C – ARRAY

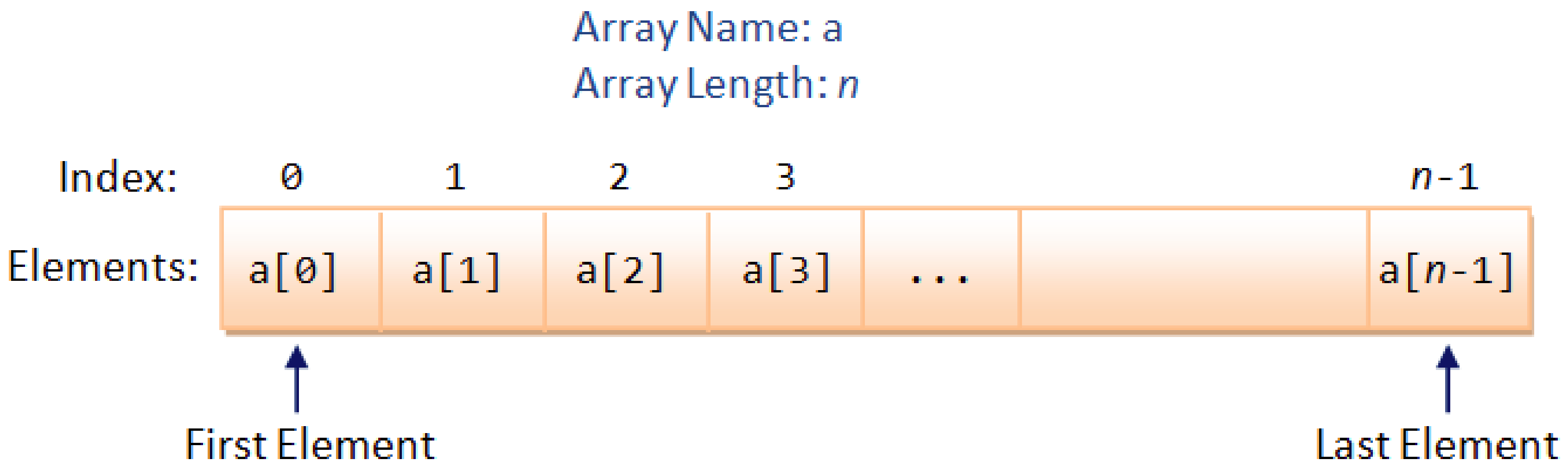
Dr. Sheak Rashed Haider Noori
Associate Professor & Associate Head
Department of Computer Science

C – ARRAY

- C programming language provides a data structure called **the array**, which can store a fixed-size sequential collection of elements of the same type.
- An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
- Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.
- A specific element in an array is accessed by an **index**.

ARRAY

- All arrays consist of contiguous memory locations.
- The lowest address corresponds to the first element and the highest address to the last element.



DECLARING ARRAYS

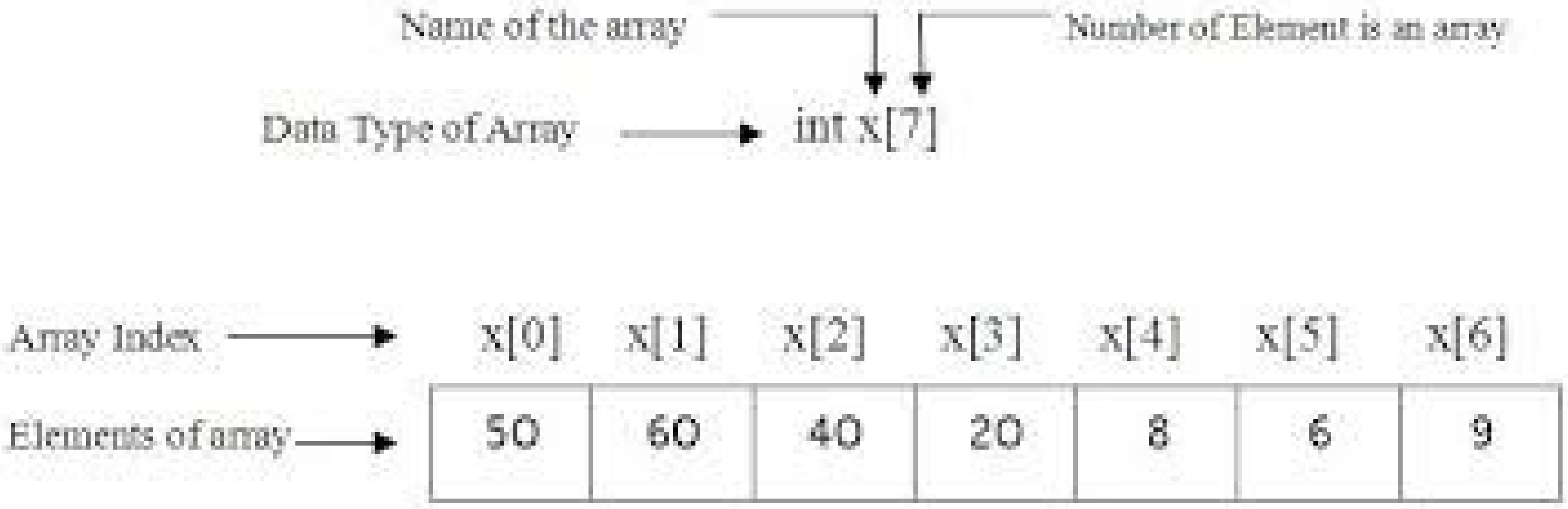
- To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows :

```
type arrayName [ arraySize ];
```

- This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and type can be any valid C data type.
- For example, to declare a 10-element array called balance of type double, use this statement:

```
double balance[10];
```

- Now balance is a variable array which is sufficient to hold up to 10 double numbers.


Name of the array
Number of Element is an array
Data Type of Array
int x[7]
Array Index
x[0]
x[1]
x[2]
x[3]
x[4]
x[5]
x[6]
Elements of array
50
60
40
20
8
6
9

INITIALIZING ARRAYS

- You can initialize array in C either one by one or using a single statement as follows:

```
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

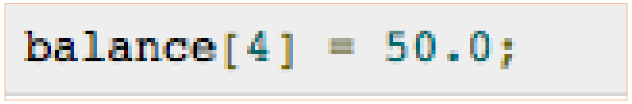
- The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ].
- If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write:

```
double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

- You will create exactly the same array as you did in the previous example.

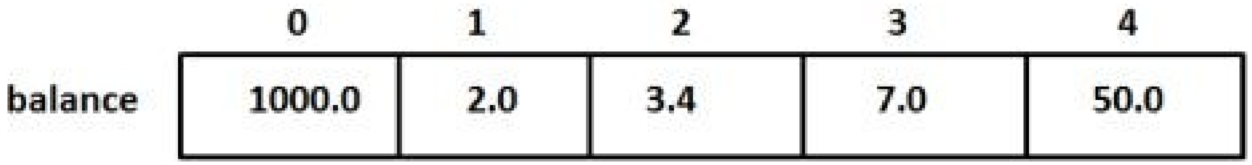
INITIALIZING ARRAYS CONT.

- Following is an example to assign a single element of the array:



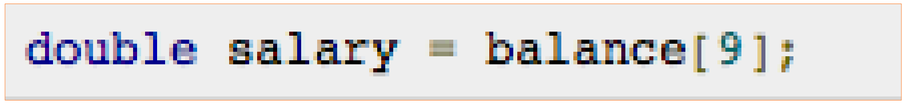
```
balance[4] = 50.0;
```

- The above statement assigns element number 5th in the array with a value of 50.0.
- All arrays have 0 as the index of their first element which is also called base index and last index of an array will be total size of the array minus 1.
- Following is the pictorial representation of the same array we discussed above:



ACCESSING ARRAY ELEMENTS

- An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array.
- For example:

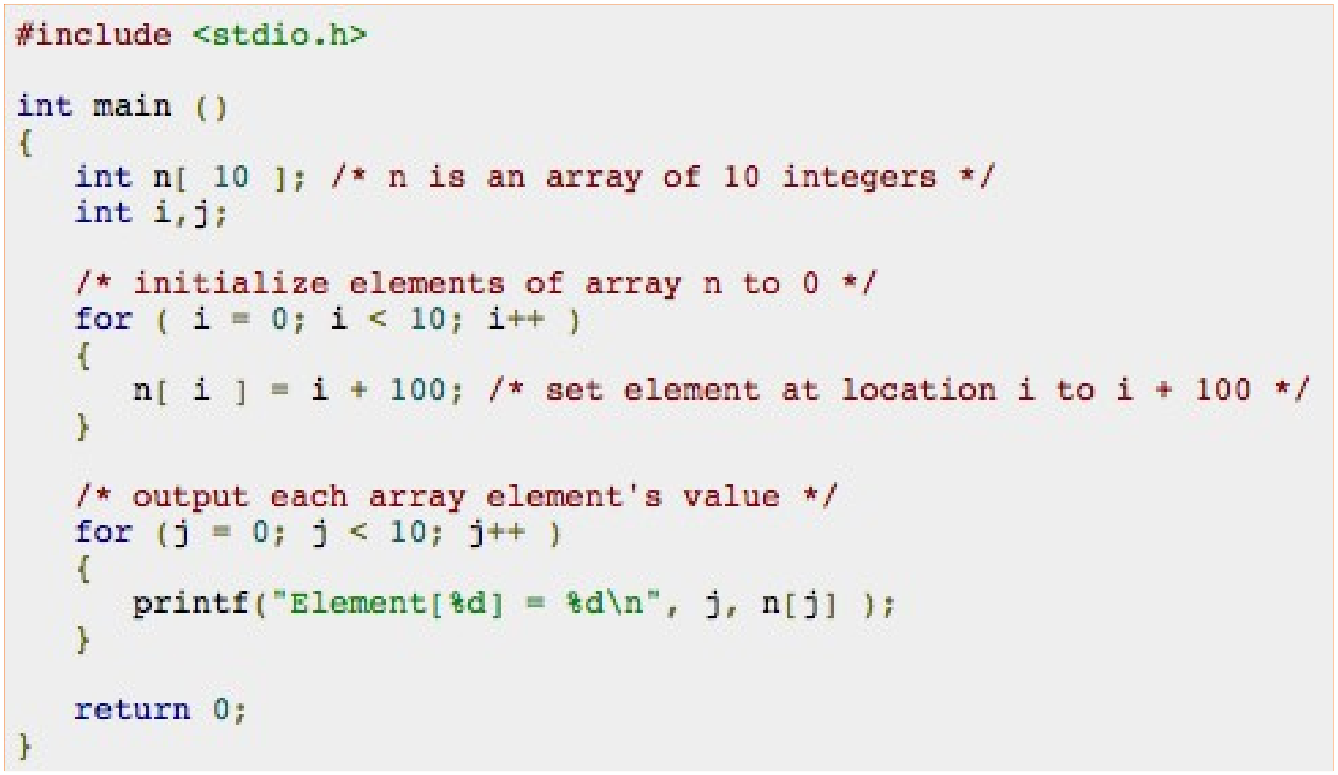


```
double salary = balance[9];
```

- The above statement will take 10th element from the array and assign the value to salary variable.

CODE EXAMPLE: ARRAY

- Following is an example which will use all the above mentioned three concepts declaration, assignment and accessing arrays:



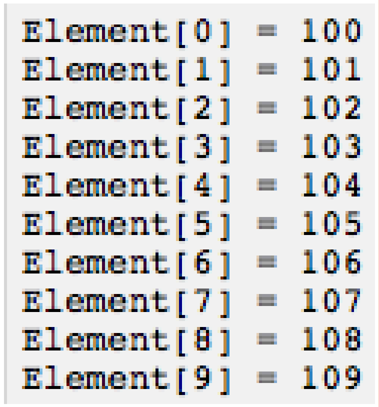
```c
#include <stdio.h>

int main ()
{
   int n[ 10 ]; /* n is an array of 10 integers */
   int i,j;

   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ )
   {
      n[ i ] = i + 100; /* set element at location i to i + 100 */
   }

   /* output each array element's value */
   for (j = 0; j < 10; j++ )
   {
      printf("Element[%d] = %d\n", j, n[j] );
   }

   return 0;
}
```



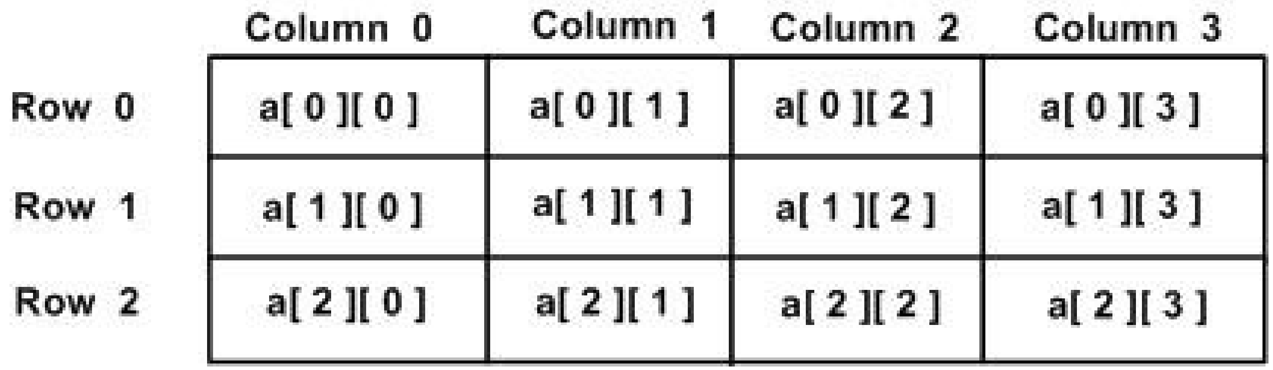
```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

TWO-DIMENSIONAL ARRAYS

- A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size x,y you would write as follows:

```
type arrayName [ x ][ y ];
```

- A two-dimensional array can be think as a table which will have x number of rows and y number of columns.
- A 2-dimensional array **a**, which contains three rows and four columns can be shown as below:

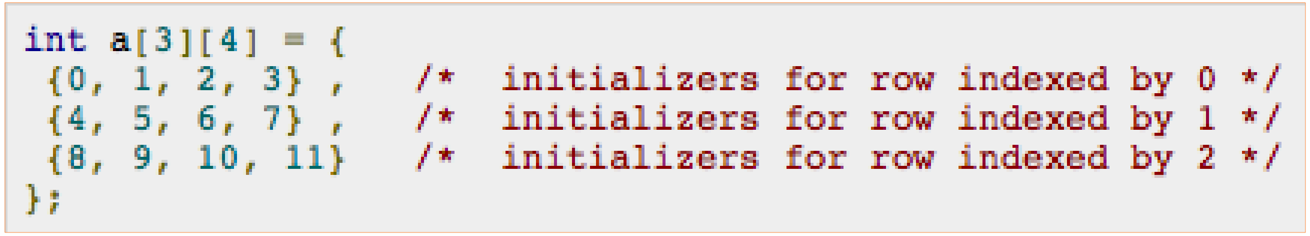


	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

- Thus, every element in array a is identified by an element name of the form **a[i][j]**, where a is the name of the array, and i and j are the subscripts that uniquely identify each element in a.

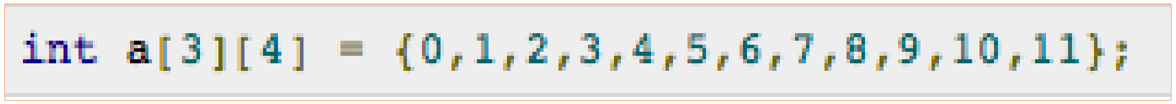
INITIALIZING TWO-DIMENSIONAL ARRAYS:

- Multidimensional arrays may be initialized by specifying bracketed values for each row.
- Following is an array with 3 rows and each row has 4 columns.



```
int a[3][4] = {
 {0, 1, 2, 3} ,   /*  initializers for row indexed by 0 */
 {4, 5, 6, 7} ,   /*  initializers for row indexed by 1 */
 {8, 9, 10, 11}   /*  initializers for row indexed by 2 */
};
```

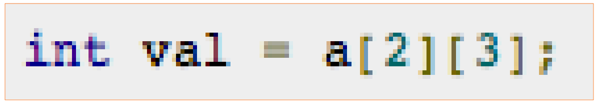
- The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to previous example:



```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

ACCESSING TWO-DIMENSIONAL ARRAY ELEMENTS

- An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example:

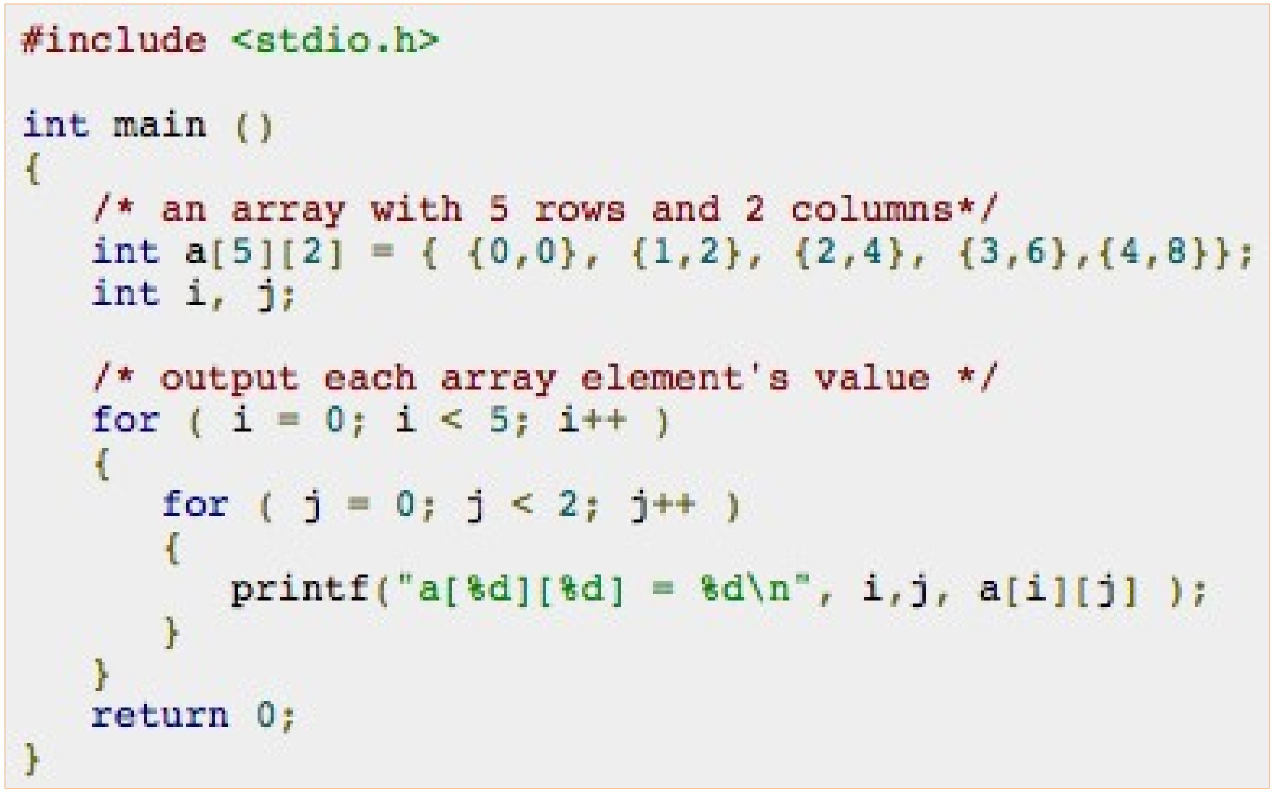


```
int val = a[2][3];
```

- The above statement will take 4th element from the 3rd row of the array. You can verify it in the above diagram.

ACCESSING TWO-DIMENSIONAL ARRAY ELEMENTS

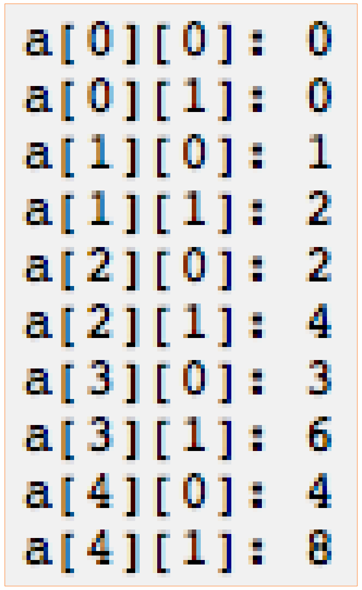
- Let us check below program where we have used nested loop to handle a two dimensional array:



```c
#include <stdio.h>

int main ()
{
   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

   /* output each array element's value */
   for ( i = 0; i < 5; i++ )
   {
      for ( j = 0; j < 2; j++ )
      {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }
   return 0;
}
```



```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

MULTI-DIMENSIONAL ARRAYS IN C

- C programming language allows multidimensional arrays. Here is the general form of a multidimensional array declaration:

```
type name[size1][size2]...[sizeN];
```

- For example, the following declaration creates a three dimensional 5 . 10 . 4 integer array:

```
int threedim[5][10][4];
```

- As explained above, you can have arrays with any number of dimensions, although it is likely that most of the arrays you create will be of one or two dimensions.