



(<https://www.educba.com/software-development/>)

← (<https://www.educba.com/arrays-in-c-programming/>)

→ (<https://www.educba.com/3d-arrays-in-c/>)

2-D Arrays in C



`#include <stdio.h>`

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

`int b[2][3];`

	Columns		
Rows	b[0][0]	b[0][1]	b[0][2]
	b[1][0]	b[1][1]	b[1][2]

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

educba.com

Introduction to 2-D Arrays in C

Arrays can be defined as collection of elements or data that are of similar or different data types, which is implemented in one or more dimensions with respect to the requirements provided to the program developer. 2-D or two dimensional array are represented as 'data variable[n][n]', where datatype can be an int, char, etc, and the [n][n] is n*n to represent the position of the value of the variable in the array. A few basic operations necessary for all the





(<https://www.educba.com/software-development/>)

CONCEPTS IN 2-D Arrays in C

We can define arrays in

Start Your Free Software Development Course

Web development, programming languages, Software testing & others

- Single-Dimensional
- Double-Dimensional

And so on up to N-Dimensional based upon the requirement. But here we are going to deal with 2-D Arrays. As the name suggests, 2-D Arrays can be a matrix representation of data, which are created to implement a relational database lookalike data structure and can be stored in tabular forms. It provides ease of holding the bulk data which can be passed to any number of functions based on the requirement. The data in these arrays can be accessed through the row and column ids.

How can we define and implement them? Where can we use them? Going further, let's understand those concepts.

In C, Dimensional arrays can be declared as follows:

Syntax

```
Datatype variablename[size1][size2][size3]...[sizen];
```

So, in the same way, we can declare the 2-D array as:

```
int b[2][3];
```





(<https://www.educba.com/software-development/>)

	Columns		
Rows	b[0][0]	b[0][1]	b[0][2]
	b[1][0]	b[1][1]	b[1][2]

The data inside the array can be accessed through the above representation. In 2-D arrays representation, the first square bracket represents the number of rows, and the second one is for the number of columns. The index representation of the array for the first element always starts with zero and ends with size-1. Array variable (here b) always holds the base address of the memory block and is called an internal pointer variable.

So, for example, if the number of rows is 3, then the index representation for accessing the data in rows will be 0, 1 and 2. The same logic applies to the column indexes too. For the above representation, to get the data of the 2nd row 3rd column, we can access by b[1][2].

🔗 Popular Course in this category



C Programming Training (3 Courses, 5 Project)

3 Online Courses | 5 Hands-on Projects | 34+ Hours | Verifiable Certificate of Completion | Lifetime Access

★★★★★ 4.5 (8,612 ratings)

Course Price

\$79 ~~\$399~~

[View Course](#)

(<https://www.educba.com/software-development/courses/c-programming-course/>)





(<https://www.educba.com/software-development/>)

Java Training (40 Courses, 29 Projects, 4 Quizzes) (<https://www.educba.com/software-development/courses/java-course/?btnz=edu-blg-inline-banner1>)

Initializing Arrays

We have two different methods in initializing the values in C. The methods only differ syntactically.

Below is one of them.

```
int b[2][3] = {
    {1,2,3}, /* Row 1,with index 0 */
    {4,5,6} /* Row 2,with index 0 */
};
```

Another way of initializing is as follows:

```
int b[2][3] = {1,2,3,4,5,6};
```

Generally, the first method of initialization is preferred as we can clearly understand and visualize the rows and columns of 2-D Arrays in C.

Below is the example for the pictorial representation of elements and their address for array b.

The elements of an array are usually stored in consecutive memory locations based upon the data type of the elements.

	b[0][0]	b[0][1]	b[0][2]	b[1][0]	b[1][1]	b[2][2]
Elements	1	2	3	4	5	6
Address	100	102	104	106	108	110
	1st row			2nd row		



Inserting Elements in 2-D Arrays



[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

Here, elements can be dynamically inserted by the user, as per the requirements. Below is an example code for inserting the elements.

```
#include <stdio.h>

int main()
{
    int b[2][3];
    int i,j,num;
    printf("Enter elements into 2-D array: ");
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d" , &b[i][j]);
        }
    }
}
```

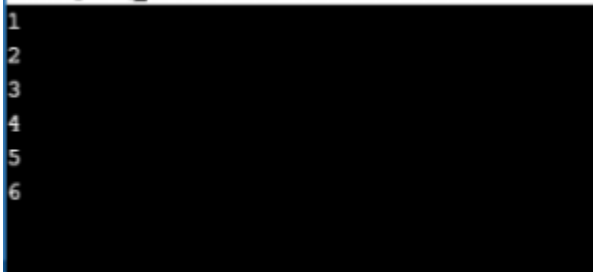
As observed in the code:

1. First, we are declaring the array variable and the dimensions of the array with the number of rows and columns.
2. We are then declaring two variables for iterating over the elements in the array.
3. Then, for loops are used. The outside for loop is for the rows iteration and the inside is for the columns.
4. Scanf function is used to read the data as we input, and then place the value inserted at

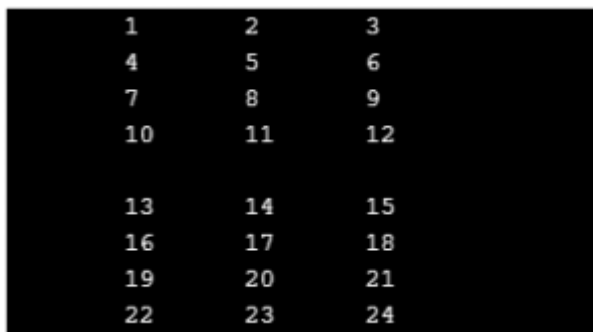




[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)



As we have not used the printf function to display the output, the program written had only read the user inputted values. After writing the print function (using for loops), the output would be displayed as:



Update Elements in 2-D Arrays

The updating of elements in an array can be done by either specifying a particular element to be replaced or by identifying a position where the replacement has to be done. For updating, we generally require the following details.

1. Elements of an array
2. Position/element, where it has to be inserted
3. The value to be inserted.



For updating the data in an array through element details, first, we need to search for that



[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

Firstly, let us go through an example where the position of the element to be updated is already known.

```
#include <stdio.h>

int main()
{
    int b[2][3];
    int i,j,num;
    printf("Enter elements into 2-D array: ");
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d" , &b[i][j]);
        }
    }
    b[0][2]=10;
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("\t%d" , b[i][j]);
        }
        printf("\n");
    }
}
```





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

the data in that position has been updated.

Output for above is as follows:

```
2
3
4
5
6
7
2    3    10
5    6    7
```

In the second example, we are going to show how the position of the element can be dynamically taken as a user inputted value and update the value of the element at that particular position.

```
#include <stdio.h>

int main()
{
    int b[2][3];
    int i,j,num;
    printf("Enter elements into 2-D array: ");
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            scanf("%d" , &b[i][j]);
        }
    }
}
```





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

```
scanf("%d" , &num);  
b[i][j]=num;  
for(i=0;i<2;i++)  
{  
for(j=0;j<3;j++)  
{  
printf("\t%d" , b[i][j]);  
}  
printf("\n");  
}  
return 0;  
}
```

Here, we used the scanf function to read the value given by the user as per their choice for the position of an element based on row and column numbers.

The output is as follows:

```
1  
2  
3  
4  
5  
6  
Enter the value of row and coulmn number :0  
2  
Enter the number you want to update with:9  
1      2      9  
4      5      6
```



As an exercise, can you try writing a program in updating the whole row of the matrix with



(<https://www.educba.com/software-development/>)

Our array size:

```
1
2
3
4
5
6
Enter the value of row and coulumn number :2
1
Enter the number you want to update with:6
      1      2      3
      4      5      6
```

Notice that as we had not written any if/else condition or try/catch blocks, the output of the matrix does not change. However, we can write the code using the above-mentioned conditions to display errors for such cases.

Deleting Elements in 2-D Arrays

After the concepts of insertion and updating of the data inside the array, let's now see how we can delete an entire row from the array.

We have written a program in a simple format so that the concept of different operations in a 2-d array can be understood easily.

```
#include <stdio.h>

int main()
{
    int b[2][3],i,j,num,x;
    printf("Enter elements into 2-D array: ");
    for(i=0;i<2;i++)
```





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

```
}  
}  
printf("Enter the value of row number :");  
scanf("%d", &x);  
for(i=0;i<2;i++)  
{  
    if(i==x)  
    {  
        for(j=0;j<3;j++)  
        {  
            if((i+1)<2)  
            {  
                printf("\t%d" , b[i+1][j]);  
            }  
        }  
        i++;  
    }  
    else  
    {  
        for(j=0;j<3;j++)  
        {  
            printf("\t%d" , b[i][j]);  
        }  
    }  
  
    printf("\n");
```





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

1. Took the values of an array dynamically
2. Asked the user to input the number (index) of the row that has to be deleted.
3. Using for loop iteration, we are comparing if the row number and the user input number are matching or not.
4. If they are matching and if the row number is less than the size of an array, we are printing the next row. Else, we are printing the row as it is.

The output is as follows:

```
1
2
3
4
5
6
Enter the value of row number :1
      1      2      3
```

What if, I give the row number outside the array boundary?

```
1
2
3
4
5
6
Enter the value of row number :2
      1      2      3
      4      5      6
```



It will not find the row to delete and exit the program by printing the whole array.



[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

As an exercise, can you try in deleting a particular element for the 2-d array now?

Conclusion

In this section, we have learned the basic operations on 2-dimensional arrays. These 2-d arrays are useful in real-time with matrix operations and many mathematical calculations.

Arrays can even be used in displaying calendars, placements of the parking lot, and we can even have a chess game.

Many other data structures like Linked lists, Queue, Graphs, Trees have to use this concept of 2-D arrays as the basic requirement in storing and accessing the locations of different elements. Try solving the basic operations of the 2d arrays and have fun learning C.

Recommended Articles


This is a guide to 2-D Arrays in C. Here we discuss the Introduction, Initializing Arrays, Inserting, Updating, Deleting Elements in a 2-D Arrays. You may also look at the following articles to learn more –

1. [Arrays in C++ \(https://www.educba.com/arrays-in-c-plus-plus/\)](https://www.educba.com/arrays-in-c-plus-plus/)
2. [Arrays in JavaScript \(https://www.educba.com/arrays-in-javascript/\)](https://www.educba.com/arrays-in-javascript/)
3. [Arrays in C# \(https://www.educba.com/arrays-in-c-sharp/\)](https://www.educba.com/arrays-in-c-sharp/)
4. [Arrays in PHP \(https://www.educba.com/arrays-in-php/\)](https://www.educba.com/arrays-in-php/)





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

 34+ Hours

☒ Verifiable Certificate of Completion

☒ Lifetime Access

Learn More

[_https://www.educba.com/software-development/courses/c-programming-course/?btnz=educba-inline-banner3\)](https://www.educba.com/software-development/courses/c-programming-course/?btnz=educba-inline-banner3)

About Us

Blog (<https://www.educba.com/blog/?source=footer>)

Who is EDUCBA? (<https://www.educba.com/about-us/?source=footer>)

Sign Up (<https://www.educba.com/software-development/signup/?source=footer>)

Corporate Training (<https://www.educba.com/corporate/?source=footer>)

Certificate from Top Institutions (<https://www.educba.com/educbalive/?source=footer>)

Contact Us (<https://www.educba.com/contact-us/?source=footer>)





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

Privacy Policy (<https://www.educba.com/privacy-policy/?source=footer>)

Apps

iPhone & iPad (<https://itunes.apple.com/in/app/educba-learning-app/id1341654580?mt=8>)

Android (<https://play.google.com/store/apps/details?id=com.educba.www>)

Resources

Free Courses (<https://www.educba.com/software-development/free-courses/?source=footer>)

Java Tutorials (<https://www.educba.com/software-development/software-development-tutorials/java-tutorial/?source=footer>)

Python Tutorials (<https://www.educba.com/software-development/software-development-tutorials/python-tutorial/?source=footer>)

All Tutorials (<https://www.educba.com/software-development/software-development-tutorials/?source=footer>)

Certification Courses

All Courses (<https://www.educba.com/software-development/courses/?source=footer>)

Software Development Course - All in One Bundle
(<https://www.educba.com/software-development/courses/software-development-course/?source=footer>)

Become a Python Developer (<https://www.educba.com/software-development/courses/python-certification-course/?source=footer>)

Java Course (<https://www.educba.com/software-development/courses/java-course/?source=footer>)





[\(https://www.educba.com/software-development/\)](https://www.educba.com/software-development/)

VB.NET Course (<https://www.educba.com/software-development/courses/vb-net-course/?source=footer>)

PHP Course (<https://www.educba.com/software-development/courses/php-course/?source=footer>)

© 2022 - EDUCBA. ALL RIGHTS RESERVED. THE CERTIFICATION NAMES ARE THE TRADEMARKS OF THEIR RESPECTIVE OWNERS.

