**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, and make sure all values are of the same data type:

int myNumbers[] = {25, 50, 75, 100};

**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:

#include <stdio.h>

int main() {

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

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

return 0;

}

**Change an Array Element**

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

#include <stdio.h>

int main() {

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

myNumbers[0] = 33;

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

return 0;

}

**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:

#include <stdio.h>

int main() {

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

int i;

for (i = 0; i < 4; i++) {

printf("%d\n", myNumbers[i]);

}

return 0;

}

**Set Array Size**

#include <stdio.h>

int main() {

// Declare an array of four integers:

int myNumbers[4];

// Add elements to it

myNumbers[0] = 25;

myNumbers[1] = 50;

myNumbers[2] = 75;

myNumbers[3] = 100;

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

return 0;

}

**Get Array Size or Length**

To get the size of an array, you can use the sizeof operator:

#include <stdio.h>

int main() {

int myNumbers[] = {10, 25, 50, 75, 100};

printf("%lu", sizeof(myNumbers));

return 0;

}

o/p

20

You learned from the [Data Types chapter](https://www.w3schools.com/c/c_data_types.php) that an int type is usually 4 bytes, so from the example above, 4 x 5 (*4 bytes x 5 elements*) = **20 bytes**.

**To find out specific length**

#include <stdio.h>

int main() {

int myNumbers[] = {10, 25, 50, 75, 100};

int length = sizeof(myNumbers) / sizeof(myNumbers[0]);

printf("%d", length);

return 0;

} o/p 5

**Real life example of Array**

To demonstrate a practical example of using arrays, let's create a program that calculates the average of different marks:

#include <stdio.h>

int main() {

// An array storing different ages

int ages[] = {20, 22, 18, 35, 48, 26, 87, 70};

float avg, sum = 0;

int i;

// Get the length of the array

int length = sizeof(ages) / sizeof(ages[0]);

// Loop through the elements of the array and accumulate the sum

for (i = 0; i < length; i++) {

sum += ages[i];

}

// Calculate the average by dividing the sum by the length

avg = sum / length;

// Print the average

printf("The average age is: %.2f", avg);

return 0;

}

**o/p**

The average age is: 40.75

**Multidimensional Arrays**

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).

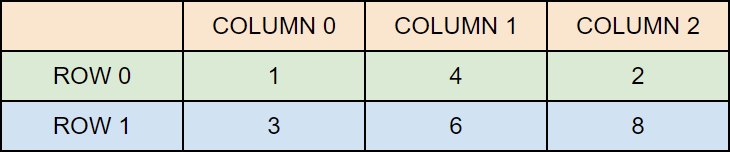
**Two-Dimensional Arrays**

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:



**Access the Elements of a 2D Array**

#include <stdio.h>

int main() {

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

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

return 0;

}

**Loop Through a 2D Array**

#include <stdio.h>

int main() {

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 ", matrix[i][j]);

}

printf("\n");

}

return 0;

}