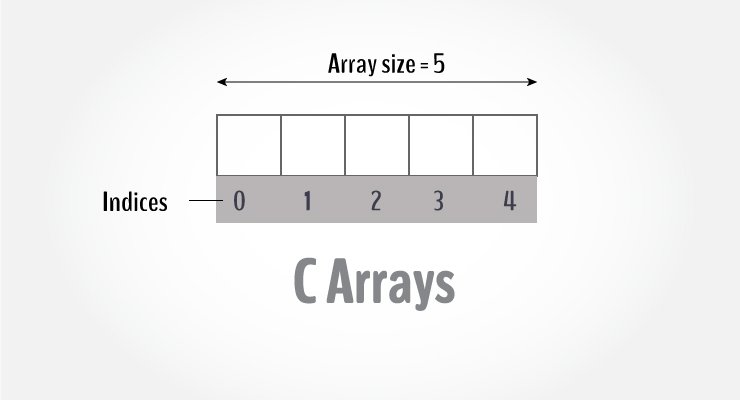
**C Arrays**

In this tutorial, you will learn to work with arrays. You will learn to declare, initialize and access elements of an array with the help of examples.

Arrays in C

An array is a variable that can store multiple values. For example, if you want to store 100 integers, you can create an array for it.

int data[100];

**How to declare an array?**

dataType arrayName[arraySize];

**For example,**

float mark[5];

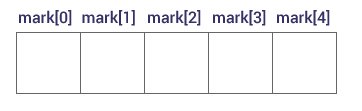
Here, we declared an array, mark, of floating-point type. And its size is 5. Meaning, it can hold 5 floating-point values.

It's important to note that the size and type of an array cannot be changed once it is declared.

**Access Array Elements**

You can access elements of an array by indices.

Suppose you declared an array mark as above. The first element is mark[0], the second element is mark[1] and so on.

Declare an Array

**Few keynotes**:

* Arrays have 0 as the first index, not 1. In this example, mark[0] is the first element.
* If the size of an array is n, to access the last element, the n-1 index is used. In this example, mark[4]
* Suppose the starting address of mark[0] is **2120d**. Then, the address of the mark[1] will be **2124d**. Similarly, the address of mark[2] will be **2128d** and so on.  
  This is because the size of a float is 4 bytes.

**How to initialize an array?**

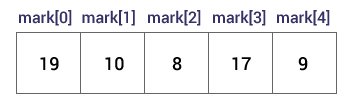
It is possible to initialize an array during declaration. For example,

int mark[5] = {19, 10, 8, 17, 9};

You can also initialize an array like this.

int mark[] = {19, 10, 8, 17, 9};

Here, we haven't specified the size. However, the compiler knows its size is 5 as we are initializing it with 5 elements.

Initialize an Array

Here,

mark[0] is equal to 19

mark[1] is equal to 10

mark[2] is equal to 8

mark[3] is equal to 17

mark[4] is equal to 9

**Change Value of Array elements**

int mark[5] = {19, 10, 8, 17, 9}

// make the value of the third element to -1

mark[2] = -1;

// make the value of the fifth element to 0

mark[4] = 0;

**Input and Output Array Elements**

Here's how you can take input from the user and store it in an array element.

// take input and store it in the 3rd element

​scanf("%d", &mark[2]);

// take input and store it in the ith element

scanf("%d", &mark[i-1]);

Here's how you can print an individual element of an array.

// print the first element of the array

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

// print the third element of the array

printf("%d", mark[2]);

// print ith element of the array

printf("%d", mark[i-1]);

**Example 1: Array Input/Output**

// Program to take 5 values from the user and store them in an array

// Print the elements stored in the array

#include <stdio.h>

int main() {

int values[5];

printf("Enter 5 integers: ");

// taking input and storing it in an array

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

scanf("%d", &values[i]);

}

printf("Displaying integers: ");

// printing elements of an array

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

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

}

return 0;

}

**Output**

Enter 5 integers: 1

-3

34

0

3

Displaying integers: 1

-3

34

0

3

Here, we have used a for loop to take 5 inputs from the user and store them in an array. Then, using another for loop, these elements are displayed on the screen.

## Example 2: Calculate Average

// Program to find the average of n numbers using arrays

#include <stdio.h>

int main() {

int marks[10], i, n, sum = 0, average;

printf("Enter number of elements: ");

scanf("%d", &n);

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

printf("Enter number%d: ",i+1);

scanf("%d", &marks[i]);

// adding integers entered by the user to the sum variable

sum += marks[i];

}

average = sum / n;

printf("Average = %d", average);

return 0;

}

**Output**

Enter n: 5

Enter number1: 45

Enter number2: 35

Enter number3: 38

Enter number4: 31

Enter number5: 49

Average = 39

Here, we have computed the average of n numbers entered by the user.

### Access elements out of its bound!

Suppose you declared an array of 10 elements. Let's say,

int testArray[10];

You can access the array elements from testArray[0] to testArray[9].

Now let's say if you try to access testArray[12]. The element is not available. This may cause unexpected output (undefined behavior). Sometimes you might get an error and some other time your program may run correctly.

Hence, you should never access elements of an array outside of its bound.

## Multidimensional arrays

In this tutorial, you learned about arrays. These arrays are called one-dimensional arrays.

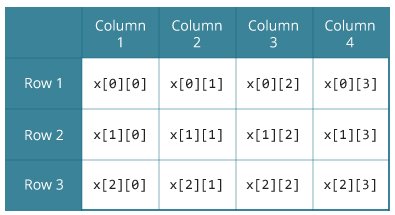
# C Multidimensional Arrays

In this tutorial, you will learn to work with multidimensional arrays (two-dimensional and three-dimensional arrays) with the help of examples.

In C programming, you can create an array of arrays. These arrays are known as multidimensional arrays. For example,

float x[3][4];

Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as a table with 3 rows and each row has 4 columns.

Two dimensional Array

Similarly, you can declare a three-dimensional (3d) array. For example,

float y[2][4][3];

Here, the array y can hold 24 elements.

## Initializing a multidimensional array

Here is how you can initialize two-dimensional and three-dimensional arrays:

### Initialization of a 2d array

// Different ways to initialize two-dimensional array

int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[][3] = {{1, 3, 0}, {-1, 5, 9}};

int c[2][3] = {1, 3, 0, -1, 5, 9};

### Initialization of a 3d array

You can initialize a three-dimensional array in a similar way like a two-dimensional array. Here's an example,

int test[2][3][4] = {

{{3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2}},

{{13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9}}};

### Example 1: Two-dimensional array to store and print values

// C program to store temperature of two cities of a week and display it.

#include <stdio.h>

const int CITY = 2;

const int WEEK = 7;

int main()

{

int temperature[CITY][WEEK];

// Using nested loop to store values in a 2d array

for (int i = 0; i < CITY; ++i)

{

for (int j = 0; j < WEEK; ++j)

{

printf("City %d, Day %d: ", i + 1, j + 1);

scanf("%d", &temperature[i][j]);

}

}

printf("\nDisplaying values: \n\n");

// Using nested loop to display vlues of a 2d array

for (int i = 0; i < CITY; ++i)

{

for (int j = 0; j < WEEK; ++j)

{

printf("City %d, Day %d = %d\n", i + 1, j + 1, temperature[i][j]);

}

}

return 0;

}

**Output**

City 1, Day 1: 33

City 1, Day 2: 34

City 1, Day 3: 35

City 1, Day 4: 33

City 1, Day 5: 32

City 1, Day 6: 31

City 1, Day 7: 30

City 2, Day 1: 23

City 2, Day 2: 22

City 2, Day 3: 21

City 2, Day 4: 24

City 2, Day 5: 22

City 2, Day 6: 25

City 2, Day 7: 26

Displaying values:

City 1, Day 1 = 33

City 1, Day 2 = 34

City 1, Day 3 = 35

City 1, Day 4 = 33

City 1, Day 5 = 32

City 1, Day 6 = 31

City 1, Day 7 = 30

City 2, Day 1 = 23

City 2, Day 2 = 22

City 2, Day 3 = 21

City 2, Day 4 = 24

City 2, Day 5 = 22

City 2, Day 6 = 25

City 2, Day 7 = 26

### Example 2: Sum of two matrices

// C program to find the sum of two matrices of order 2\*2

#include <stdio.h>

int main()

{

float a[2][2], b[2][2], result[2][2];

// Taking input using nested for loop

printf("Enter elements of 1st matrix\n");

for (int i = 0; i < 2; ++i)

for (int j = 0; j < 2; ++j)

{

printf("Enter a%d%d: ", i + 1, j + 1);

scanf("%f", &a[i][j]);

}

// Taking input using nested for loop

printf("Enter elements of 2nd matrix\n");

for (int i = 0; i < 2; ++i)

for (int j = 0; j < 2; ++j)

{

printf("Enter b%d%d: ", i + 1, j + 1);

scanf("%f", &b[i][j]);

}

// adding corresponding elements of two arrays

for (int i = 0; i < 2; ++i)

for (int j = 0; j < 2; ++j)

{

result[i][j] = a[i][j] + b[i][j];

}

// Displaying the sum

printf("\nSum Of Matrix:");

for (int i = 0; i < 2; ++i)

for (int j = 0; j < 2; ++j)

{

printf("%.1f\t", result[i][j]);

if (j == 1)

printf("\n");

}

return 0;

}

**Output**

Enter elements of 1st matrix

Enter a11: 2;

Enter a12: 0.5;

Enter a21: -1.1;

Enter a22: 2;

Enter elements of 2nd matrix

Enter b11: 0.2;

Enter b12: 0;

Enter b21: 0.23;

Enter b22: 23;

Sum Of Matrix:

2.2 0.5

-0.9 25.0

### Example 3: Three-dimensional array

// C Program to store and print 12 values entered by the user

#include <stdio.h>

int main()

{

int test[2][3][2];

printf("Enter 12 values: \n");

for (int i = 0; i < 2; ++i)

{

for (int j = 0; j < 3; ++j)

{

for (int k = 0; k < 2; ++k)

{

scanf("%d", &test[i][j][k]);

}

}

}

// Printing values with proper index.

printf("\nDisplaying values:\n");

for (int i = 0; i < 2; ++i)

{

for (int j = 0; j < 3; ++j)

{

for (int k = 0; k < 2; ++k)

{

printf("test[%d][%d][%d] = %d\n", i, j, k, test[i][j][k]);

}

}

}

return 0;

}

**Output**

Enter 12 values:

1

2

3

4

5

6

7

8

9

10

11

12

Displaying Values:

test[0][0][0] = 1

test[0][0][1] = 2

test[0][1][0] = 3

test[0][1][1] = 4

test[0][2][0] = 5

test[0][2][1] = 6

test[1][0][0] = 7

test[1][0][1] = 8

test[1][1][0] = 9

test[1][1][1] = 10

test[1][2][0] = 11

test[1][2][1] = 12

# Pass arrays to a function in C

In this tutorial, you'll learn to pass arrays (both one-dimensional and multidimensional arrays) to a function in C programming with the help of examples.

In C programming, you can pass an entire array to functions. Before we learn that, let's see how you can pass individual elements of an array to functions.

## Pass Individual Array Elements

Passing array elements to a function is similar to [passing variables to a function](https://www.programiz.com/c-programming/c-user-defined-functions).

### Example 1: Pass Individual Array Elements

#include <stdio.h>

void display(int age1, int age2) {

printf("%d\n", age1);

printf("%d\n", age2);

}

int main() {

int ageArray[] = {2, 8, 4, 12};

// pass second and third elements to display()

display(ageArray[1], ageArray[2]);

return 0;

}

**Output**

8

4

Here, we have passed array parameters to the display() function in the same way we pass variables to a function.

// pass second and third elements to display()

display(ageArray[1], ageArray[2]);

We can see this in the function definition, where the function parameters are individual variables:

void display(int age1, int age2) {

// code

}

## Example 2: Pass Arrays to Functions

// Program to calculate the sum of array elements by passing to a function

#include <stdio.h>

float calculateSum(float num[]);

int main() {

float result, num[] = {23.4, 55, 22.6, 3, 40.5, 18};

// num array is passed to calculateSum()

result = calculateSum(num);

printf("Result = %.2f", result);

return 0;

}

float calculateSum(float num[]) {

float sum = 0.0;

for (int i = 0; i < 6; ++i) {

sum += num[i];

}

return sum;

}

**Output**

Result = 162.50

To pass an entire array to a function, only the name of the array is passed as an argument.

result = calculateSum(num);

However, notice the use of [] in the function definition.

float calculateSum(float num[]) {

... ..

}

This informs the compiler that you are passing a one-dimensional array to the function.

## Pass Multidimensional Arrays to a Function

To pass multidimensional arrays to a function, only the name of the array is passed to the function (similar to one-dimensional arrays).

### Example 3: Pass two-dimensional arrays

#include <stdio.h>

void displayNumbers(int num[2][2]);

int main() {

int num[2][2];

printf("Enter 4 numbers:\n");

for (int i = 0; i < 2; ++i) {

for (int j = 0; j < 2; ++j) {

scanf("%d", &num[i][j]);

}

}

// pass multi-dimensional array to a function

displayNumbers(num);

return 0;

}

void displayNumbers(int num[2][2]) {

printf("Displaying:\n");

for (int i = 0; i < 2; ++i) {

for (int j = 0; j < 2; ++j) {

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

}

}

}

**Output**

Enter 4 numbers:

2

3

4

5

Displaying:

2

3

4

5

Notice the parameter int num[2][2] in the function prototype and function definition:

// function prototype

void displayNumbers(int num[2][2]);

This signifies that the function takes a two-dimensional array as an argument. We can also pass arrays with more than 2 dimensions as a function argument.

When passing two-dimensional arrays, it is not mandatory to specify the number of rows in the array. However, the number of columns should always be specified.

For example,

void displayNumbers(int num[][2]) {

// code

}

**Note:** In C programming, you can pass arrays to functions, however, you cannot return arrays from functions.