Object Oriented Programming (OOP)

Semester: 2nd
Credit Hour: (01)
Department of Computer Science
Course Instructor: Engr. Zakria Bacha

Lab Task No. 3

Dynamic 2-D-Arrays in C++

Previous Lab:

In previous lab, we have covered Three topics:

1. **Pointer Arrays**
2. **Double Pointers**
3. **1-D Dynamic Arrays**

Current Topics for Lab:

1. **Creating a 2-D array dynamically**
2. **Storing data in a 2-D array**
3. **Processing data using nested loops**
4. **Using pointers to access 2-D arrays**

What is 2-D Dimensional Array in C++:

A 2-D array stores data in **rows and columns**, similar to a matrix. It is commonly used to represent tabular data such as marksheets, images, and matrices.

Static 2-D Array Declaration:

A static 2-D array is declared when the size is known at compile time.

```
23
24      int row = 3, col = 3;
25      int twoD_Array[row][col];
26
```

Nested for loops are usually used to store and display values in a 2-D array.

```
36      for(int row=0;row<3;row++)
37      {
38          for(int col=0;col<3;col++)
39          {
40              cin>>[row][col];
41          }
42
43      }
```

But How to Create 2D Array Dynamically in C++?

Answer For the above Question:

A dynamically created 2-D array in C++ is actually an array of pointers, where each pointer points to a 1-D array.

Creating a 1-D Dynamic Array

First, we Should know that how to create 1-Dimensional Array in C++? Below Figure shows the answer for the above-mentioned Question.

```
8
9       int *One_D_Array = new int[3];
10
11  //
```

Step 2: Using a Double Pointer

Here we have created **three (03)** 1-Dimensional Pointer arrays dynamically. Therefore, we need Double Pointer to store the base address of pointer array.

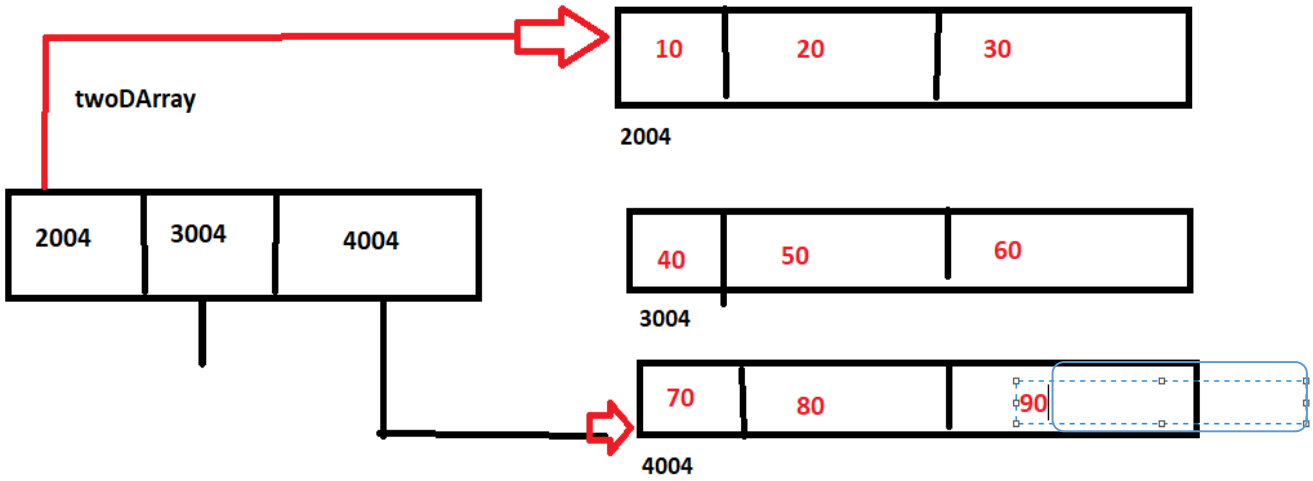
```
28      int row = 3, col = 3;
29      int **twoDarray = new int*[row];
30
```

Step 3: Allocating Memory for Rows and Columns

Here now we are creating three 1-D arrays of integer type and store the base addresses of each array in **twoDarray.**

```
31      for(int i=0;i<3;i++)
32      {
33          twoDarray[i] = new int[col];
34      }
```

Step-4: To see all the process visually.



Step 5: Show elements of 2-D array in C++:

```
36      for(int row=0;row<3;row++)
37      {
38          for(int col=0;col<3;col++)
39          {
40              cin>>[row][col];
41          }
42
43      }
```

How to use Pointer to show data of 2-D array:

Use below generalize form to process 2D array with pointer.

((a + i) + j)

a[0][0] can be accessed by using	:	**a (or) *(*(a+0)+0)
a[1][2] can be accessed by using	:	*(*(a+1)+2)
a[i][j] can be accessed by using	:	*(*(a + i) + j)

```
46      for(int i=0;i<3;i++)
47      {
48          for(int j=0;j<3;j++)
49          {
50              cout<< *(*(twoDarray+i)+j);
51          }
52          cout<<"\n";
53      }
```

Step 06: Deallocating Memory

This prevents memory leaks.

```
for(int i = 0; i < rows; i++)
{
    delete[] twoDArray[i];
}

delete[] twoDArray;
```

Summary:

Aspect	1-D Dynamic Array	2-D Dynamic Array
Definition	Stores elements in a single linear sequence	Stores elements in rows and columns (matrix form)
Pointer Used	Single pointer (int*)	Double pointer (int**)
Memory Allocation	Allocated using new[] for one dimension	Allocated using new[] for rows and then columns
Accessing Elements	arr[i]	arr[i][j]
Loop Required	Single loop	Nested loops
Memory Structure	Contiguous memory	Non-contiguous (array of pointers)
Example Use	List of numbers	Tables, matrices, marks of students
Complexity	Simple to manage	More complex memory handling
Deallocation	delete[] arr;	delete[] arr[i] for rows, then delete[] arr

Lab Task No.1

Implement a C++ program using a **dynamic 2-D array** to store student data for different universities and their departments. The array should have **at least 3 rows and 3 columns**, where each **row represents a university** and each **column represents a department** such as Computer Science, Software Engineering, and Data Science.

Note:

Dynamically allocate memory using a **double pointer (int**)**, take input for the number of students in each department, display the data in tabular form using **nested loops**, and properly **deallocate the memory** after use.

Input:

```
Enter students for Peshawar Campus - Computer Science: 120
Enter students for Peshawar Campus - Software Engineering: 95
Enter students for Peshawar Campus - Data Science: 60
```

Output:

```
Campus / Department        CS   SE   DS
-------------------------------
Peshawar Campus            120  95   60
```

National University of Computer and Emerging Sciences
Peshawar Campus

Lab Task No.2 (Enemy Hit Detection in a 3×3 Grid)

You are given a 3×3 grid where 1 shows a live enemy and 0 shows a dead one. Write a C++ program that initializes this grid and first counts how many live enemies are present. Then, repeatedly ask the user to enter row and column indices (0–2). If a live enemy is found at that position, mark it as dead and count it as a hit.
Keep taking input until all enemies are destroyed. Finally, display the total number of hits and print the updated grid.

```
1 0 1                    1 = Live enemy 0 = Dead enemy
1 1 0
0 1 1

Enter row and column (0-2): 0 0
Enter row and column (0-2): 0 2

Total live enemies: 6

Hit.  Enemy destroyed.
Hit.  Enemy destroyed.

All enemies are destroyed.
Total hits required: 6

Final Grid:
0 0 0
0 0 0
0 0 0
```