Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To create an array, define the data type (like int) and specify the name of the array followed by **square brackets []**.

To insert values to it, use a comma-separated list, inside curly braces:

```
int myNumbers[] = {25, 50, 75, 100};
```

We have now created a variable that holds an array of four integers.

Access the Elements of an Array

To access an array element, refer to its **index number**.

Array indexes start with **0**: [0] is the first element. [1] is the second element, etc.

This statement accesses the value of the **first element [0]** in myNumbers:

Example

```
int myNumbers[] = {25, 50, 75, 100};
printf("%d", myNumbers[0]);

// Outputs 25
```

Change an Array Element

To change the value of a specific element,

refer to the index number:

Example

```
myNumbers[0] = 33;
```

Example

```
int myNumbers[] = {25, 50, 75, 100};
myNumbers[0] = 33;

printf("%d", myNumbers[0]);

// Now outputs 33 instead of 25
```

C Arrays

❮ PreviousNext ❯

Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To create an array, define the data type (like int) and specify the name of the array followed by **square brackets []**.

To insert values to it, use a comma-separated list, inside curly braces:

```
int myNumbers[] = {25, 50, 75, 100};
```

We have now created a variable that holds an array of four integers.

Access the Elements of an Array

To access an array element, refer to its **index number**.

Array indexes start with **0**: [0] is the first element. [1] is the second element, etc.

This statement accesses the value of the **first element [0]** in myNumbers:

Example

```
int myNumbers[] = {25, 50, 75, 100};
printf("%d", myNumbers[0]);

// Outputs 25
```

Try it Yourself »

Change an Array Element

To change the value of a specific element, refer to the index number:

Example

```
myNumbers[0] = 33;
```

Example

```
int myNumbers[] = {25, 50, 75, 100};
myNumbers[0] = 33;

printf("%d", myNumbers[0]);

// Now outputs 33 instead of 25
```

Try it Yourself »

Loop Through an Array

You can loop through the array elements with the for loop.

The following example outputs all elements in the myNumbers array:

Example

```
int myNumbers[] = {25, 50, 75, 100};
int i;

for (i = 0; i < 4; i++) {
  printf("%d\n", myNumbers[i]);
}
```

Set Array Size

Another common way to create arrays, is to specify the size of the array, and add

elements later:

Example

```
// Declare an array of four integers:
int myNumbers[4];

// Add elements
myNumbers[0] = 25;
myNumbers[1] = 50;
myNumbers[2] = 75;
myNumbers[3] = 100;
```

Using this method, **you should know the size of the array,** in order for the program to store enough memory.

You are not able to change the size of the array after creation.

Multidimensional Arrays

In the previous chapter, you learned about arrays, which is also known as **single dimension arrays**. These are great, and something you will use a lot while programming in C. However, if you want to store data as a tabular form, like a table with rows and columns, you need to get familiar with **multidimensional arrays**.

A multidimensional array is basically an array of arrays.

Arrays can have any number of dimensions. In this chapter, we will introduce the most common; two-dimensional arrays (2D).

Two-Dimensional Arrays

A 2D array is also known as a matrix (a table of rows and columns).

To create a 2D array of integers, take a look at the following example:

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
```

The first dimension represents the number of rows **[2]**, while the second dimension represents the number of columns **[3]**. The values are placed in row-order, and can be visualized like this:

	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

Access the Elements of a 2D Array

To access an element of a two-dimensional array, you must specify the index number of both the row and column.

This statement accesses the value of the element in the **first row (0)** and **third column (2)** of the **matrix** array.

Example

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };

printf("%d", matrix[0][2]);  // Outputs 2
```

Change Elements in a 2D Array

To change the value of an element, refer to the index number of the element in each of

the dimensions:

The following example will change the value of the element in the **first row (0)** and **first column (0)**:

Example

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
matrix[0][0] = 9;

printf("%d", matrix[0][0]);  // Now outputs 9 instead of 1
```

Loop Through a 2D Array

To loop through a multi-dimensional array, you need one loop for each of the array's dimensions.

The following example outputs all elements in the **matrix** array:

Example

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };

int i, j;
for (i = 0; i < 2; i++) {
  for (j = 0; j < 3; j++) {
    printf("%d\n", matrix[i][j]);
  }
}
```