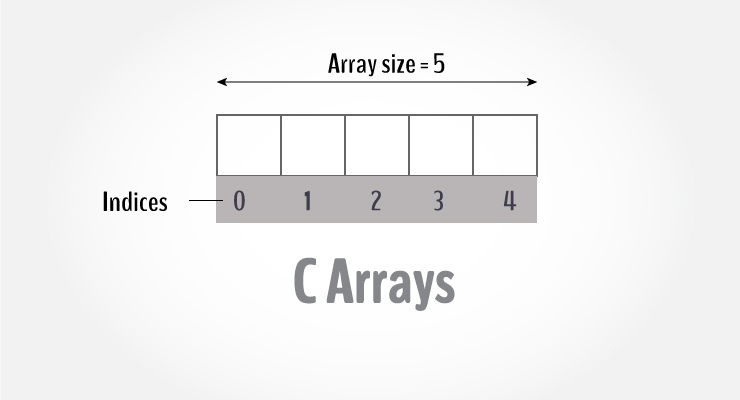
Content 16

Arrays In C

**Arrays :**

“An array is a collection of data items of the same data type. And it is also known as subscript variable”.

* Items are stored at contiguous memory locations in Arrays.
* It can also store the collection of derived data types such as pointers, structures etc.
* The C Language places no limits on the number of dimensions in an array i.e. we can create any no. of dimension array E.g. 2d array, 3d array etc.



**Most Commonly used dimensions of Array :**

* A one-dimensional array is like a list.
* A two-dimensional array is like a table.(like an matrices which could have n number of rows and column)

Some texts refer to **one-dimensional arrays as vectors** and **two-dimensional arrays as matrices** and use the general term **Arrays** when the no. of dimensions is unspecified or unimportant.

#### Advantage of Arrays :

* It is used to represent multiple data items of same type by using only single name.
* Accessing an item in a given array is very fast.

#### Properties of Array :

* Data in an array is stored in contiguous memory locations in RAM.
* Each element of an array is of same size i.e. their data types are same so memory consumed by each is also same.
* Any element of the array with given index can be accessed very quickly by using its address which can be calculated using the base address and the index.

## Disadvantages

* It allows us to enter only fixed number of elements into it. We cannot alter the size of the array once array is declared. Hence if we need to insert more number of records than declared then it is not possible. We should know array size at the compile time itself.
* Inserting and deleting the records from the array would be costly since we add / delete the elements from the array, we need to manage memory space too.
* It does not verify the indexes while compiling the array. In case there is any indexes pointed which is more than the dimension specified, then we will get run time errors rather than identifying them at compile time.

## How to declare an array?

dataType arrayName[arraySize];

**For example,**

int number[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.

## How to initialize an array?

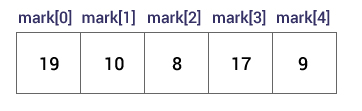
It is possible to initialize an array during declaration. For example,

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.



## Code1 for 1D- Array:

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

//Print the elements stored in the array

#include <stdio.h>

int main(int argc, char const \*argv[])

{

    int number[4];

    printf("Enter 4 numbers \n");

    for (int i = 0; i < 4; i++)

    {

        scanf("%d", &number[i]);

    }

    printf("Diplaying your enterd numbers\n");

    for (int i = 0; i < 4; i++)

    {

        printf("The n%d: %d\n", i, number[i]);

    }

    return 0;

}

**Code2 for 1D- Array:**

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

#include <stdio.h>

int main(int argc, char const \*argv[])

{

    int n, number[5];

    double average, sum = 0;

    printf("Enter the numbers of elements you want to add: ");

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        printf("Element E%d: ", i + 1);

        scanf("%d", &number[i]);

        sum = sum + number[i]; //sum += number[i];

    }

    printf("The sum is %.3lf \n", sum);

    average = sum / n;

    printf("The average is %.3lf", average);

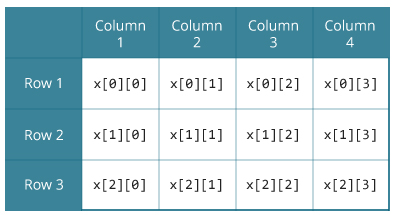
    return 0;

}

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.



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}}};

**“There are lot of ways of initialization so we can use any good or understandable way.”**

**Code3 for 2D- Array:**

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

#include <stdio.h>

const city = 2;

const week = 7;

int main()

{

    int temprature[2][7];

    // 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", &temprature[i][j]);

        }

    }

    printf("\n Displaying your Entry \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, temprature[i][j]);

        }

    }

    return 0;

}

**Output:**

City: 1, Day: 1 =>11

City: 1, Day: 2 =>23

City: 1, Day: 3 =>45

City: 1, Day: 4 =>6

City: 1, Day: 5 =>78

City: 1, Day: 6 =>66

City: 1, Day: 7 =>88

City: 2, Day: 1 =>9

City: 2, Day: 2 =>3

City: 2, Day: 3 =>22

City: 2, Day: 4 =>11

City: 2, Day: 5 =>12

City: 2, Day: 6 =>11

City: 2, Day: 7 =>33

Displaying your Entry

City: 1, Day: 1 =>11

City: 1, Day: 2 =>23

City: 1, Day: 3 =>45

City: 1, Day: 4 =>6

City: 1, Day: 5 =>78

City: 1, Day: 6 =>66

City: 1, Day: 7 =>88

City: 2, Day: 1 =>9

City: 2, Day: 2 =>3

City: 2, Day: 3 =>22

City: 2, Day: 4 =>11

City: 2, Day: 5 =>12

City: 2, Day: 6 =>11

City: 2, Day: 7 =>33

**Code4 for 2D-Array:**

#include <stdio.h>

int main(int argc, char const \*argv[])

{

    int matricA[2][2], matricB[2][2], resultmatricR[2][2];

    //[Code of 1st Matric] Taking user's input for Matric B as per place of element.

    printf("\n Enter the elements of Matric A (1st matric) \n");

    for (int i = 0; i < 2; i++)

        for (int j = 0; j < 2; j++)

        {

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

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

        }

    //[Code of 2nd Matric] Taking user's input for Matric B as per place of element.

    printf("\n Enter the Elements of Matric B (2nd matric) \n");

    for (int i = 0; i < 2; i++)

        for (int j = 0; j < 2; j++)

        {

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

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

        }

    //code for calculation of sum of both Matrics A and B

    for (int i = 0; i < 2; i++)

        for (int j = 0; j < 2; j++)

        {

            resultmatricR[i][j] = matricA[i][j] + matricB[i][j];

        }

    //displaying the result

    printf("\n Sum Of Matrics (A+B): \n");

    for (int i = 0; i < 2; ++i)

        for (int j = 0; j < 2; ++j)

        {

            printf("\t %d ", resultmatricR[i][j]);

            if (j == 1)          //this is for printing the second row from next line

                printf("\n");

        }

    return 0;

}

Here you can also use double or float for storing or calculation of point’.’ Values.

**Output:**

Enter the elements of Matric A (1st matric)

A11: 2

A12: 3

A21: 2

A22: 2

Enter the Elements of Matric B (2nd matric)

B11: 3

B12: 4

B21: 5

B22: 2

Sum Of Matrics (A+B):

5 7

7 4

**Code4 for 3D Array:**

#include <stdio.h>

int main()

{

    int test[2][3][2];

    printf("Enter 12 numbers for testing \n");

//code for accepting users input

    for (int x = 1; x <= 12; x++)

        for (int i = 0; i < 2; i++)

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

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

                {

                    printf("Number %d: ", x++);

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

                }

//code for Displaying

    printf("\n Diplaying Test 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 Values for Number %d are: [%d] [%d] [%d] \n ", test[i][j][k], i, j, k);

            }

    return 0;

}

**Output:**

Enter 12 numbers for testing

Number 1: 1

Number 2: 2

Number 3: 3

Number 4: 4

Number 5: 5

Number 6: 45

Number 7: 63

Number 8: 21

Number 9: 85

Number 10: 8

Number 11: 3

Number 12: 7

Diplaying Test Values

Test Values for Number 1 are: [0] [0] [0]

Test Values for Number 2 are: [0] [0] [1]

Test Values for Number 3 are: [0] [1] [0]

Test Values for Number 4 are: [0] [1] [1]

Test Values for Number 5 are: [0] [2] [0]

Test Values for Number 45 are: [0] [2] [1]

Test Values for Number 63 are: [1] [0] [0]

Test Values for Number 21 are: [1] [0] [1]

Test Values for Number 85 are: [1] [1] [0]

Test Values for Number 8 are: [1] [1] [1]

Test Values for Number 3 are: [1] [2] [0]

Test Values for Number 7 are: [1] [2] [1]