2 D ARRAY

2 D Array

- The 2D Array is also called as **matrix**.
- Initializing a 2D Array

  [storage class] data-type array-name[rowSize][columnSize];
  int arr[10][5];
- A 2D array is stored row-wise in memory, treating each row simple array.
- To access individual elements of 2D array both row and column indices are required.
- While initializing, second(column) dimension is compulsory, whereas the first(row) dimension is optional.

2D Array Declarations

- You can take values from user by running nested loop for both dimensions.
- int stud[4][2] = {{1,80},{2,75},{3,45},{4,67}};
- int stud[4][2] = {1, 80, 2, 75, 3, 45, 4, 67};
- int arr[2][3] = {12, 24, 45, 56, 78, 34};
- int arr[ ][3] = {12, 24, 45, 56, 78, 34};
- int arr[2][ ] = {12, 24, 45, 56, 78, 34};   //WRONG
- int arr[ ][ ] = {12, 24, 45, 56, 78, 34}; //WRONG

Memory Map of 2D Array

- int s[4][2] = { {1,80}, {2,75}, {3,45}, {4,67} };

s[0][0]	s[0][1]	s[1][0]	s[1][1]	s[2][0]	s[2][1]	s[3][0]	s[3][1]
1	**80**	**2**	**75**	**3**	**45**	**4**	**67**
65508	65512	65516	65520	65524	65528	65532	65536

- The array elements are stored in continuous chain

Accessing Elements of 2D array

- C treats parts of array as arrays
- int stud[4][2] → Array of 4 elements , each of which is 1D array containing 2 integers.
- Imagine stud to be an 1D array, then

➢ stud[0] → gives address of zeroth 1D array.

➢ stud[1] → gives address of first 1D array.

➢ stud[2] → gives address of second 1D array.

➢ stud[3] → gives address of third 1D array.

- We can access elements of 2D array by two ways :

1. Using Subscript notation
2. Using Pointer

- Suppose we want to access **s[2][1]** using pointers.

➢ **s[2]** will give address of second 1D array which is the address of **s[2][0]**.

➢ So we need to add 1 to **s[2]** to get next address.

➢ **(s[2] +1)** will give the address of **s[2][1]**

➢ ***(s[2]+1)** will give value at **s[2][1]**.

➢ ***(s[2]+1) → *(*(s+2)+1)**

Passing 2D Array to a Function

- A 2D Array can be passed to a function in two ways :

➢ Using Subscript notation
➢ Using Pointers

- Pass the array name(base address) and the dimensions to the function

Array Of Pointers

- The way there can be array of **int**s and floats, similarly, there can be array of pointers.
- An array of pointer would be collection of addresses.
- The addresses present in it can be :

➢Isolated variables

➢Addresses of array elements

➢Any other addresses