Structured Programming
CSE 103

Professor Dr. Mohammad Abu Yousuf

Multi-dimensional Array

- ***Syntax:***

  storage-class data- type array[expression1] [expression2] . . . [expression n] ;

  where storage-class refers to the storage class of the array, data- type is its data type, array is the array name, and expression 1, expression 2, . . ., expression n are positive-valued integer expressions that indicate the number of array elements associated with each subscript.

  Remember that the storage-class is optional; the default values are automatic for arrays that are defined inside of a function, and external for arrays defined outside of a function.

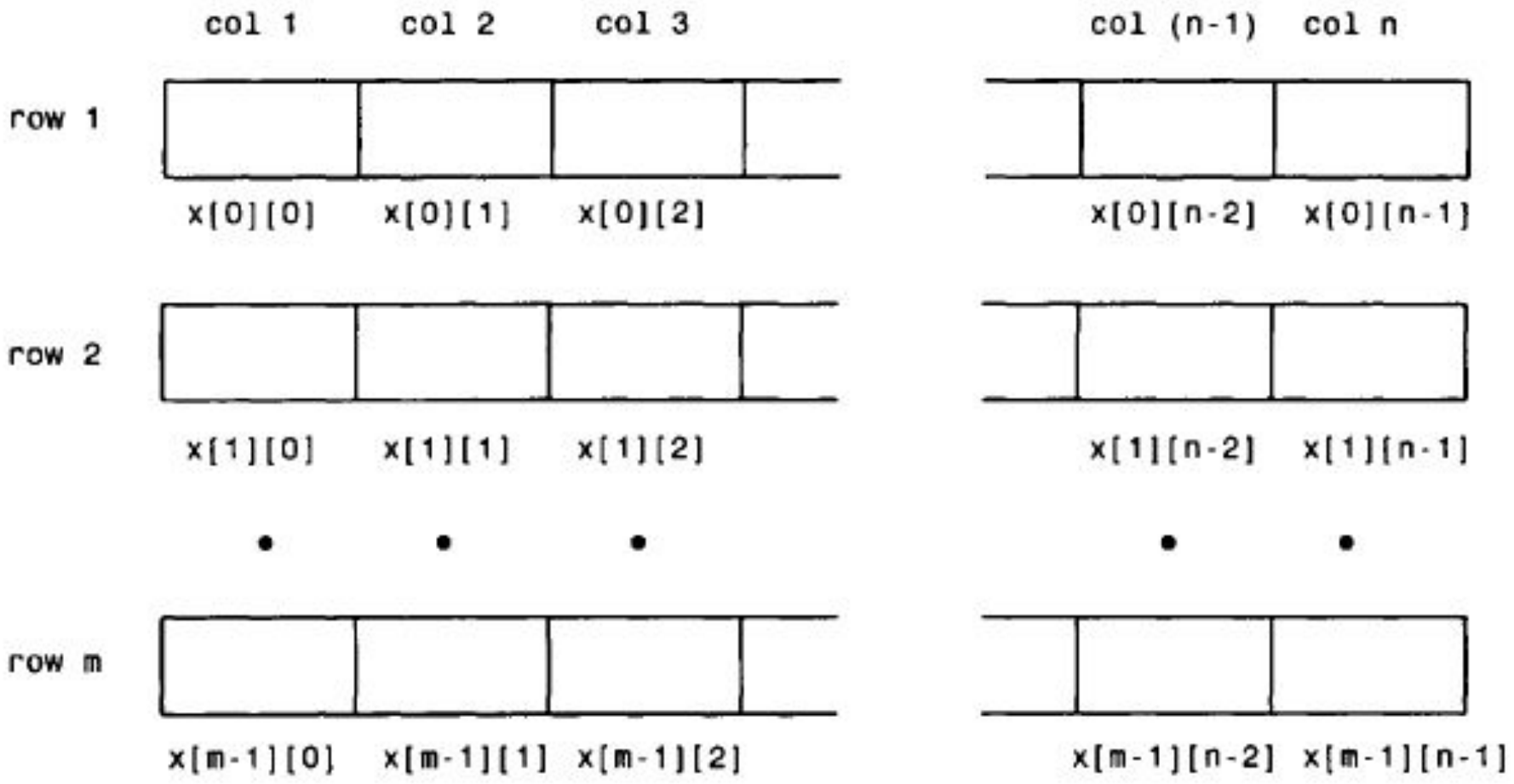
Multi-dimensional Array

float x[3][4];

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Multi-dimensional Array

- An ***m x n***, two-dimensional array can be thought of as a table of values having ***m*** rows and ***n*** columns



x is a m × n, two-dimensional array

- Example:

  float table[50][50];

  char page[24][80];

  double records[100][66][255];

  double records[L][M][N];

  int values[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

  Note that values can be thought of as a table having 3 rows and 4 columns (4 elements per row).

values[0][0] = 1 values[0][1] = 2 values[0][2] = 3 values[0][3] = 4
values[1][0] = 5 values[1][1] = 6 values[1][2] = 7 values[1][3] = *8*
values[2][0] = 9 values[2][1] = 10 values[2][2] = 11 values[2][3] = 12

- **Variation of the two-dimensional array definition**

```
int values[3][4] = {
                       {1, 2, 3, 4},
                       {5, 6, 7, 8},
                       {9, 10, 11, 12}
                   };
```

This definition results in the same initial assignments as in the last example.

Now consider :

```
int values[3][4] = {
                       {1, 2, 3},
                       {4, 5, 6},
                       {7, 8, 9}
                   };
```

This definition assigns values only to the first three elements in each row.

```
values[0][0] = 1    values[0][1] = 2    values[0][2] = 3    values[0][3] = 0
values[1][0] = 4    values[1][1] = 5    values[1][2] = 6    values[1][3] = 0
values[2][0] = 7    values[2][1] = 8    values[2][2] = 9    values[2][3] = 0
```

Multi-dimensional Array

```
int values[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
```

- Then three of the array elements will again be assigned zeros, though the order of the assignments will be different.

```
values[0][0] = 1    values[0][1] = 2    values[0][2] = 3    values[0][3] = 4
values[1][0] = 5    values[1][1] = 6    values[1][2] = 7    values[1][3] = 8
values[2][0] = 9    values[2][1] = 0    values[2][2] = 0    values[2][3] = 0
```

Multi-dimensional Array

- Finally, consider the array definition

```
int values[3][4] = {
                       {1, 2, 3, 4, 5},
                       {6, 7, 8, 9, 10},
                       {11, 12, 13, 14, 15}
                   };
```

- This will result in a compilation error, since the number of values in each inner pair of braces (five values in each pair) exceeds the defined array size (four elements in each row).

- Consider the following three-dimensional array definition

```
int t[10][20][30] = {
                        {                       /* table 1 */
                            {1, 2, 3, 4},        /* row 1 */
                            {5, 6, 7, 8},        /* row 2 */
                            {9, 10, 11, 12}      /* row 3 */
                        },
                        {                       /* table 2 */
                            {21, 22, 23, 24},    /* row 1 */
                            {25, 26, 27, 28},    /* row 2 */
                            {29, 30, 31, 32}     /* row 3 */
                        }
                    };
```

```
t[0][0][0] = 1    t[0][0][1] = 2    t[0][0][2] = 3    t[0][0][3] = 4
t[0][1][0] = 5    t[0][1][1] = 6    t[0][1][2] = 7    t[0][1][3] = 8
t[0][2][0] = 9    t[0][2][1] = 10   t[0][2][2] = 11   t[0][2][3] = 12

t[1][0][0] = 21   t[1][0][1] = 22   t[1][0][2] = 23   t[1][0][3] = 24
t[1][1][0] = 25   t[1][1][1] = 26   t[1][1][2] = 27   t[1][1][3] = 28
t[1][2][0] = 29   t[1][2][1] = 30   t[1][2][2] = 31   t[1][2][3] = 32
```

- All of the remaining array elements will be assigned zeros

Multi-dimensional Array

```
1.#include<stdio.h>
2.int main(){
3.int i=0,j=0;
4.int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
5.//traversing 2D array
6.for(i=0;i<4;i++){
7.    for(j=0;j<3;j++){
8.       printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
9.    }//end of j
10.}//end of i
11.return 0;
12.}
```

Output:

```
arr[0][0] = 1
arr[0][1] = 2
arr[0][2] = 3
arr[1][0] = 2
arr[1][1] = 3
arr[1][2] = 4
arr[2][0] = 3
arr[2][1] = 4
arr[2][2] = 5
arr[3][0] = 4
arr[3][1] = 5
arr[3][2] = 6
```

```
1.#include <stdio.h>
2.void main ()
3.{
4.    int arr[3][3],i,j;
5.    for (i=0;i<3;i++)
6.    {
7.        for (j=0;j<3;j++)
8.        {
9.            printf("Enter a[%d][%d]: ",i,j);
10.           scanf("%d",&arr[i][j]);
11.       }
12.   }
13.   printf("\n printing the elements ....\n");
14.   for(i=0;i<3;i++)
15.   {
16.       printf("\n");
17.       for (j=0;j<3;j++)
18.       {
19.           printf("%d\t",arr[i][j]);
20.       }
21.   }
22.}
```

2D array example: Storing elements in a matrix and printing it.

- C program to add two matrix:

```c
int main()
{
   int m, n, c, d, first[10][10], second[10][10], sum[10][10];

   printf("Enter the number of rows and columns of matrix\n");
   scanf("%d%d", &m, &n);
   printf("Enter the elements of first matrix\n");

   for (c = 0; c < m; c++)
      for (d = 0; d < n; d++)
         scanf("%d", &first[c][d]);

   printf("Enter the elements of second matrix\n");

   for (c = 0; c < m; c++)
      for (d = 0 ; d < n; d++)
            scanf("%d", &second[c][d]);

   printf("Sum of entered matrices:-\n");

   for (c = 0; c < m; c++) {
      for (d = 0 ; d < n; d++) {
         sum[c][d] = first[c][d] + second[c][d];
         printf("%d\t", sum[c][d]);
      }
      printf("\n");
   }

   return 0;
}
```

- Output of previous program:

```
Enter the number of rows and columns of matrix
2
2
Enter the elements of first matrix
1 2
3 4
Enter the elements of second matrix
5 6
2 1
Sum of entered matrices:-
6       8
5       5
```

Thank you