**UNIT III**

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.

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.



**Example**

#include <stdio.h>

int main () {

   int a[5];

   int i,j;

   for (i=0;i<5;i++) {

      a[i] = i+100;

   }

   for (j=0;j<5;j++) {

      printf("Element[%d] = %d
", j, a[j] );

   }

   return 0;

}

**Output**

Element[0] = 100

Element[1] = 101

Element[2] = 102

Element[3] = 103

Element[4] = 104

## Types of arrays

Arrays are broadly classified into three types. They are as follows −

* One – dimensional arrays
* Two – dimensional arrays
* Multi – dimensional arrays

### One – dimensional array

The Syntax is as follows −

datatype array name [size]

For example, int a[5]

**Initialization**

An array can be initialized in two ways, which are as follows −

* Compile time initialization
* Runtime initialization

## Example

Following is the C program on compile time initialization −

#include<stdio.h>

int main ( ){

   int a[5] = {10,20,30,40,50};

   int i;

   printf ("elements of the array are");

   for ( i=0; i<5; i++)

      printf ("%d", a[i]);

}

## Output

Upon execution, you will receive the following output −

Elements of the array are

10 20 30 40 50

**Declaration of One-dimensional Arrays**

**Syntax**

The syntax is as follows for declaring an array −

datatype array\_name [size];

### One – dimensional array

The Syntax is as follows −

datatype array name [size]

For example, int a[5]

**Initialization**

An array can be initialized in two ways, which are as follows −

* Compile time initialization
* Runtime initialization

## Example

Following is the C program on compile time initialization −

[Live Demo](http://tpcg.io/Esq4K8oc)

#include<stdio.h>

int main ( ){

   int a[5] = {10,20,30,40,50};

   int i;

   printf ("elements of the array are");

   for ( i=0; i<5; i++)

      printf ("%d", a[i]);

}

## Output

Upon execution, you will receive the following output −

Elements of the array are

10 20 30 40 50

## Example

Following is the C program on **runtime initialization** −

[Live Demo](http://tpcg.io/VFvMhMej)

#include<stdio.h>

main ( ){

   int a[5],i;

   printf ("enter 5 elements");

   for ( i=0; i<5; i++)

      scanf("%d", &a[i]);

   printf("elements of the array are");

   for (i=0; i<5; i++)

      printf("%d", a[i]);

}

## Output

The output is as follows −

enter 5 elements 10 20 30 40 50

elements of the array are : 10 20 30 40 50

# Two Dimensional Array in C

The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns.

## Declaration of two dimensional Array in C

The syntax to declare the 2D array is given below.

1. data\_type array\_name[rows][columns];

Consider the following example.

1. **int** twodimen[4][3];

Here, 4 is the number of rows, and 3 is the number of columns.

Initialization of 2D Array in C

In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously. However, this will not work with 2D arrays. We will have to define at least the second dimension of the array. The two-dimensional array can be declared and defined in the following way.

1. **int** arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};

Two-dimensional array example in C

#include<stdio.h>

**int** main(){

**int** i=0,j=0;

**int** arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};

//traversing 2D array

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

 **for**(j=0;j<3;j++){

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

 }

}

**return** 0;

}

**Output**

arr[0][0] = 1

arr[0][1] = 2

arr[0][2] = 3

arr[1][0] = 2

arr[1][1] = 3

arr[1][2] = 4

arr[2][0] = 3

arr[2][1] = 4

arr[2][2] = 5

arr[3][0] = 4

arr[3][1] = 5

arr[3][2] = 6

**Multi-dimensional array**

A multi-dimensional array can be termed as an array of arrays that stores homogeneous data in tabular form. Data in multidimensional arrays are stored in row-major order.

The***general form of declaring N-dimensional arrays*** is:

data\_type array\_name[size1][size2]....[sizeN];

* **data\_type**: Type of data to be stored in the array.
* **array\_name**: Name of the array
* **size1, size2,… ,sizeN**: Sizes of the dimension

**Examples**:

**Two dimensional array:** int two\_d[10][20];

**Three dimensional array:** int three\_d[10][20][30];

**Syntax:**

**data\_type array\_name[x][y];**

Here,**data\_type**is the type of data to be stored.

We can declare a two-dimensional integer array say ‘x’ of size 10,20 as:

int x[10][20];



Example

#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;

}

#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 –

**Output**

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

## C program to print the sum of all elements in an array

1. #include <stdio.h>
2. **int** main()
3. {
4. //Initialize array
5. **int** arr[] = {1, 2, 3, 4, 5};
6. **int** sum = 0;
7.
8. //Calculate length of array arr
9. **int** length = **sizeof**(arr)/**sizeof**(arr[0]);
10.
11. //Loop through the array to calculate sum of elements
12. **for** (**int** i = 0; i < length; i++) {
13. sum = sum + arr[i];
14. }
15. printf("Sum of all the elements of an array: %d", sum);
16. **return** 0;
17. }

**Output:**

Sum of all the elements of an array: 15

**2. Write a C Program to sort array in ascending order**

#include <stdio.h>

void main (){

   int num[20];

   int i, j, a, n;

   printf("enter number of elements in an array
");

   scanf("%d", &n);

   printf("Enter the elements
");

   for (i = 0; i < n; ++i)

      scanf("%d", &num[i]);

   for (i = 0; i < n; ++i){

      for (j = i + 1; j < n; ++j){

         if (num[i] > num[j]){

            a = num[i];

            num[i] = num[j];

            num[j] = a;

         }

      }

   }

   printf("The numbers in ascending order is:
");

   for (i = 0; i < n; ++i){

      printf("%d
", num[i]);

   }

}

## Output

When the above program is executed, it produces the following result −

enter number of elements in an array

5

Enter the elements

12

23

89

11

22

The numbers in ascending order is:

11

12

22

23

89