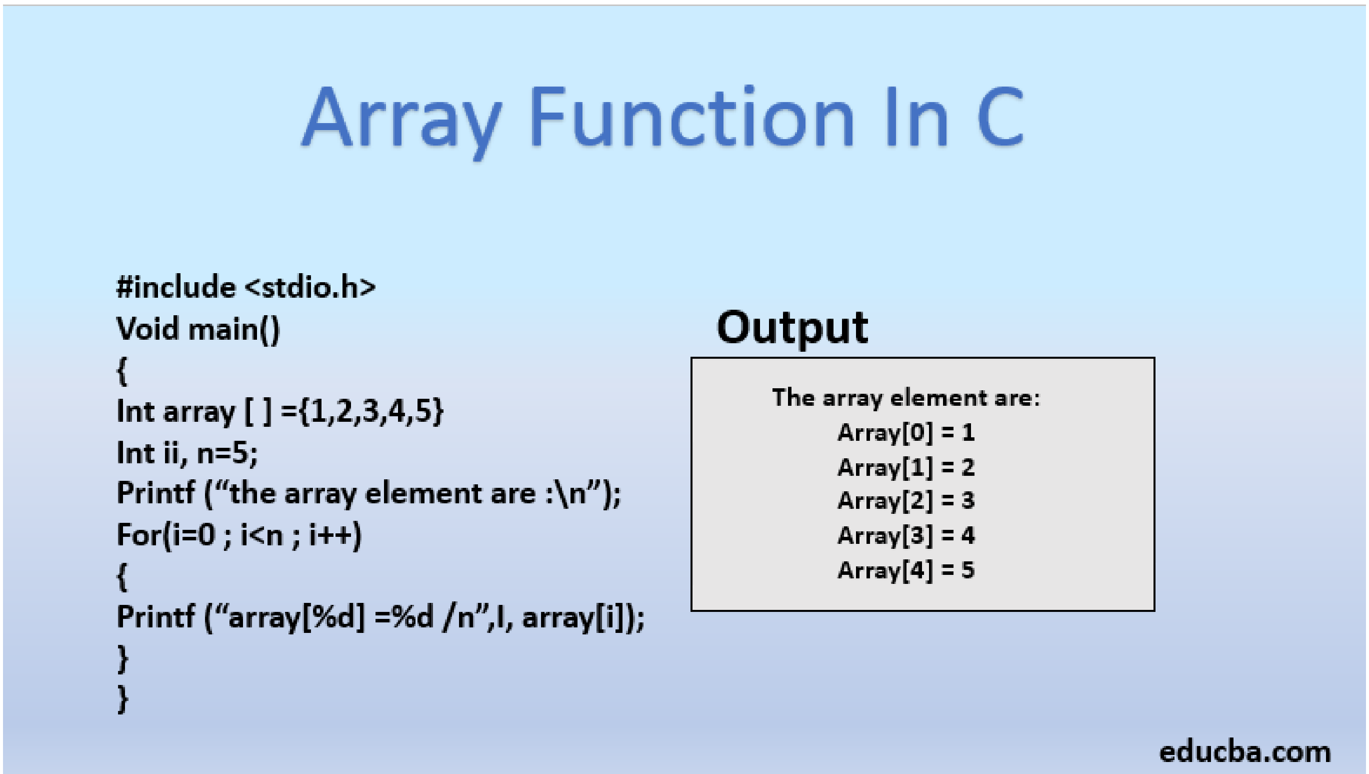
EDUCBA (https://www.educba.com/software-development/)

← (https://www.educba.com/multidimensional-array-in-c/)

→ (https://www.educba.com/strings-array-in-c/)



Array Function In C

```
#include <stdio.h>
Void main()
{
Int array [ ] ={1,2,3,4,5}
Int ii, n=5;
Printf (“the array element are :\n”);
For(i=0 ; i<n ; i++)
{
Printf (“array[%d] =%d /n”,I, array[i]);
}
}
```

Output

```
The array element are:
    Array[0] = 1
    Array[1] = 2
    Array[2] = 3
    Array[3] = 4
    Array[4] = 5
```

educba.com

Introduction to Array Functions in C

Array Functions in C is a type of data structure that holds multiple elements of the same type. The size of an array is fixed and the elements are collected in a sequential manner. There can be different dimensions of arrays and C programming does not limit the number of

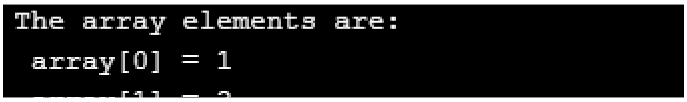
1) Traversing

Traversing an Array means going through each element of an Array exactly once. We start from the first element and go to the last element. An example of such a program that performs traversing operation on a linear Array is given below in C language.

Code:

```
#include <stdio.h>
void main()
{
int array[] = {1,2,3,4,5};
int i, n = 5;
printf(" The array elements are: \n " );
for( i=0;i < n; i++)
{
printf(" array[%d] = %d \n " , i, array[i] );
}
}
```

Output:



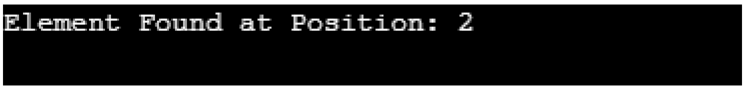
```
The array elements are:
 array[0] = 1
 array[1] = 2
```

The search operation is used to find a particular data item or element in an Array. We can perform searching in an unsorted array with the help of traversal of the Array. The linear traversal from the first element to the last element can be used to search if a given number is present in an Array and can also be used to find its position if present.

This is done by comparing each element with the given element (which is to be searched). Once the element is found, the search operation is stopped. Here is an example to show search operation performed on an Array in C

```
#include<stdio.h>
int findElement(int arr[], int n,  int key)
{
int i;
for (i = 0; i < n; i++)
if (arr[i] == key
return i;
return -1;
}
int main()
{
int arr[] = {1, 4, 0, 6, 3};
int n = sizeof(arr) / sizeof(arr[0]);
int key = 4;
int position = findElement(arr, n, key);
if (position == - 1)
printf("Element not found");
else
printf("Element Found at Position: %d", position + 1 );
return 0;
}
```

Output:



```
Element Found at Position: 2
```

be inserted in an array only if it has sufficient space to add it. If the size of an array is full already, a new element cannot be added. An example to show insert operation in an unsorted Array in C.

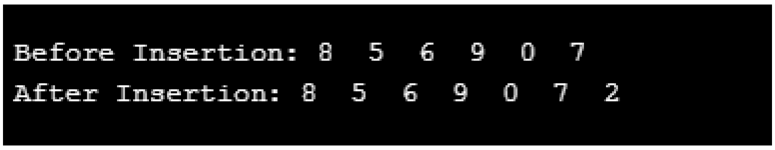
Code:

```
#include<stdio.h>
int insertSorted(int arr[], int n,  int key, int capacity)
{
if (n >= capacity)
return n;
arr[n] = key;
return (n + 1);
}
int main()
{
int arr[20] = {8, 5, 6, 9, 0, 7} ;
int capacity = sizeof(arr) / sizeof(arr[0]);
int n = 6;
int i, key = 2;
printf("\n Before Insertion: ");
for (i = 0; i < n; i++)

printf("%d  ", arr[i]);
n = insertSorted(arr, n, key, capacity);
printf("\n After Insertion: ");
```

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

Output:



```
Before Insertion: 8  5  6  9  0  7
After Insertion: 8  5  6  9  0  7  2
```

4) Deletion

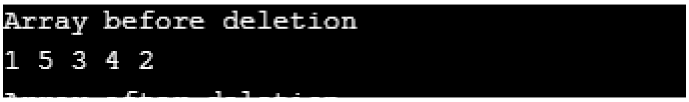
In delete operation, the element which already exists in the Array is searched (using linear search) and deleted, followed by the shifting of elements. The user enters the position of the element which is to be deleted from the array. Deletion operation, just like the insertion operation, does not affect the size of array. Also, the position of the element to be deleted should be within the size of array, since the deletion of an element beyond the size of Array is not possible. C program to show delete operation in an unsorted array.

Code:

```
#include<stdio.h>
int findElement(int arr[], int n, int key);
int deleteElement(int arr[], int n, int key)
{
int pos = findElement(arr, n, key);
if (pos == - 1)
{

printf("Element not found");
return n;}
int i;
int findElement(int arr[], int n, int key)
{
int i;
for (i = 0; i < n; i++)
if (arr[i] == key)
return i;return - 1;
}
int main()
{
int i;
int arr[] = {1, 5, 3, 4, 2};
int n = sizeof(arr) / sizeof(arr[0]);
int key = 3;
printf("Array before deletion\n");
for (i = 0; i < n; i++)
printf("%d  ", arr[i]);
n = deleteElement(arr, n, key);
printf("\nArray after deletion\n");
for (i = 0; i < n; i++)
printf("%d  ", arr[i]);
return 0;
}
```

Output:



```
Array before deletion
1 5 3 4 2
Array after deletion
```

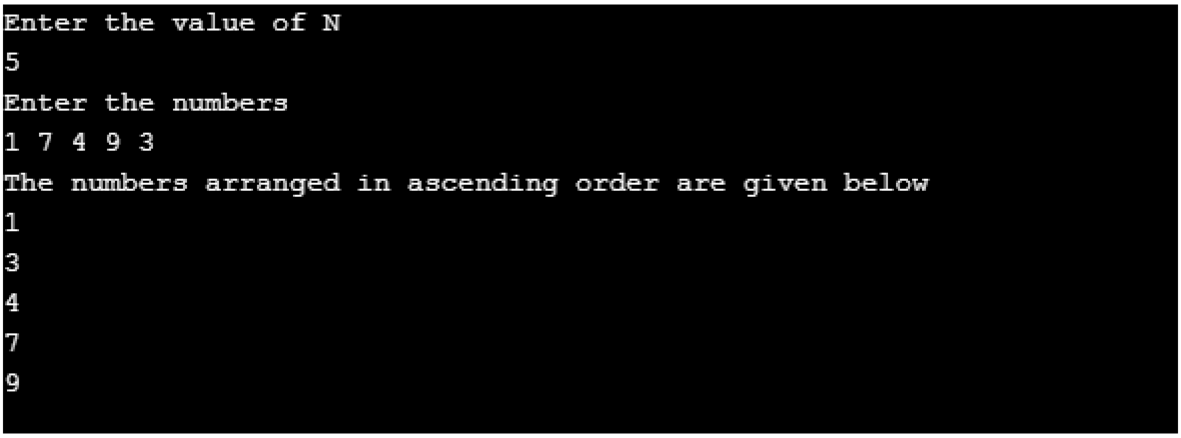
descending. Here is an example of sort operation on an Array in C

Code:

```
#include <stdio.h>
void main()
{
int i, j, a, n, number[30];
printf("Enter the value of N \n");
scanf("%d", &n);
printf("Enter the numbers \n");
for (i = 0; i < n; ++i)
scanf("%d", &number[i]);
for (i = 0; i < n; ++i)
{
for (j = i + 1; j < n; ++j)
{
if (number[i] > number[j])
{
a =  number[i];
number[i] = number[j];
number[j] = a;
}
}
}
printf("The numbers arranged in ascending order are given below
```

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

Output:



```
Enter the value of N
5
Enter the numbers
1 7 4 9 3
The numbers arranged in ascending order are given below
1
3
4
7
9
```

Different ways of Sorting an Array

Below are the different sorting methods for Array:

1) Bubble Sort

Bubble sort compares all the elements one by one and sorts them based on their values. It starts by comparing the first element with the second, if the first element is greater than the second element, it will swap both the elements, and carry on comparing the second and the third element, and so on.

2) Selection Sort

The basic idea behind selection sort is finding the least element in the unsorted array, replacing it with the first element. Then continue the same process with the rest of the unsorted array, i.e., from the second position, then from the third and so on.

3) Merge Sort

This method of sorting is based on the divide and conquer technique. It splits the array into two equal subarrays and continues until each subarray contains a single element, and then merges second element. The array elements are compared with each other in a sequential manner. The current element(the value to be sorted) is compared with all the elements in the sorted subarray. All the elements in the sorted subarray which are greater than the current element are shifted, and the current value is inserted. This process is repeated until the whole array is sorted.

5) Quick Sort

Quicksort, just like the merge sort, is also based on the divide and conquer algorithm. In this method, an element is picked as the pivot (generally the first element). Then, partitions of an array are made around the picked pivot i.e., all the elements less than the pivot will form one sub-array and all the elements greater than the pivot will form another. The procedure is repeated with the sub-arrays too until the whole array is sorted.

6) Heap Sort

The algorithm of heap sort (https://www.educba.com/heap-sort-in-c/) is based on the comparison. The maximum element is selected and placed in the end position. Then the second largest element is found and placed in the second last position. This process is repeated for all the elements.

Recommended Articles

This is a guide to Array Functions in C. Here we discuss the basic concept and different functions and ways of sorting an Array respectively. You can also go through our other related articles to learn more –

1. Arrays in C++ (https://www.educba.com/arrays-in-c-plus-plus/)
2. Arrays in R (https://www.educba.com/arrays-in-r/)
3. Functions in R (https://www.educba.com/functions-in-r/)

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)

QUIZ

EDUCBA

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

Terms and Conditions (https://www.educba.com/terms-and-conditions/?source=footer)

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)

QUIZ

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

ASP.NET Course (https://www.educba.com/software-development/courses/asp-net-course/?source=footer)

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)

QUIZ