Jiangxi University of Science and Technology

Chapter 8 Arrays

- lecture0803 Two-Dimensional Arrays

8.5 Two-Dimensional Arrays

- Two-Dimensional Arrays
  - A two-dimensional array, or table, consists of both rows and columns of elements
  - **int** val[3][4];

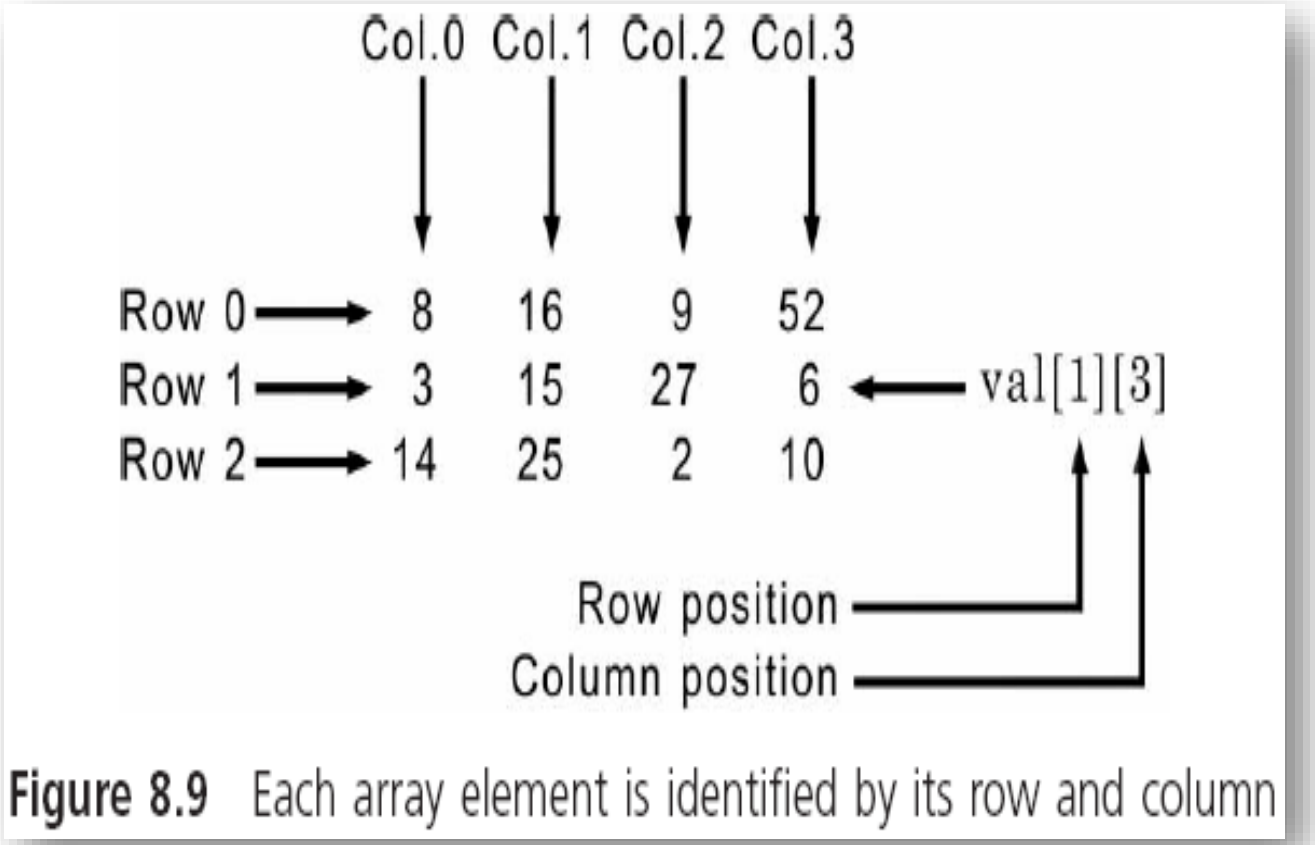

Figure 8.9 Each array element is identified by its row and column

8.5 Two-Dimensional Arrays

- Initialization:
  - int val[3][4] = { {8,16,9,52},
    {3,15,27,6},
    {14,25,2,10}};
  - The inner braces can be omitted:
  - int val[3][4]={8,16,9,52,3,15,27, 6,14,25,2,10};
  - Initialization is done in **row order**

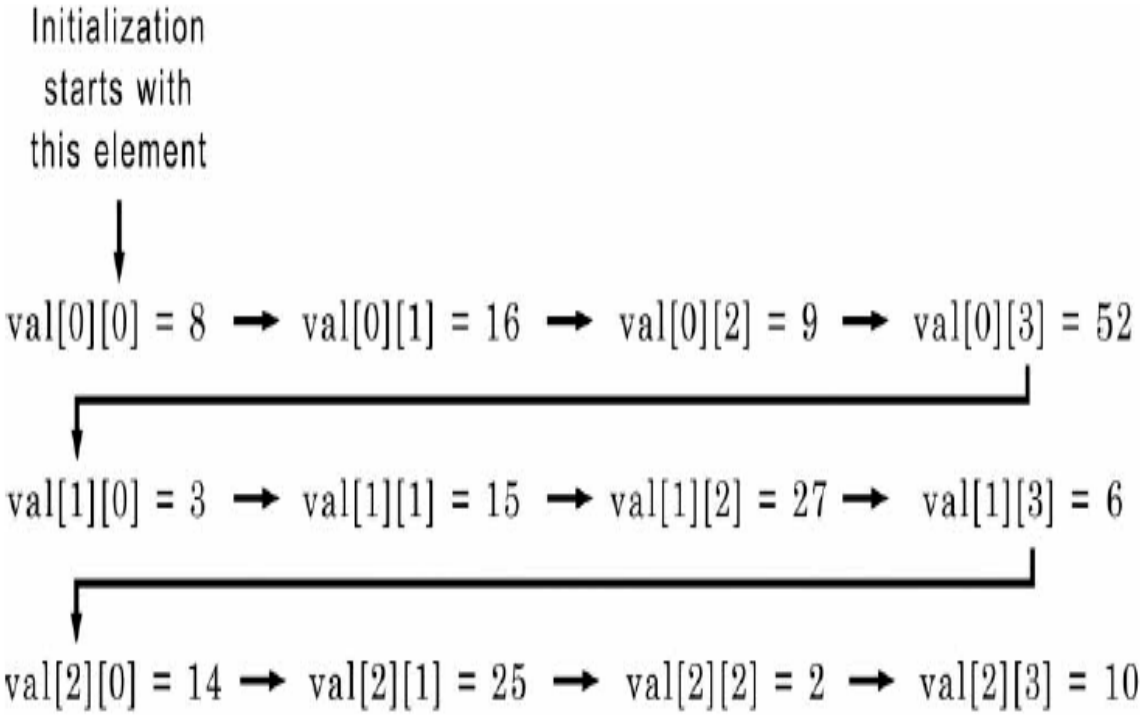

Figure 8.10 Storage and initialization of the `val[]` array

8.5 Two-Dimensional Arrays

Program 8.7 Display of array by explicit element

```
1.  #include <stdio.h>
2.  int main(){
3.          #define ROWS 3
4.          #define COLS 4
5.          int val[ROWS][COLS] = {8, 16, 9, 52, 3, 15, 27, 6, 14, 25, 2, 10};
6.          printf("\nDisplay of val array by explicit element");
7.          printf("\n%2d %2d %2d %2d", val[0][0], val[0][1], val[0][2], val[0][3]);
8.          printf("\n%2d %2d %2d %2d", val[1][0], val[1][1],val[1][2],val[1][3]);
9.          printf("\n%2d %2d %2d %2d", val[2][0], val[2][1], val[2][2],val[2][3]);
10.         printf("\n\nDisplay of val array using a nested for loop");
11.         for (int i = 0; i < ROWS; i++){
12.                 printf("\n"); //start a new line for each row
13.                 for (int j = 0; j < COLS; j++)
14.                         printf("%2d ", val[i][j]);
15.         }
16.         printf("\n");
17.         return 0;
18. }
```

Program 8.7
Display of array
using a nested for loop

8.5 Two-Dimensional Arrays

Program 8.8 multiply each element by 10 and display it

```
1.  #include <stdio.h>
2.  int main(){
3.          #define NUMROWS 3
4.          #define NUMCOLS 4
5.          int val[NUMROWS][NUMCOLS] = {8, 16, 9, 52, 3, 15, 27, 6, 14, 25, 2, 10};
6.          printf("\nDisplay of multiplied elements\n");
7.          for (int i = 0; i < NUMROWS; i++){
8.                  printf("\n"); /* start a new line */
9.                  for (int j = 0; j < NUMCOLS; ++j){
10.                         val[i][j] = val[i][j] * 10;
11.                         printf("%3d ", val[i][j]);
12.                 } /* end of inner loop */
13.         } /* end of outer loop */
14.         printf("\n");
15.         return 0;
16. }
```

8.5 Two-Dimensional Arrays

Program 8.8 Two-Dimensional Arrays as Function Arguments

```
1.  #include <stdio.h>
2.  #define ROWS 3
3.  #define COLS 4
4.  void display(int [ROWS][COLS]);
5.  int main(){
6.          int val[ROWS][COLS] = {8, 16, 9, 52, 3, 15, 27, 6, 14, 25, 2, 10};
7.          display(val);
8.          return 0;
9.  }
```

Program 8.8 Two-Dimensional Arrays as Function Arguments

```
11. void display(int nums[ROWS][COLS],int n){
12.         for (int i= 0; i< ROWS; i++)        {
13.                 for(int j= 0; j< COLS; j++)
14.                         printf("%4d",nums[i][j]);
15.                 printf("\n");
16.         }
17. }
```

Row size can be omitted

8.5 Two-Dimensional Arrays

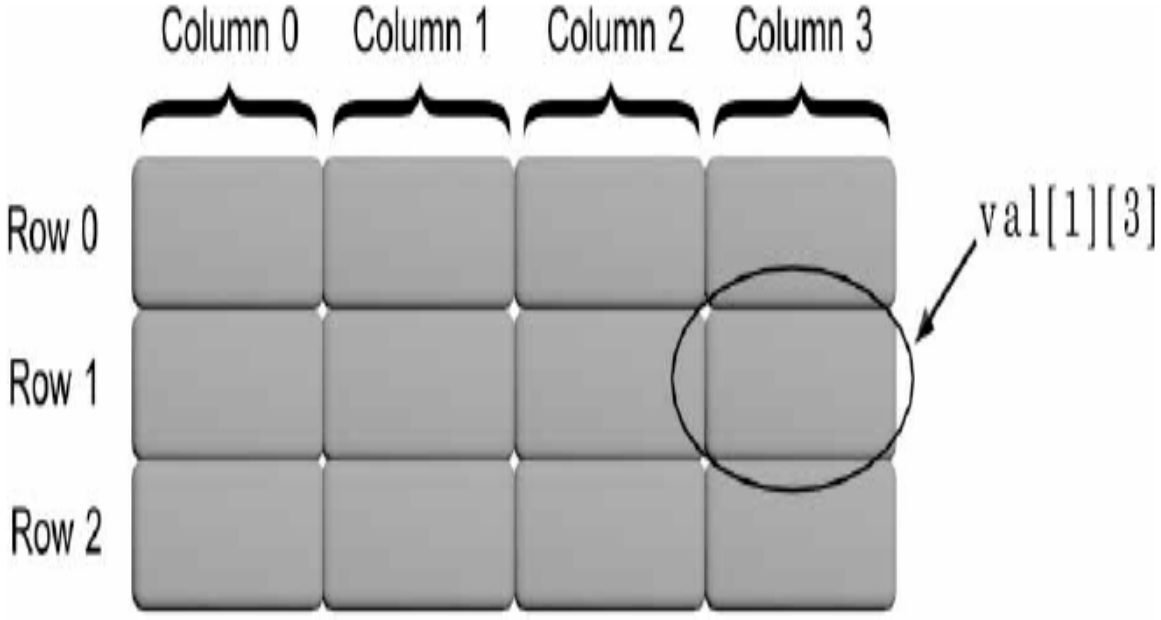

Figure 8.11 Storage of the `val` array

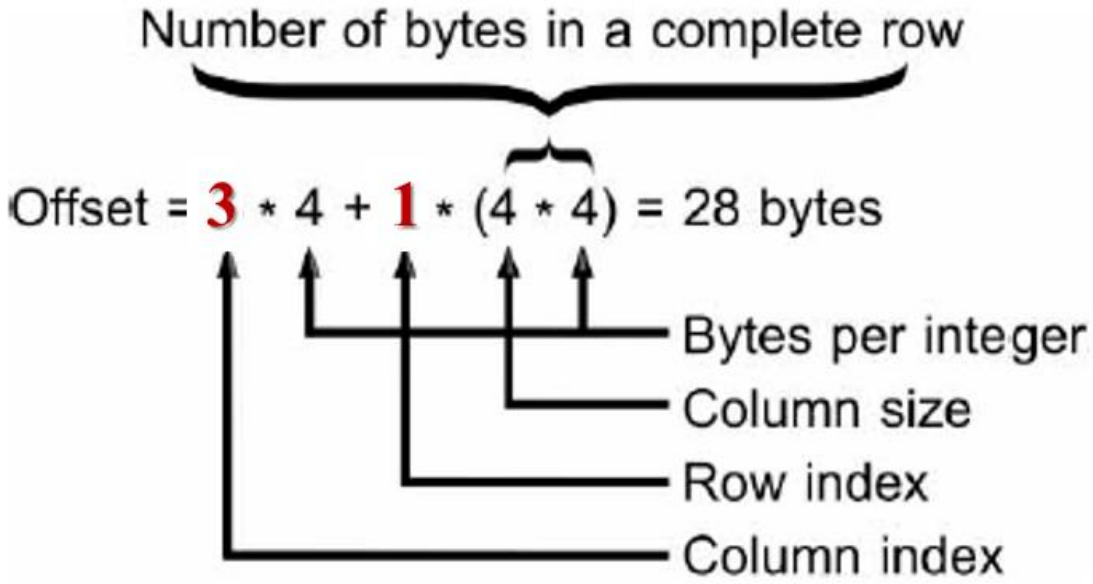

Figure 8.12 Determining an element's offset

8.5 Two-Dimensional Arrays

- Internal Array Element Location Algorithm
  - ***Address element i*** = ***starting array address*** + ***offset***
  - For single-dimensional arrays:
    - *Offset=**i** *ElementSize*
  - For two-dimensional arrays:
    - *Offset=**i***(ColumnSize*elementSize)+**j***ElementSize*

8.5 Two-Dimensional Arrays

Program 8.10 Internal Array Element Location Algorithm

```
1.  #include <stdio.h>
2.  #define NUMELS 20
3.  int main(){
4.    int numbers[NUMELS];
5.          printf("The starting address of the numbers array is: 0x%x\n", &(numbers[0]));
6.          printf("The storage size of each array element is: %d\n",  sizeof(int));
7.          printf("The address of element numbers[5] is : 0x%x\n", &(numbers[5]));
8.          printf("The starting address of the array,\n");
9.          printf("  using the notation numbers, is: 0x%x\n", numbers);
10.         return 0;
11. }
```

```
The starting address of the numbers array is: 0x1efe8c
The storage size of each array element is: 4
The address of element numbers[5] is : 0x1efea0
The starting address of the array,
  using the notation numbers, is: 0x1efe8c
Press any key to continue . . .
```

8.5 Two-Dimensional Arrays

➢ Larger Dimensional Arrays

- A three-dimensional array can be viewed as a book of data tables

  (the third subscript is called the rank)

  - int response[4][10][6];

- A four-dimensional array can be represented as a shelf of books where the fourth dimension is used to declare a desired book on the shelf
- A five-dimensional array can be viewed as a bookcase filled with books where the fifth dimension refers to a selected shelf in the bookcase
- Arrays of three, four, five, six, or more dimensions can be viewed as mathematical n-tuples

8.7 Summary

- A single-dimensional array is a data structure that can store a list of values of the same data type
- Elements are stored in contiguous locations
  - Referenced using the array name and a subscript
- Single-dimensional arrays may be initialized when they are declared
- Single-dimensional arrays are passed to a function by passing the name of the array as an argument
- A two-dimensional array is declared by listing both a row and a column size with the data type and name of the array
- Two-dimensional arrays may be initialized when they are declared
- Two-dimensional arrays are passed to a function by passing the name of the array as an argument

Reference

- https://www.codesdope.com/blog/article/int-main-vs-void-main-vs-int-mainvoid-in-c-c/