Arrays a kind of data structure that 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.

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];

Here *balance* is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

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

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

The number of values between braces { } cannot 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, 7.0, 50.0};

You will create exactly the same array as you did in the previous example. Following is an example to assign a single element of the array −

balance[4] = 50.0;

The above statement assigns the 5th element in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1. Shown below is the pictorial representation of the 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 the 10th element from the array and assign the value to salary variable. The following example Shows how to use all the three above mentioned concepts viz. declaration, assignment, and accessing arrays −

#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;

}

When the above code is compiled and executed, it produces the following result −

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

Arrays in Detail

Arrays are important to C and should need a lot more attention. The following important concepts related to array should be clear to a C programmer −

 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 integer array −

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

Two-dimensional Arrays

The simplest form of multidimensional array is the two-dimensional array. 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 something as follows −

type arrayName [ x ][ y ];

Where **type** can be any valid C data type and **arrayName** will be a valid C identifier. A two-dimensional array can be considered as a table which will have x number of rows and y number of columns. A two-dimensional array **a**, which contains three rows and four columns can be shown as follows −



Thus, every element in the 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 the 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 a two-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 the 4th element from the 3rd row of the array. You can verify it in the above figure. Let us check the following program where we have used a 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;

}

When the above code is compiled and executed, it produces the following result −

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

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.







