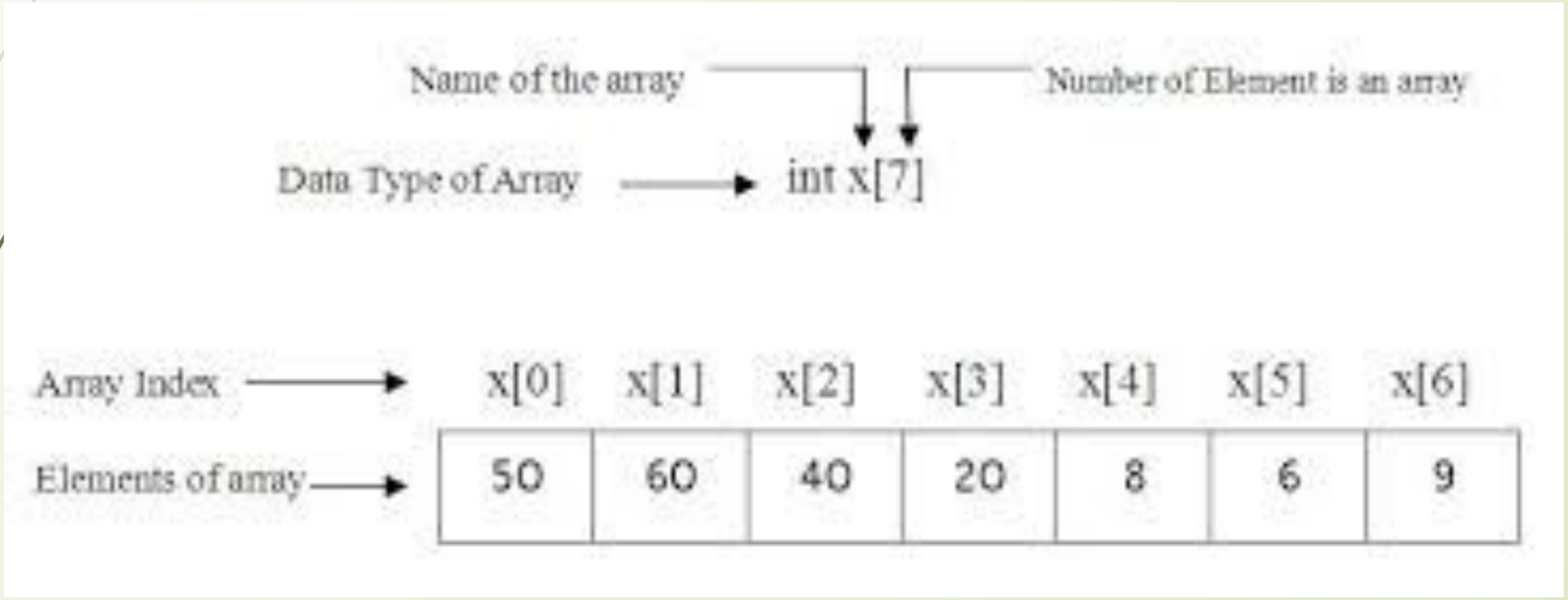
Array

Array is a Data Structure which can store a fixed-size sequential collection of elements of the same data type. An array is used to store a collection of data. A specific element in an array is accessed by an **index**.



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.

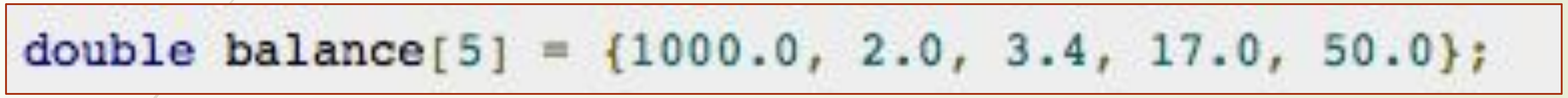
```
type arrayName [ arraySize ];
```

For example, to declare a 10-element array called balance of type double, use this statement.

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

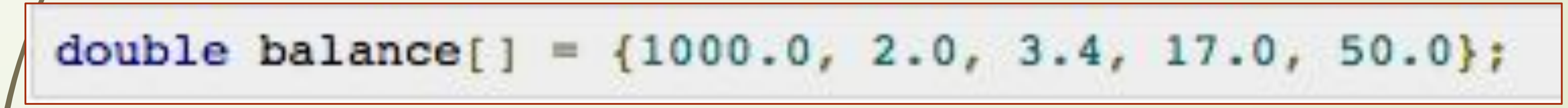
Declaring Arrays

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};
```

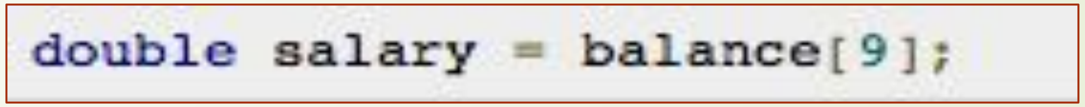
The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ]. If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write:



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

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.



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

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

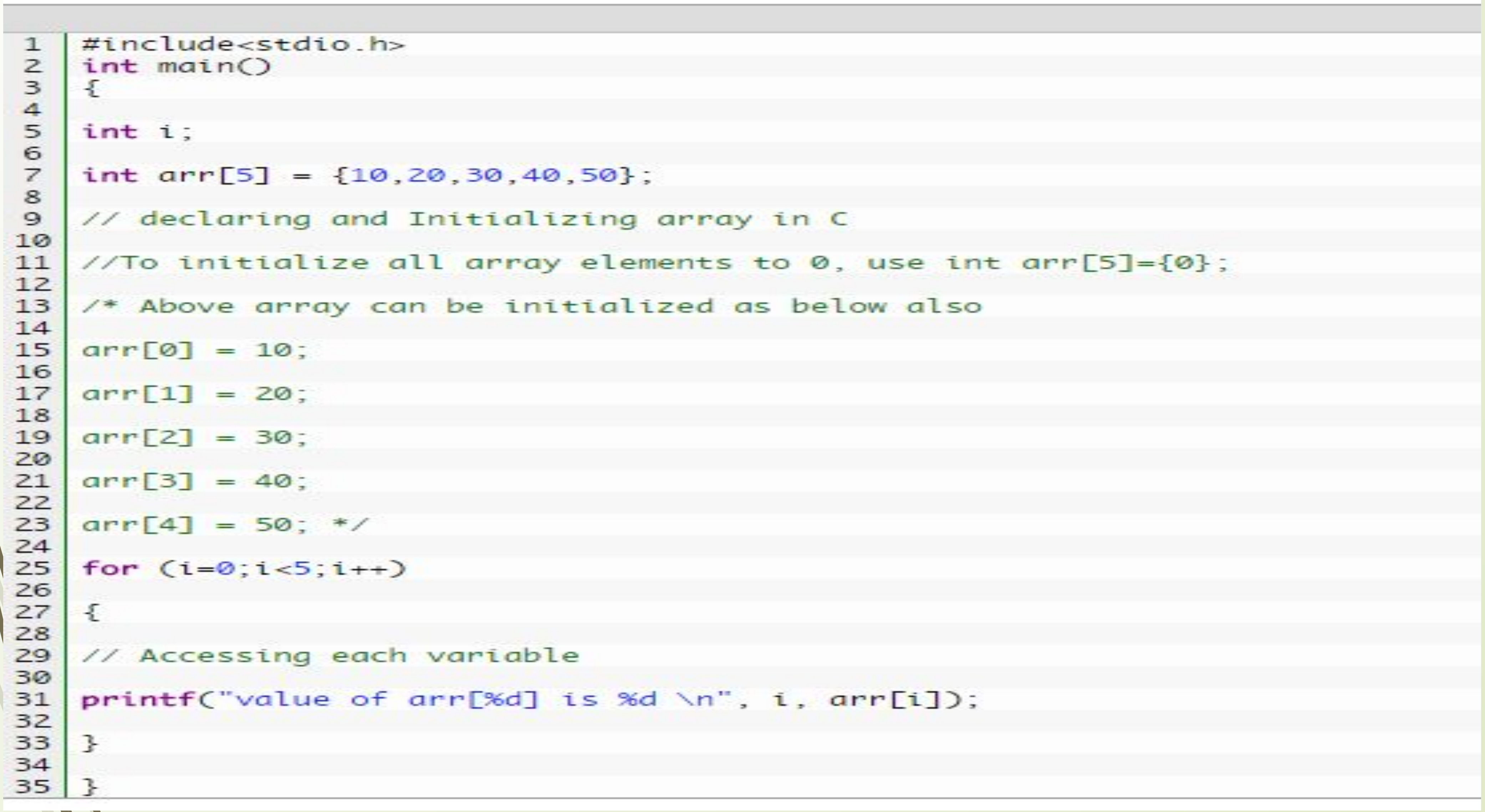
Copying Entire Arrays

We cannot directly copy one array to another, even if they have the same length and share the same data type.

Instead, we can use a for loop to copy values:

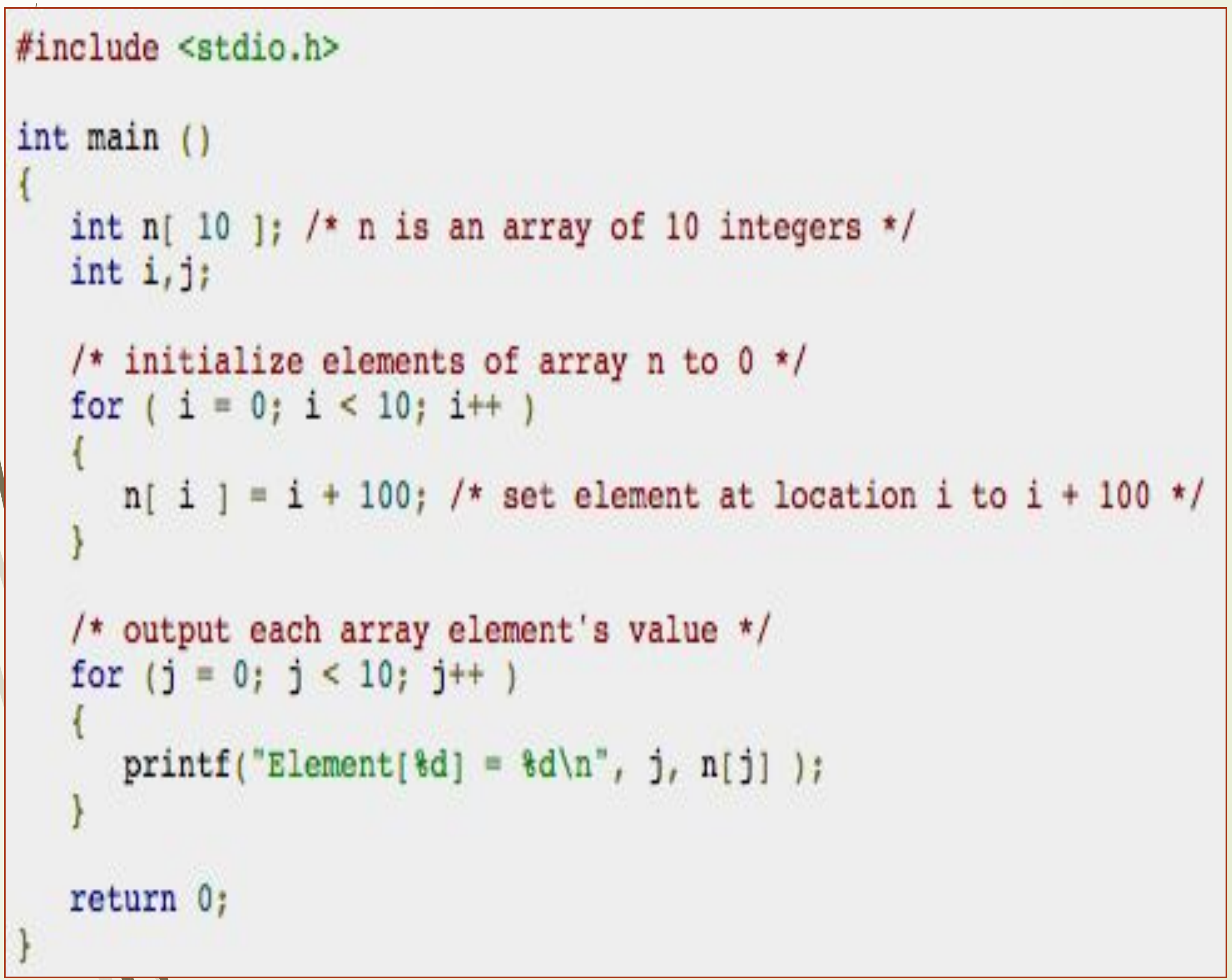
```
for(m = 0; m < 25; m++)
{
    a2[m] = a1[m];
}//end for
```

Declaring Array Example

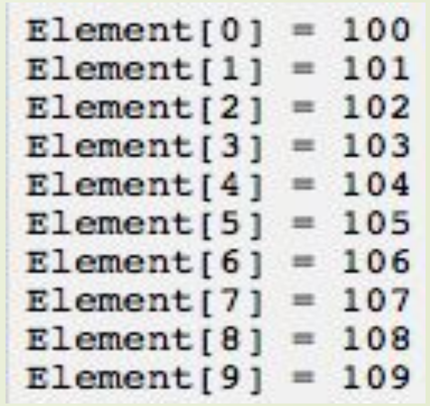


```c
#include<stdio.h>
int main()
{

int i;

int arr[5] = {10,20,30,40,50};

// declaring and Initializing array in C

//To initialize all array elements to 0, use int arr[5]={0};

/* Above array can be initialized as below also

arr[0] = 10;

arr[1] = 20;

arr[2] = 30;

arr[3] = 40;

arr[4] = 50; */

for (i=0;i<5;i++)

{

// Accessing each variable

printf("value of arr[%d] is %d \n", i, arr[i]);

}

}
```

Example:



```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;
}
```



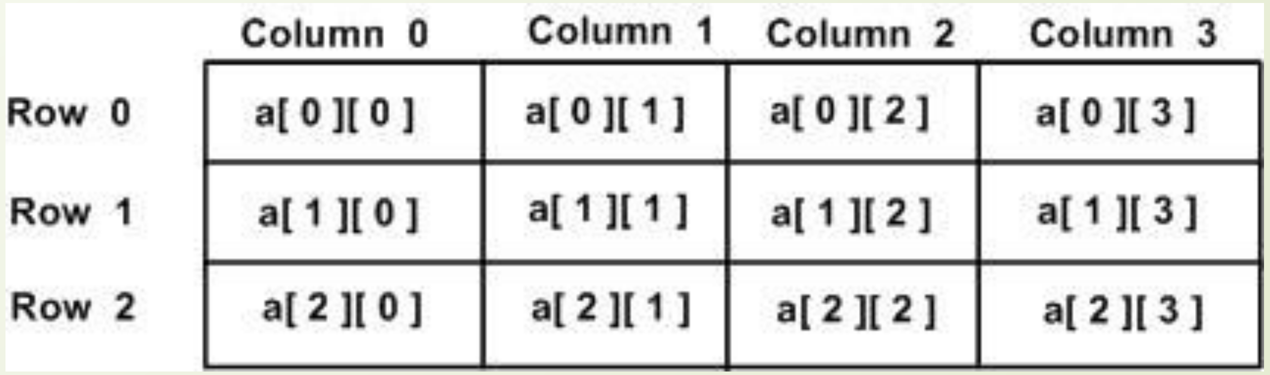
```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

Two-Dimensional Array

A two-dimensional array is 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: a[3][4]



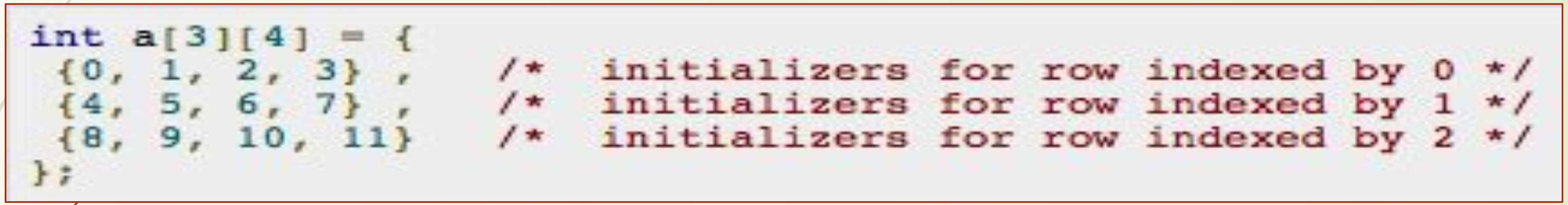
	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

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 Array

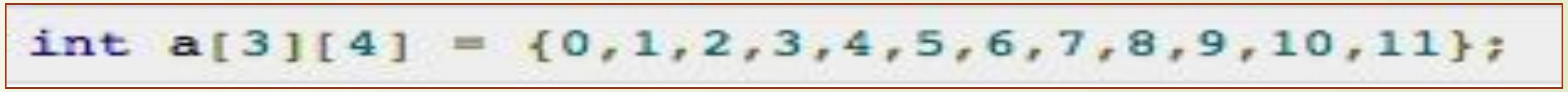
Multidimensional arrays may be initialized by specifying bracketed values for each row.

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 */
};
```

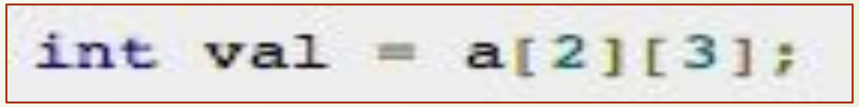
The nested braces, which indicate the row, are optional. The following initialization is equivalent to previous example:



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

Accessing Two-Dimensional Array Element

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

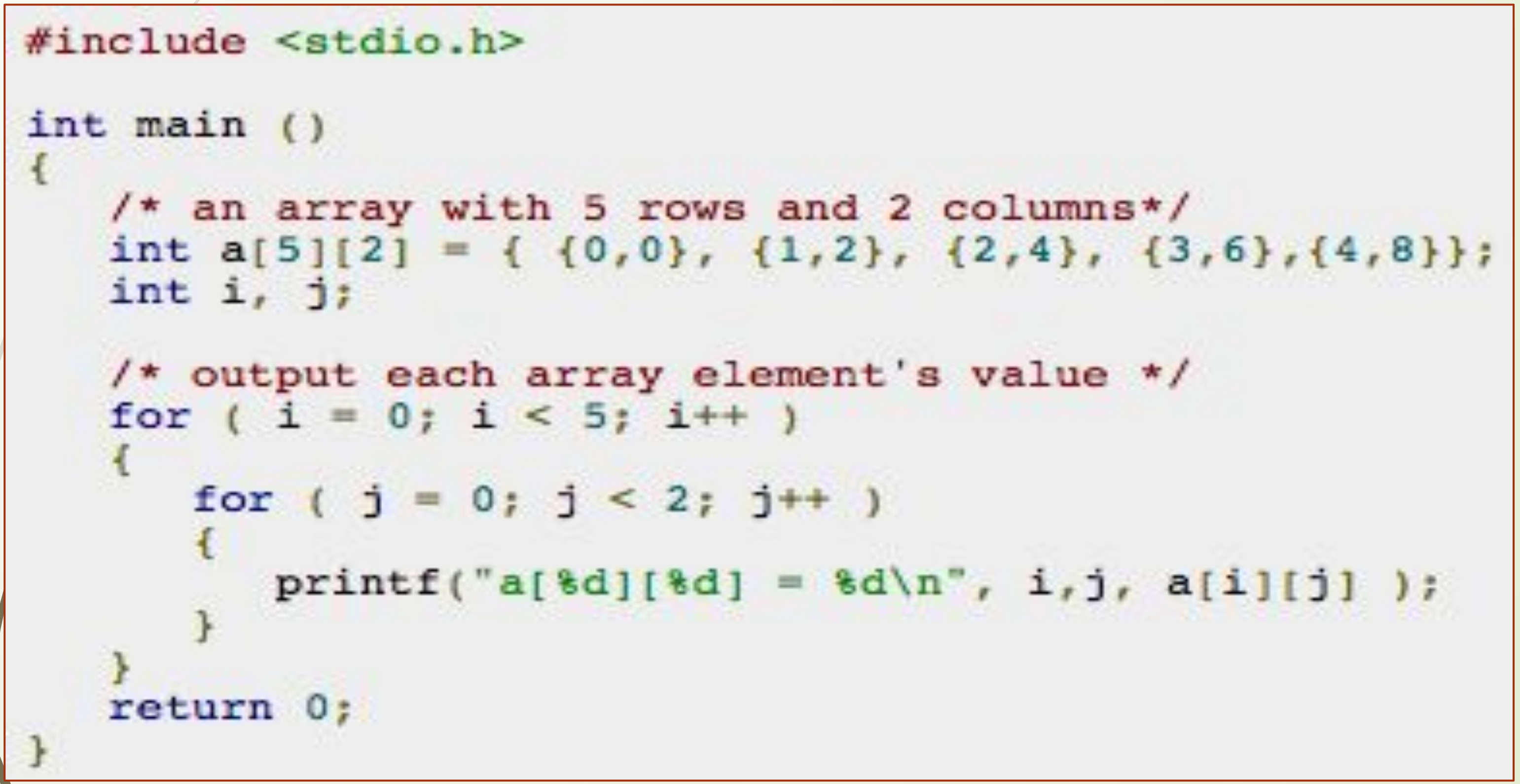


```
int val = a[2][3];
```

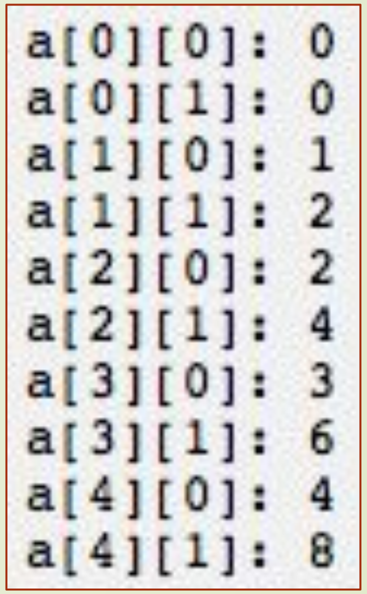
The above statement will take 4th element from the from the 3rd row of the array. You can verify it in the above diagram.

Example

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



```
#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
```

Memory Address Calculation

1. Row Major Order formula

$$X = B + W * [I * C + J]$$

2. Column Major Order formula

$$X = B + W * [I + J * R]$$

Memory Address Calculation

An array S[10][15] is stored in the memory with each element requiring 4 bytes of storage. If the base address of S is 1000, determine the location of S[8][9] when the array is S stored by

(i) Row major (ii) Column major.

Solution:

- ✔ Let us assume that the Base index number is [0][0].
- ✔ Number of Rows = R = 10
- ✔ Number of Columns = C = 15
- ✔ Size of data = W = 4
  - Base address = B = S[0][0] = 1000
  - Location of S[8][9] = X

Memory Address Calculation

(i) When S is stored by Row Major

X = B + W * [8 * C + 9]

= 1000 + 4 * [8 * 15 + 9]

= 1000 + 4 * [120 + 9]

= 1000 + 4 * 129

= 1516

(ii) When S is stored by Column Major

X = B + W * [8 + 9 * R]

= 1000 + 4 * [8 + 9 * 10]

= 1000 + 4 * [8 + 90]

= 1000 + 4 * 98

= 1000 + 392

= 1392

Lab Task:
Write a C program to sort an array

```
#include<stdio.h>
void main ()
{
   int i, j,temp;
    int a[10] = { 10, 9, 7, 101, 23, 44, 12, 78,
34, 23};
   for(i = 0; i<10; i++)
   {
      for(j = i+1; j<10; j++)
      {
if(a[j] > a[i])
            {
               temp = a[i];
               a[i] = a[j];
               a[j] = temp;
            }
         }
   }
     printf("Printing Sorted Element List
...\n");
   for(i = 0; i<10; i++)
   {
      printf("%d\n",a[i]);
   }
}
```

Write a C program to find the largest value from an array

```
#include<stdio.h>
void main ()
{
    int arr[100],i,n,largest,sec_largest;
    printf("Enter the size of the array?");
    scanf("%d",&n);
    printf("Enter the elements of the array?");
    for(i = 0; i<n; i++)
    {
        scanf("%d",&arr[i]);
    }
```

```
largest = arr[0];
    sec_largest = arr[1];
    for(i=0;i<n;i++)
    {
        if(arr[i]>largest)
        {
            sec_largest = largest;
            largest = arr[i];
        }
        else if (arr[i]>sec_largest &&
arr[i]!=largest)
        {
            sec_largest=arr[i];
        }
    }
    printf("largest = %d, second largest =
%d",largest,sec_largest);
```

Write a program to find average marks obtained by a class of 30 students in a test.

```
#include<stdio.h>
int main( )
{
int avg, sum = 0 ;
int i ;
int marks[30] ; /* array declaration */
for ( i = 0 ; i <= 29 ; i++ )
{
printf ( "Enter marks " ) ;
scanf ( "%d", &marks[i] ) ; /* store data in array */
}
for ( i = 0 ; i <= 29 ; i++ )
sum = sum + marks[i] ; /* read data from an array*/
```

Lab Task

1. C program take size 10 array and assign 10 values and print all 10 values using array.
2. C program to take 10 number from user by using loop and store it in array.
3. Print the number stored in array by using loop.
4. C program to find the sum of marks of n students given by user using Arrays

Hints-

1. Use for loop.
2. use array