National University of Computer and Emerging Sciences

# Lab Manual 03
# Object Oriented Programming – CL1004

| | |
|---|---|
| Course Instructor | Dr. Saira Karim |
| Lab Instructor(s) | Ms. Amna Zulfiqar<br>Mr. Muhammad Adeel |
| Section | BCS-2B |
| Semester | Spring 2023 |
| Date | 14-02-2023 |

Department of Computer Science
FAST-NU, Lahore, Pakistan

# Lab Manual 03 – Pointer to Pointer and 2D arrays

**Important Note:**

- **You may find the syntax to accomplish these exercises from lecture demo.**
- **Add Necessary Comments in you code to justify your logic.**
- **Comment exercise number or statement at the start of your code**
- **Save each exercise in .cpp file with your roll no, ex and lab number e.g.**
- **22LXXXX_EX01_Lab01.cpp**
- **Place all of your exercises in a folder a Zip it (Do not create .rar file) with roll no and lab no. e.g. 22LXXX_Lab01.zip**
- **Make sure that the interface of your program is user friendly i.e. properly display information.**
- **Properly follow the coding standards.**

1. **Exercise 1.1**: Write a function **int** AllocateMemory(int& rows, int& cols)** that takes size of matrix (rows and columns) from user, allocates memory for the matrix and return its pointer.
   What is the advantage of sending the two parameters by reference?

   **Exercise 1.2**: Write a function **void InputMatrix(int** matrix, const int& rows, const int& cols)** that inputs the matrix elements from user.
   Why are we passing the parameters as const?

   **Exercise 1.3**: Write a function **void DisplayMatrix(int** matrix, const int& rows, const int& cols)** that displays the matrix in proper format.

   **Exercise 1.4**: Write an inline function void **DeallocateMemory(int** matrix, const int& rows)** that deallocates all the memory.

   Make all the above mentioned functions inline and Test your program in main.
   An example run is given below.

```
Enter total rows:4
Enter total columns:3
The matrix is:
0 0 0
0 0 0
0 0 0
0 0 0
```

   (Suppose user entered 0 for all elements)

2. **Exercise**: Write a function called **MaxOfColumn** that takes as parameters a pointer to a 2D array and its dimensions. It should return the largest element in each column of the array. Since there is more than one column in 2D array, you have to return an array that contains largest of each column.
For example, if the Sample Matrix is

| | | | |
|---|---|---|---|
| 1 | 4 | 8 | 5 |
| 9 | 1 | 6 | 10 |
| 5 | 7 | 2 | 8 |

Your function will return array containing maximum elements of all the columns i.e. 9, 7, 8, 10. This is just a sample matrix; your program should run for any matrix/dimensions given by user. Display the returned array in main function and delete the dynamically allocated memory after using.

3. **Exercise:** Write a method **sortColumns** that will sort the each column of 2-D array into ascending order. Your program should run for any matrix/dimensions given by user. Display the original matrix entered by the user and then after sorting display the sorted matrix

4. **Exercise:** Write a program to demonstrate the use of pointer to pointer. Make a list of characters (a word) **by char *word**, another list of words (a sentence) using **char **sentence**. Print the sentence using a double pointer.

5. **Exercise** A square matrix in which all the entries below the main diagonal are zero is called upper triangular. Write a function **IsMatrixUpperTriangular** which takes a matrix and returns true if the matrix is upper triangular and false otherwise.
For example matrix A shown below is upper triangular while matrix B is not upper triangular.

| | | | |
|---|---|---|---|
| -1 | 2 | 7 | 0 |
| 0 | 5 | 0 | -1 |
| 0 | 0 | 7 | 0 |
| 0 | 0 | 0 | 0 |

Matrix A

| | | | |
|---|---|---|---|
| -1 | 2 | 7 | 0 |
| 0 | 5 | 0 | -1 |
| -9 | 0 | 7 | 0 |
| 0 | 0 | 0 | 0 |

Matrix B