Programming Fundamentals

Aamina Batool

cin.ignore()

- **`cin.ignore`**()
- is used to skip the very next character.
- **`cin.ignore(100, '\n');`**
- it ignores either the next **`100`** characters or all characters until the newline character is found, whichever comes first. For example, if the next **`120`** characters do not contain the newline character, then only the first **`100`** characters are discarded and the next input data is the character **`101.`**
- **`cin.ignore(100, 'A');`**
- results in ignoring the first **`100`** characters or all characters until the character **`'A'`** is found, whichever comes first.

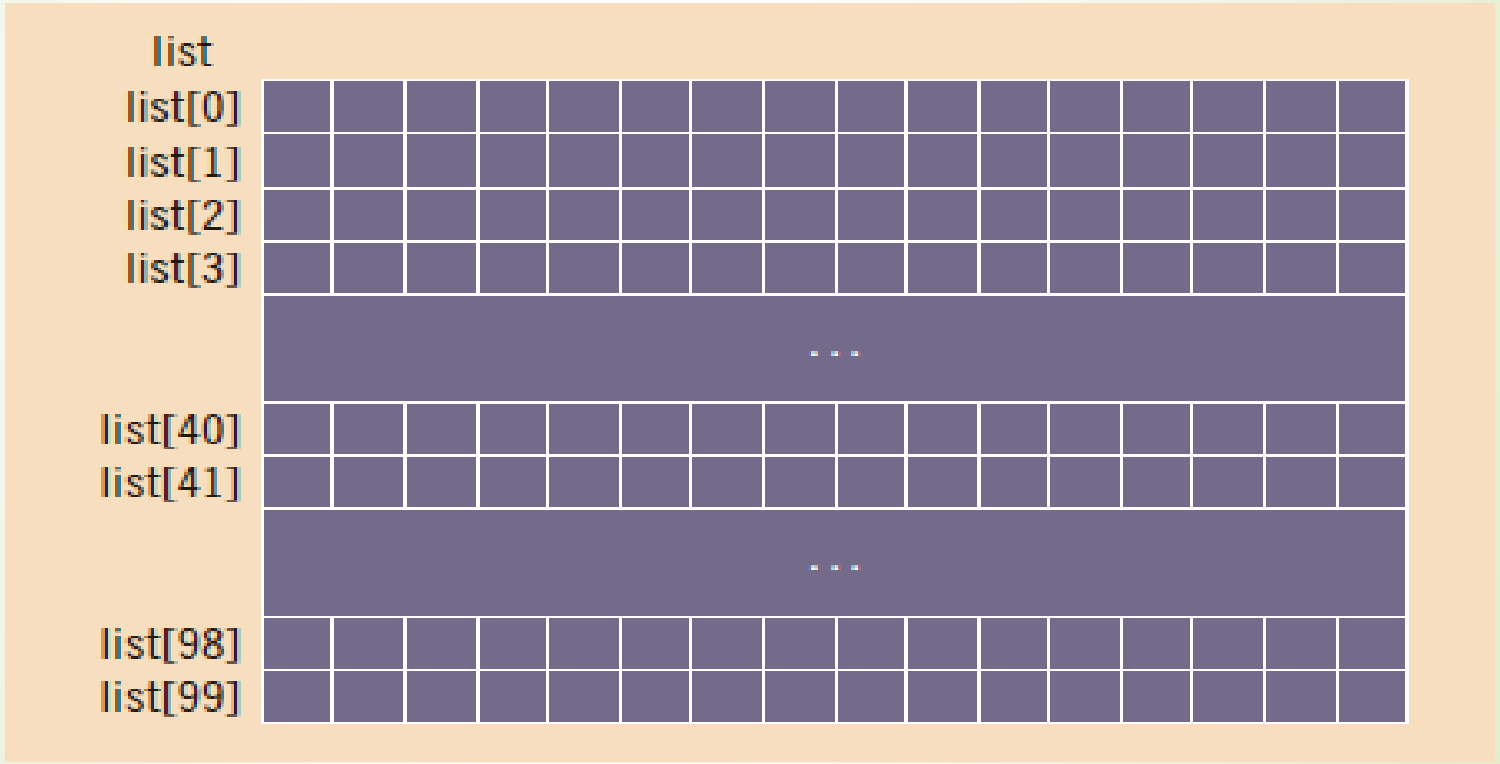
get and getline functions

- `char str1[26];`
- `char str2[26];`
- `char discard;`
- two lines of input:
- **`Summer is warm.`**
- **`Winter will be cold.`**
- Both **`str1`** and **`str2`** can store **C**-strings that are up to **`25`** characters in length. Because the number of characters in the first line is **`15`**, the reading stops at **`'\n'`**. Now the newline character remains in the input buffer and must be manually discarded.
- **`cin.get(str1, 26);`**
- **`cin.get(discard);`**
- **`cin.get(str2, 26);`**

- **We can also use `cin.ignore()` here to dicard newline character from the buffer**

2-D Character Arrays

- `char list[100][16];`



Inputting/outputting char 2-d arrays

- Suppose that you want to read and store data in **`list`** and that there is one entry per line.
- The following **`for`** loop accomplishes this task:

```
for (int j = 0; j < 100; j++)
    cin.get(list[j], 16);
```

- The following **`for`** loop outputs the string in each row:

```
for (int j = 0; j < 100; j++)
    cout << list[j] << endl;
```

- You can also use other string functions (such as **`strcmp`** and **`strlen`**) and **`for`** loops to manipulate **`list`**.

Reading/writing data from/to a list and there is one entry per line

```
char list[100][16];
for (int i=0; i< 100; i++)
{
        cin.get(list[i], 16);
        cin.ignore();
        cout << "iteration no. is: " << i + 1<< endl;
        cout << "input read in this iteration is: "<< endl;
        cout << list[i] << endl;
}
```

Task:

a) Comment out the second statement of loop see what input is stored in list.

b) Use getline instead of get and see if you still need to use cin.ignore() or not.

Palindrome

- A palindrome is a string, which when read in both forward and backward ways is the same.
- Example: pop, radar, madam, etc.

Palindrome

```
char list[20] = { ‘m’ , ‘a’ , ‘d’, ‘a’, ‘m’ , ‘\0’ }
int flag = 0;
int length = strlen(list);
for(i=0;i < length ;i++)
{
      If (list[i] != list[length-i-1])
            { flag = 1;
              break; }
 }
```

Matrix Addition

$$\begin{bmatrix} A11 & A12 \\ A21 & A22 \end{bmatrix} + \begin{bmatrix} B11 & B12 \\ B21 & B22 \end{bmatrix} = \begin{bmatrix} A11+B11 & A12+B12 \\ A21+B21 & A22+B22 \end{bmatrix}$$

For Example:

$$A = \begin{bmatrix} 0 & 1 \\ 4 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 5 \\ 3 & 6 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 0 & 1 \\ 4 & 2 \end{bmatrix} + \begin{bmatrix} 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 0+2 & 1+5 \\ 4+3 & 2+6 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 2 & 6 \\ 7 & 8 \end{bmatrix}$$

techcrashcourse.com

Matrix Subtraction

$$\begin{bmatrix} A11 & A12 \\ A21 & A22 \end{bmatrix} - \begin{bmatrix} B11 & B12 \\ B21 & B22 \end{bmatrix} = \begin{bmatrix} A11-B11 & A12-B12 \\ A21-B21 & A22-B22 \end{bmatrix}$$

For Example:

$$A = \begin{bmatrix} 0 & 1 \\ 4 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 5 \\ 3 & 6 \end{bmatrix}$$

$$A - B = \begin{bmatrix} 0 & 1 \\ 4 & 2 \end{bmatrix} - \begin{bmatrix} 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 0-2 & 1-5 \\ 4-3 & 2-6 \end{bmatrix}$$

$$A - B = \begin{bmatrix} -2 & -4 \\ 1 & -4 \end{bmatrix}$$

techcrashcourse.com

Matrix Transpose

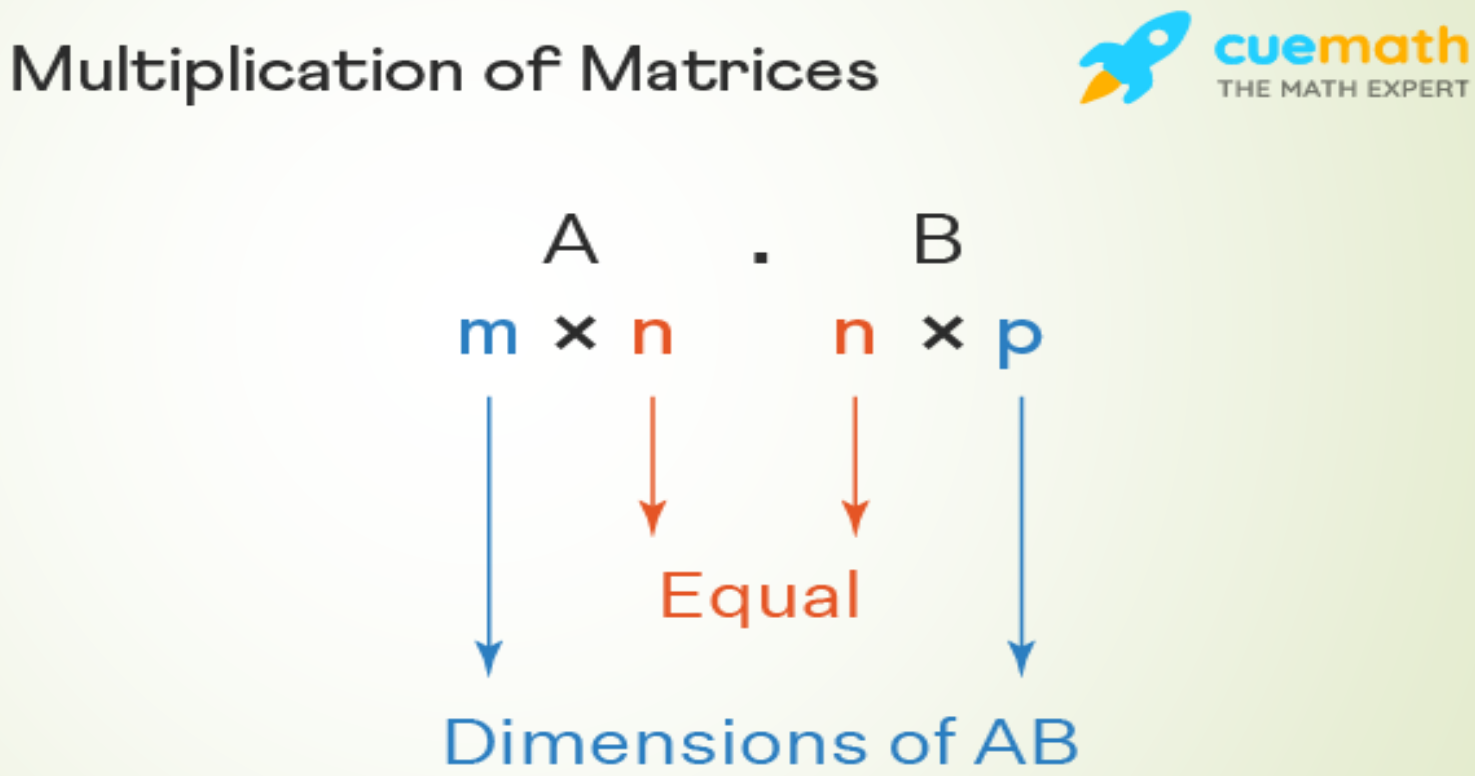
Transpose of a Matrix- examples

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

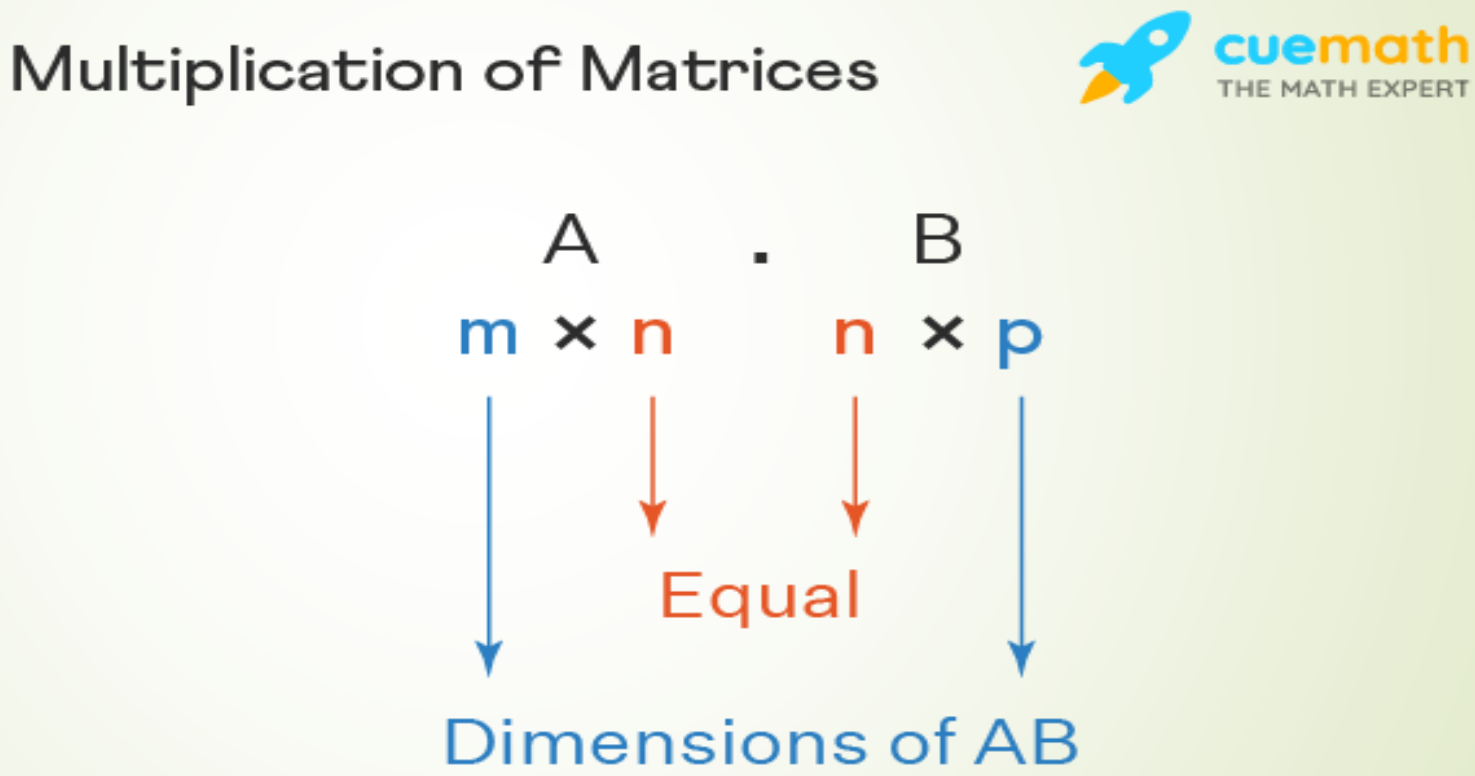
Matrix Transpose

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^{T} = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

Matrix Multiplication



Matrix Multiplication



Matrix Multiplication

2 × 2 Matrix Multiplication

cuemath
THE MATH EXPERT

$$\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} \times \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix} = \begin{bmatrix} a_1a_2 + b_1c_2 & a_1b_2 + b_1d_2 \\ c_1a_2 + d_1c_2 & c_1b_2 + d_1d_2 \end{bmatrix}$$

Matrix Multiplication

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41}$$

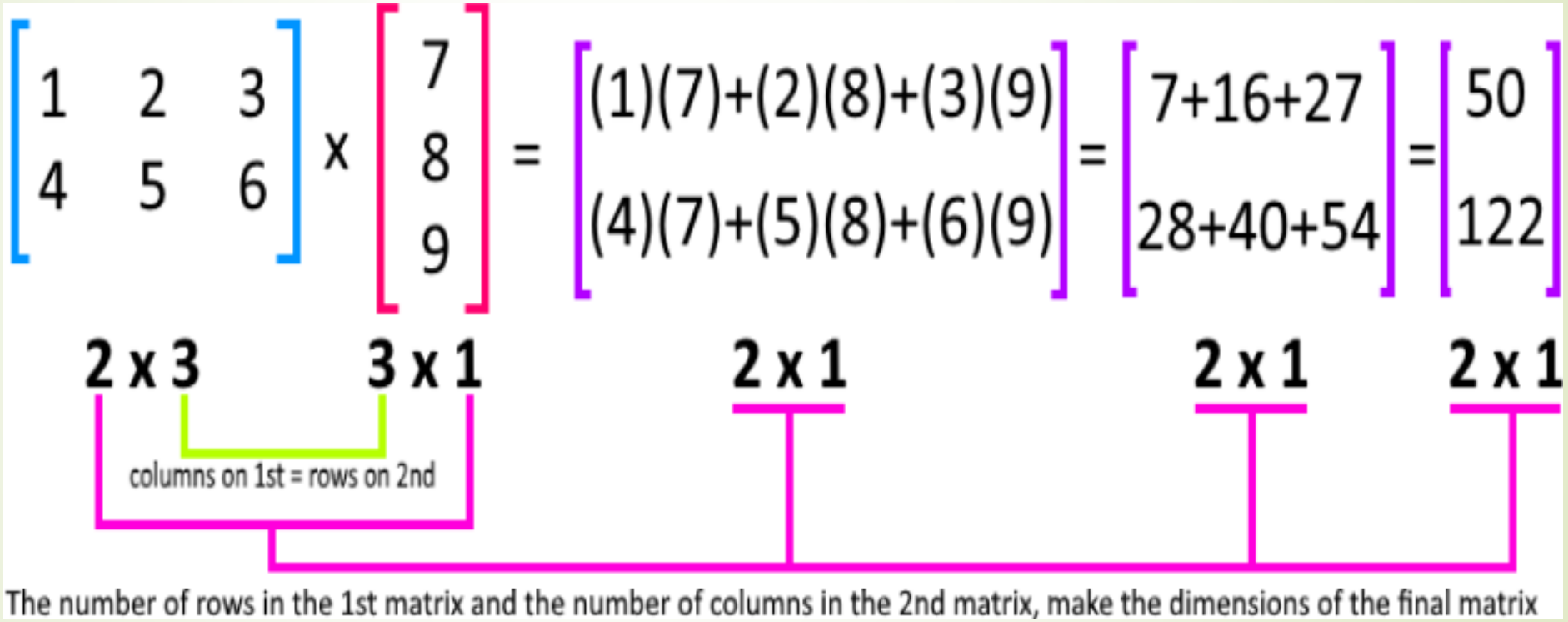
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

2 x 4 4 x 3 2 x 3

$$c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

Matrix Multiplication



Matrix Multiplication

- The algorithm for multiplication of matrices **A with order m*n** and **B with order n*p** can be written as:

```
for i from 0 to m-1
    for j from 0 to p-1
        c_ij = 0
        for k in 1 to n
            c_ij += a_ik*b_kj
```

Diagonal Matrix

Diagonal Matrix

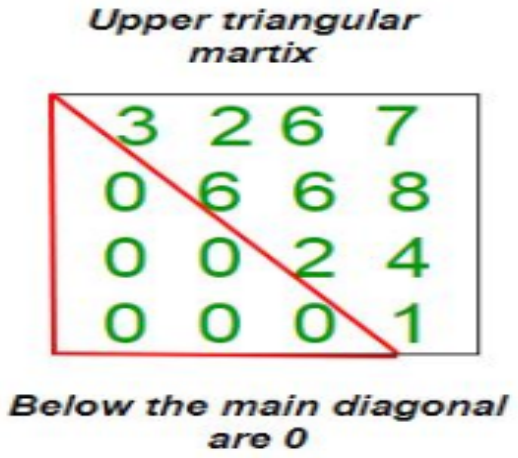
$$A = \begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

is a diagonal matrix when all entries that are not on the main diagonal are zero.

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \qquad A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Upper triangular matrix

An upper diagonal matrix is a square matrix in which all elements below the diago are zero. And the elements in the diagonal and above are different from zero. The following is an example of an upper diagonal matrix.



Write a function that receives a matrix as a parameter and returns **true** if it is an upper diagonal matrix.

Task: Also study lower triangular matrix.

Symmetric Matrix

Diagonal matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Symmetric matrix

$$\begin{bmatrix} 1 & 2 & 6 \\ 2 & 8 & 4 \\ 6 & 4 & 5 \end{bmatrix}$$

Symmetric Matrix
(transpose of matrix is same as original matrix)

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix}^{T}$$

Symmetric matrix

Exercises

- C++ Program to store temperature of two different cities for a week and display it.
- Find Column/Row wise max, min, average, sum.
- Sort the array Row/Column wise.
- Write a program for adding two matrices of size 2x2, take input from the user.
- Write a program for multiplying two matrices of size 2x2, take input from the user.
- Write a program to find transpose of a 3x3 matrix and a 3x2 matrix.
- Write a program to find inverse of a 2x2 matrix.

References

1. C++ Programming: From Problem Analysis to Program Design, Third Edition
2. https://www.just.edu.jo/~yahya-t/cs115/