1]*(b[1][0]*b[2][2]-b[1][2]*b[2][0])
        + b[0][2]*(b[1][0]*b[2][1]-b[1][1]*b[2][0]);
    printf("Determinant of second matrix = %d\n", det);
}
break;
```

A[0][0] = 3
A[0][1] = 4
A[0][2] = 6
A[1][0] = 4
A[1][1] = 5
A[1][2] = 7
A[2][0] = 5
A[2][1] = 3
A[2][2] = 1
a[0][0]*(a[1][1]*a[2][2]-a[1][2]*a[2][1]) = 3x(5x1-7x3) = 3(5-21) = 3(-16) = -48
- a[0][1]*(a[1][0]*a[2][2]-a[1][2]*a[2][0]) = 4x(4x1-7x5) = 4(4-35) =-4(-29) = 116
+ a[0][2]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]) = 6(4x3-5x5) = 6(12-25) =6(13) = 78

Det = -48 + 116 +78 = 146

The End

- In the end we have default statement that if choice dosent matches any case it displays the invalid choice and our program ends

```
            break;

        default:
            printf("Invalid choice!");
    }

    return 0;
}
```