

**Software Development** ▼[Tutorials](#)[Articles](#)[Ebooks](#)[Free Practice Tests](#)[On-demand Webinars](#)

[Home](#) > [Resources](#) > [Software Development](#) > [Data Structure Tutorial for Beginners](#) > Arrays in Data Structure: A Guide With Examples



Arrays in Data Structure: A Guide With Examples

Lesson 1 of 63

By Ravikiran A S

Last updated on Jul 15, 2024

 339187

Array In Data Structure | What Is An Array In Data Structure? | Data Structur...



[< Previous](#)

[Next >](#)

Tutorial Playlist



Table of Contents

[What Are Arrays in Data Structures?](#)

[Why Do You Need an Array in Data Structures?](#)

[What Are the Types of Arrays?](#)

[How Do You Declare an Array?](#)

[How Do You Initialize an Array?](#)

[View More](#)

Arrays in [data structures](#) help solve some high-level problems like the "longest consecutive subsequence" program or some easy tasks like arranging the same things in ascending order. The concept is to collect many objects of the same kind.

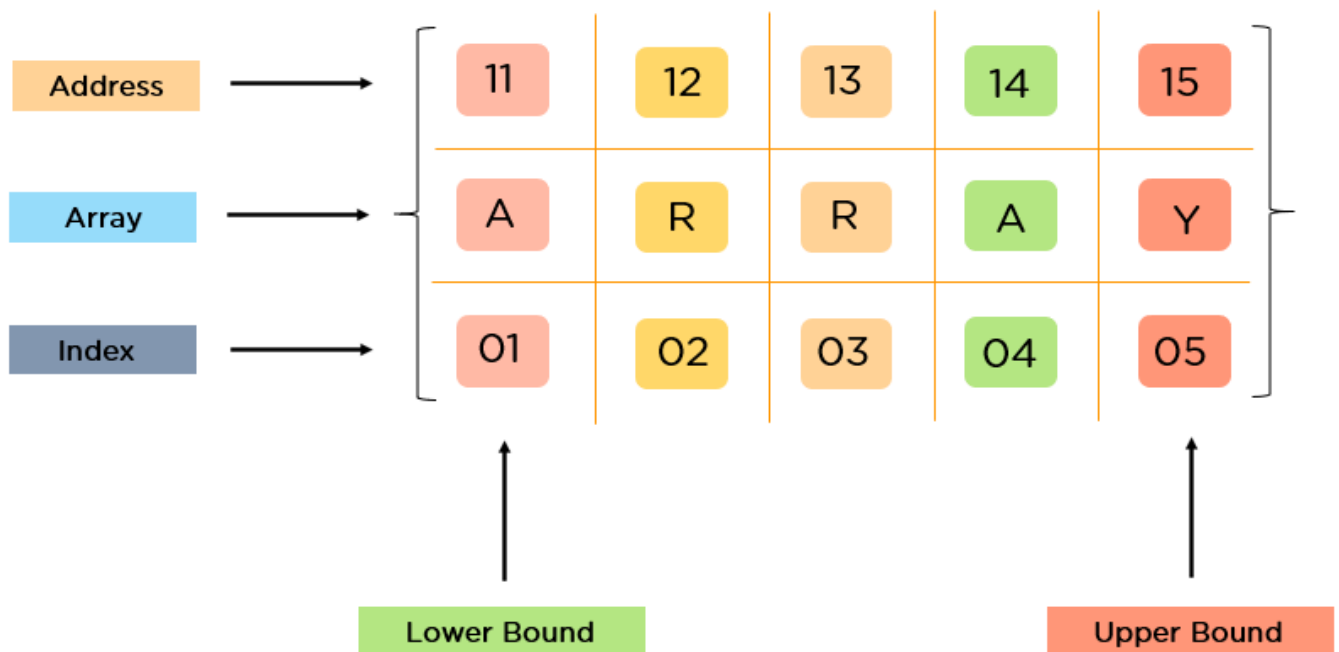


Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN Stack

EXPLORE PROGRAM

What Are Arrays in Data Structures?

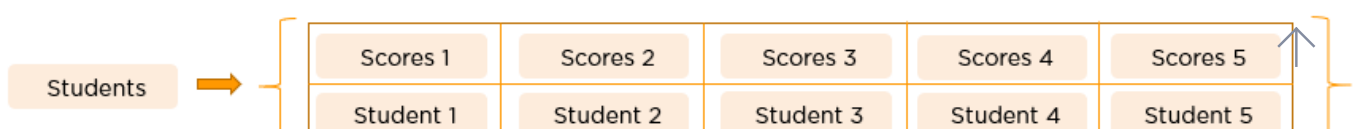


An array is a linear [data structure](#) that collects elements of the same data type and stores them in contiguous and adjacent memory locations. Arrays work on an index system starting from 0 to $(n-1)$, where n is the size of the array.

It is an array, but there is a reason that arrays came into the picture.

Also Read: [Top Data Structures and Algorithms Every Data Science Professional Should Know](#)

Why Do You Need an Array in Data Structures?



Let's suppose a class consists of ten students, and the class has to publish their results. If you had declared all ten variables individually, it would be challenging to manipulate and maintain the data.

If more students were to join, it would become more difficult to declare all the variables and keep track of it. To overcome this problem, arrays came into the picture.

What Are the Types of Arrays?

There are majorly two types of arrays, they are:

One-Dimensional Arrays:

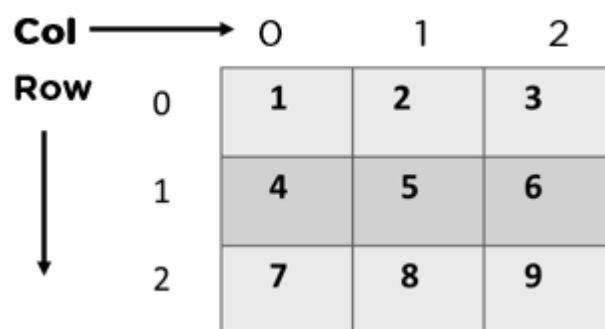


You can imagine a 1d array as a row, where elements are stored one after another.

Multi-Dimensional Arrays:

These multi-dimensional arrays are again of two types. They are:

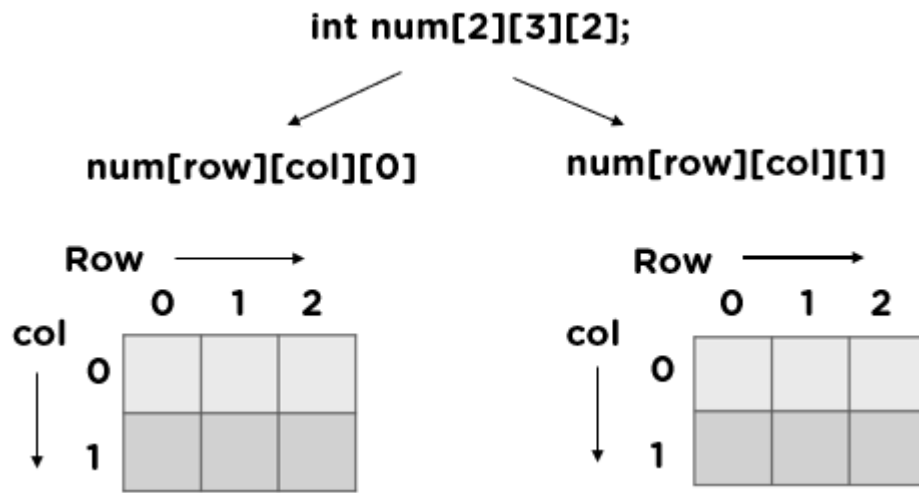
Two-Dimensional Arrays:



You can imagine it like a table where each cell contains elements.

Three-Dimensional Arrays:





You can imagine it like a cuboid made up of smaller cuboids where each cuboid can contain an element.

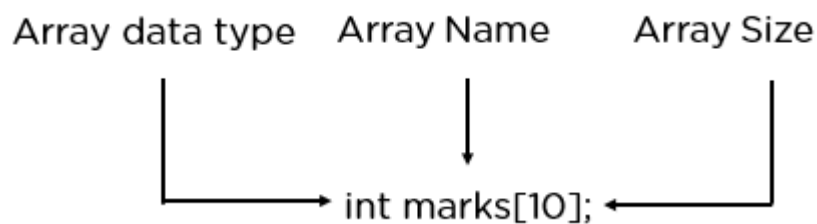
In this "arrays in data structures" tutorial, you will work around one-dimensional arrays.

Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN Stack

EXPLORE PROGRAM

How Do You Declare an Array?



Arrays are typically defined with square brackets with the size of the arrays as its argument.

Here is the syntax for arrays:



1D Arrays: `int arr[n];`

2D Arrays: `int arr[m][n];`

3D Arrays: `int arr[m][n][o];`

How Do You Initialize an Array?

You can initialize an array in four different ways:

- **Method 1:**

```
int a[6] = {2, 3, 5, 7, 11, 13};
```

- **Method 2:**

```
int arr[] = {2, 3, 5, 7, 11};
```

- **Method 3:**

```
int n;
```

```
scanf("%d",&n);
```

```
int arr[n];
```

```
for(int i=0;i<n;i++)
```

```
{
```

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

```
}
```

- **Method 4:**

```
int arr[5];
```



```
arr[0]=1;
```

```
arr[1]=2;
```

```
arr[2]=3;
```

```
arr[3]=4;
```

```
arr[4]=5;
```

Preparing Your Blockchain Career for 2024

Free Webinar | 5 Dec, Tuesday | 9 PM IST

REGISTER NOW

How Can You Access Elements of Arrays in Data Structures?

5	10	25	30	50
0	1	2	3	4

You can access elements with the help of the index at which you stored them. Let's discuss it with a code:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
int a[5] = {2, 3, 5, 7, 11};
```

```
printf("%d\n",a[0]); // we are accessing
```



```
printf("%d\n",a[1]);

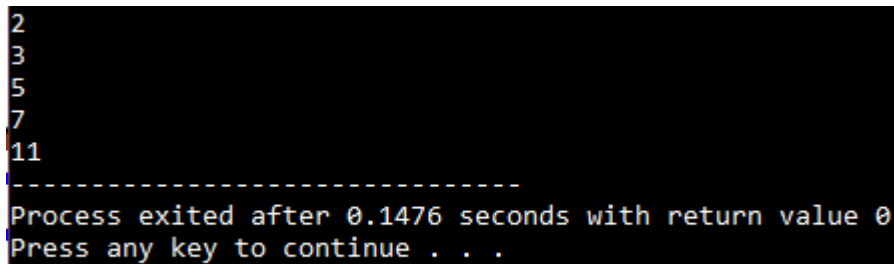
printf("%d\n",a[2]);

printf("%d\n",a[3]);

printf("%d",a[4]);

return 0;

}
```



```
2
3
5
7
11
-----
Process exited after 0.1476 seconds with return value 0
Press any key to continue . . .
```

In this “Arrays in Data structures” tutorial, you have seen the basics of array implementation, now you will perform operations on arrays.

<

What Operations Can You Perform on an Array?

- Traversal
- Insertion
- Deletion
- Searching
- Sorting

Traversing the Array:



10	20	30	40	50
----	----	----	----	----

Traversal in an array is a process of visiting each element once.

Code:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
int a[5] = {2, 3, 5, 7, 11};
```

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

```
{
```

```
//traversing ith element in the array
```

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

```
}
```

```
return 0;
```

```
}
```

```
2
3
5
7
11
-----
Process exited after 0.1476 seconds with return value 0
Press any key to continue . . .
```

Want a Top Software Development Job? Start Here!



Full Stack Developer - MERN Stack

[EXPLORE PROGRAM](#)

Insertion:



Insertion in an array is the process of including one or more elements in an array.

Insertion of an element can be done:

- At the beginning
- At the end and
- At any given index of an array.

At the Beginning:

Code:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int array[10], n,i, item;
```

```
    printf("Enter the size of array: ");
```

```
    scanf("%d", &n);
```



```
    printf("\nEnter Elements in array: ");
```

```
for(i=0;i<n;i++)  
  
{  
  
scanf("%d", &array[i]);  
  
}  
  
printf("\n enter the element at the beginning");  
  
scanf("%d", &item);  
  
n++;  
  
for(i=n; i>1; i--)  
  
{  
  
array[i-1]=array[i-2];  
  
}  
  
array[0]=item;  
  
printf("resultant array element");  
  
for(i=0;i<n;i++)  
  
{  
  
printf("\n%d", array[i]);  
  
}  
  
getch();  
  
return 0;  
  
}
```



```

Enter the size of array: 5

Enter Elements in array: 2 3 5 7 11

enter the element at the beginning:1
resultant array element:
1
2
3
5
7
11
-----
Process exited after 19.94 seconds with return value 0
Press any key to continue . . .

```

At the End:

Code:

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
int main()
```

```
{
```

```
int array[10], i, values;
```

```
printf("Enter 5 Array Elements: ");
```

```
for(i=0; i<5; i++)
```

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

```
printf("\nEnter Element to Insert: ");
```

```
scanf("%d", &values);
```

```
array[i] = values;
```

```
printf("\nThe New Array is:\n");
```

```
for(i=0; i<6; i++)
```

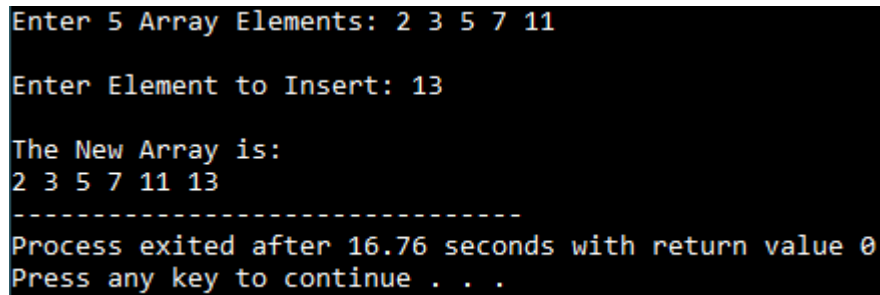


```
printf("%d ", array[i]);

getch();

return 0;

}
```

A terminal window with a black background and yellow text. It shows the execution of a program where 5 array elements (2, 3, 5, 7, 11) are entered, followed by an element to insert (13). The output shows the new array (2, 3, 5, 7, 11, 13) and a message indicating the process exited after 16.76 seconds with a return value of 0, followed by a prompt to press any key to continue.

```
Enter 5 Array Elements: 2 3 5 7 11

Enter Element to Insert: 13

The New Array is:
2 3 5 7 11 13
-----
Process exited after 16.76 seconds with return value 0
Press any key to continue . . .
```

At a Specific Position:

Code:

```
#include <stdio.h>

int main()

{

int array[100], pos, size, val;

printf("Enter size of the array:");

scanf("%d", &size);

printf("\nEnter %d elements\n", size);

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

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

printf("Enter the insertion location\n");
```



```
scanf("%d", &pos);

printf("Enter the value to insert\n");

scanf("%d", &val);

for (int i = size - 1; i >= pos - 1; i--)

    array[i+1] = array[i];

array[pos-1] = val;

printf("Resultant array is\n");

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

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

return 0;

}
```

```
Enter size of the array:5
Enter 5 elements
2 3 5 7 11
Enter the insertion location
2
Enter the value to insert
13
Resultant array is
2
13
3
5
7
11
-----
Process exited after 14.82 seconds with return value 0
Press any key to continue . . .
```

Want a Top Software Development Job? Start Here!

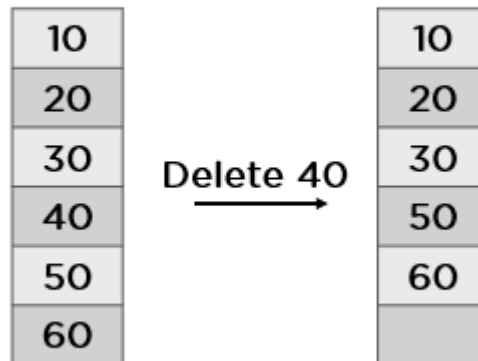


Full Stack Developer - MERN Stack

[EXPLORE PROGRAM](#)

Deletion:

Before deletion After deletion



Deletion of an element is the process of removing the desired element and re-organizing it.

You can also do deletion in different ways:

- At the beginning
- At the end

At the Beginning:

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int n,array[10];
```

```
    printf("enter the size of an array");
```

```
    scanf("%d",&n);
```

```
    printf("enter elements in an array");
```



```

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

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

n--;

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

    array[i]=array[i+1];

printf("\nafter deletion ");

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

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

}

```

```

enter the size of an array: 5
enter elements in an array: 2 3 5 7 11

after deletion:
3
5
7
11

-----
Process exited after 7.997 seconds with return value 0
Press any key to continue . . .

```

At the End:

```

#include<stdio.h>

int main()

{

    int n,array[10];

    printf("enter the size of an array");

    scanf("%d" ,&n);

```




```
printf("enter elements in an array");

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

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

printf("\nafter deletion array elements are");

for(int i=0;i<n-1;i++)

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

}
```

```
enter the size of an array:5
enter elements in an array:2 3 5 7 11

after deletion array elements:
2
3
5
7

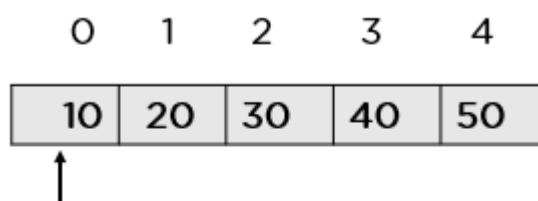
-----
Process exited after 15.99 seconds with return value 0
Press any key to continue . . .
```

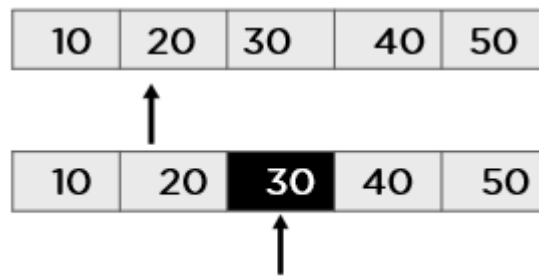
Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN Stack

EXPLORE PROGRAM

Searching for an Element





The method of searching for a specific value in an array is known as searching.

There are two ways we can search in an array, they are:

- [Linear search](#)
- Binary search

Linear Search:

Code:

```
#include <stdio.h>
```

```
int linear(int a[], int n, int x)
```

```
{
```

```
for (int i = 0; i < n; i++)
```

```
if (a[i] == x)
```

```
return i;
```

```
return -1;
```

```
}
```

```
int main(void)
```

```
{
```

```
int a[] = { 2, 3, 4, 10, 40 };
```

```
int x = 10;
```



```
int n = sizeof(a) / sizeof(a[0]);

// Function call

int result = linear(a, n, x);

if(result == -1)

{

printf("Element is not present in array");

}

else

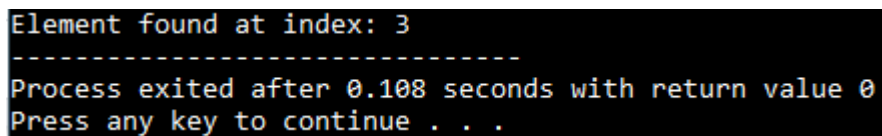
{

printf("Element found at index: %d", result);

}

return 0;

}
```



```
Element found at index: 3
-----
Process exited after 0.108 seconds with return value 0
Press any key to continue . . .
```

Binary Search:

```
#include <stdio.h>

int binary(int a[], int lt, int rt, int k)

{

if (rt >= lt) {
```



```
int mid = lt + (rt - l) / 2;

// check If element is at the middle

if (a[mid] == k)

return mid;

//check if element is at the left side of mid

if (a[mid] > x)

return binary(a, lt, mid - 1, k);

// Else element is at the right side of mid

return binary(a, mid + 1, rt, k);

}

// if element not found

return -1;

}

int main(void)

{

int a[] = { 2, 3, 5, 7, 11 };

int n = sizeof(a) / sizeof(a[0]);

int k = 11;

int res = binary(arr, 0, n - 1, k);

if(res == -1)
```



```
{  
  
printf("Element is not found")  
  
}  
  
else  
  
{  
  
printf("Element found at index %d",res);  
  
}  
  
return 0;  
  
}
```

```
Element found at index 4  
-----  
Process exited after 0.08186 seconds with return value 0  
Press any key to continue . . .
```

Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN Stack

EXPLORE PROGRAM

Sorting:

Before sorting

30
10
50



After sorting

10
20
30



20	40
40	50

Sorting in an array is the process in which it sorts elements in a user-defined order.

Code:

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int temp, size, array[100];
```

```
    printf("Enter the size \n");
```

```
    scanf("%d", &size);
```

```
    printf("Enter the numbers \n");
```

```
    for (int i = 0; i < size; ++i)
```

```
    {
        scanf("%d", &array[i]);
```

```
    }
    for (int i = 0; i < size; ++i)
```

```
    {
```

```
        for (int j = i + 1; j < size; ++j)
```

```
        {
```

```
            if (array[i] > array[j])
```

```
            {
```

```
                temp = array[i];
```



```
        array[i] = array[j];

        array[j] = temp;

    }

}

}

printf("sorted array:\n");

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

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

}
```

```
Enter the size
5
Enter the numbers
5 2 6 7 1
sorted array:
1
2
5
6
7

-----
Process exited after 34.49 seconds with return value 0
Press any key to continue . . .
```

Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN Stack

EXPLORE PROGRAM



What Are the Advantages of Arrays in Data Structures?



- Arrays store multiple elements of the same type with the same name.
- You can randomly access elements in the array using an index number.
- Array memory is predefined, so there is no extra memory loss.
- Arrays avoid memory overflow.
- 2D arrays can efficiently represent the tabular data.

What Are the Disadvantages of Arrays in Data Structures?





- The number of elements in an array should be predefined
- An array is static. It cannot alter its size after declaration.
- Insertion and deletion operation in an array is quite tricky as the array stores elements in continuous form.
- Allocating excess memory than required may lead to memory wastage.

Want a Top Software Development Job? Start Here!

Full Stack Developer - MERN Stack

EXPLORE PROGRAM

Next Steps

"[Linked list Data structures](#)" can be your next stop. Linked lists are made up of nodes that point to the next node. Linked lists are dynamic and have faster insertion/deletion time complexities.

Simplilearn is the world's #1 online bootcamp for digital economy skills training focused on helping people acquire the skills they need to thrive in the digital economy. We provide rigorous online training in disciplines such as [Cyber Security](#), [Cloud Computing](#), [Project Management](#), Digital Marketing, and [Data Science](#), among others.

If you have any questions about this 'Arrays in Data structures' tutorial, please feel free to leave them in the comments section below. Our 24/7 expert team will answer all your queries for you at the earliest.

If you are perhaps looking to explore and become a data professional, Simplilearn's Post Graduate Program in Data Engineering, offered in partnership with Purdue University and in collaboration with IBM would be the ideal starting point. Within just 8 months, this globally-[↑](#) recognized, completely online program will transform you into a fully-equipped, work-ready big

data and analytics professional. Aligned with top AWS and Azure certifications, and endorsed by Purdue and other top names in the industry, you will have all you need to fast track your career in the field, anywhere in the world.

Happy learning!

FAQs

1. How are arrays useful in programming?

Multiple pieces of data can be organized and managed using arrays. They are helpful for storing lists, collections, and tables of data, which facilitates more effective searching, sorting, and editing of elements.

2. What is the index of an array?

The position of an element within an array is indicated by a numeric value known as the array's index. Usually, it starts at 0 for the first element and goes up by 1 for each additional element.


3. What is dynamic array?

A dynamic array is an array-like data structure that adapts its size as elements are added or removed automatically. It offers dynamic memory allocation flexibility without requiring manual resizing.

4. How are multidimensional arrays different?

Multidimensional arrays are grid-like structures made up of arrays inside of arrays. They can be used to depict matrices, tables, and other intricate data structures where the elements have numerous dimensions.

Find our Full Stack Java Developer Masters Program Online Bootcamp in top cities:

Name	Date	Place
Full Stack Java Developer	Cohort starts on 18th Dec 2024,	

Masters Program	Weekend batch	Your City
Full Stack Java Developer Masters Program	Cohort starts on 8th Jan 2025, Weekend batch	Your City

About the Author



Ravikiran A S

Ravikiran A S works with Simplilearn as a Research Analyst. He an enthusiastic geek always in the hunt to learn the latest technologies. He is proficient with Java Programming Language, Bi...

[View More](#)

Recommended Resources



An Easy Guide To Understand
The C++ Ar...



A Guide on How to
Site Reliabili...

© 2009 -2024- Simplilearn Solutions.

Disclaimer

- PMP, PMI, PMBOK, CAPM, PgMP, PfMP, ACP, PBA, RMP, SP, OPM3 and the PMI ATP seal are the registered marks of the Project Management Institute, Inc.



