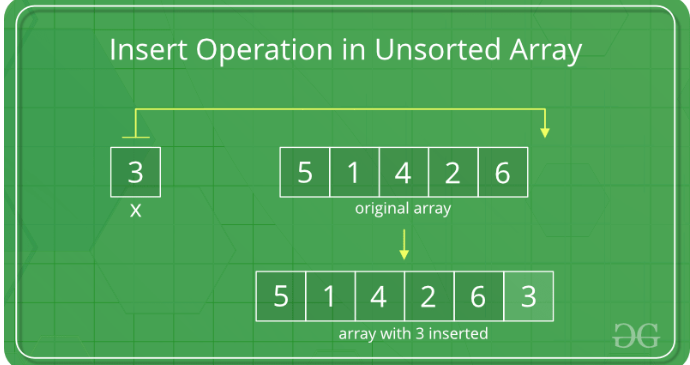
# **Array Operations**

# **Insert, and Delete in an Unsorted Array**

## **Insert Operation**

### **1. Insert at the end:**

In an unsorted array, the insert operation is faster as compared to a sorted array because we don’t have to care about the position at which the element is to be placed.



// C program to implement insert

// operation in an unsorted array.

#include <stdio.h>

// Inserts a key in arr[] of given capacity.

// n is current size of arr[]. This

// function returns n + 1 if insertion

// is successful, else n.

int insertSorted(int arr[], int n, int key, int capacity)

{

// Cannot insert more elements if n is

// already more than or equal to capacity

if (n >= capacity)

return n;

arr[n] = key;

return (n + 1);

}

// Driver Code

int main()

{

int arr[20] = { 12, 16, 20, 40, 50, 70 };

int capacity = sizeof(arr) / sizeof(arr[0]);

int n = 6;

int i, key = 26;

printf("\n Before Insertion: ");

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

printf("%d ", arr[i]);

// Inserting key

n = insertSorted(arr, n, key, capacity);

printf("\n After Insertion: ");

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

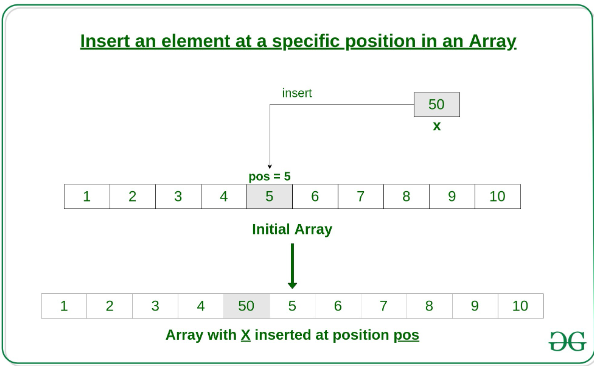
printf("%d ", arr[i]);

return 0;

}

### **Insert at any position:**

Insert operation in an array at any position can be performed by shifting elements to the right, which are on the right side of the required position



// C Program to Insert an element

// at a specific position in an Array

#include <stdio.h>

// Function to insert element

// at a specific position

void insertElement(int arr[], int n, int x, int pos)

{

// shift elements to the right

// which are on the right side of pos

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

arr[i + 1] = arr[i];

arr[pos] = x;

}

// Driver's code

int main()

{

int arr[15] = { 2, 4, 1, 8, 5 };

int n = 5;

printf("Before insertion : ");

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

printf("%d ", arr[i]);

printf("\n");

int x = 10, pos = 2;

// Function call

insertElement(arr, n, x, pos);

n++;

printf("After insertion : ");

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

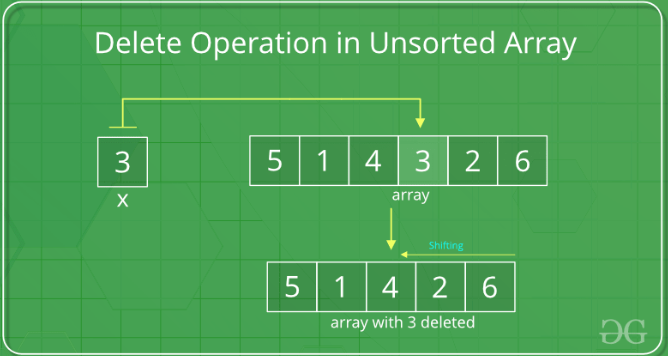
printf("%d ", arr[i]);

return 0;

}

## **Delete Operation**

In the delete operation, the element to be deleted is searched using the [linear search](https://www.geeksforgeeks.org/linear-search/), and then the delete operation is performed followed by shifting the elements.



// C program to implement delete operation in a

// unsorted array

#include <stdio.h>

// To search a key to be deleted

int findElement(int arr[], int n, int key);

// Function to delete an element

int deleteElement(int arr[], int n, int key)

{

// Find position of element to be deleted

int pos = findElement(arr, n, key);

if (pos == -1) {

printf("Element not found");

return n;

}

// Deleting element

int i;

for (i = pos; i < n - 1; i++)

arr[i] = arr[i + 1];

return n - 1;

}

// Function to implement search operation

int findElement(int arr[], int n, int key)

{

int i;

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

if (arr[i] == key)

return i;

return -1;

}

// Driver's code

int main()

{

int i;

int arr[] = { 10, 50, 30, 40, 20 };

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

int key = 30;

printf("Array before deletion\n");

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

printf("%d ", arr[i]);

// Function call

n = deleteElement(arr, n, key);

printf("\nArray after deletion\n");

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

printf("%d ", arr[i]);

return 0;

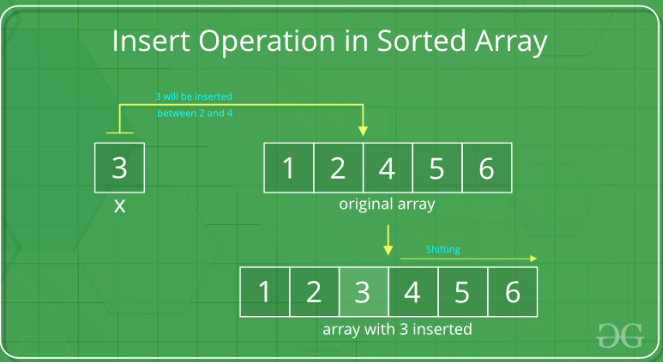
}

# **Insert, and Delete in an Sorted Array**

## **Insert Operation**

## **How to Insert in a Sorted Array?**

In a sorted array, a search operation is performed for the possible position of the given element by using[Binary search,](https://www.geeksforgeeks.org/binary-search/) and then an insert operation is performed followed by shifting the elements. And in an unsorted array, the insert operation is faster as compared to the sorted array because we don’t have to care about the position at which the element is placed.



// C program to implement insert operation in

// an sorted array.

#include <stdio.h>

// Inserts a key in arr[] of given capacity. n is current

// size of arr[]. This function returns n+1 if insertion

// is successful, else n.

int insertSorted(int arr[], int n, int key, int capacity)

{

// Cannot insert more elements if n is already

// more than or equal to capacity

if (n >= capacity)

return n;

int i;

for (i = n - 1; (i >= 0 && arr[i] > key); i--)

arr[i + 1] = arr[i];

arr[i + 1] = key;

return (n + 1);

}

/\* Driver code \*/

int main()

{

int arr[20] = { 12, 16, 20, 40, 50, 70 };

int capacity = sizeof(arr) / sizeof(arr[0]);

int n = 6;

int i, key = 26;

printf("\nBefore Insertion: ");

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

printf("%d ", arr[i]);

// Function call

n = insertSorted(arr, n, key, capacity);

printf("\nAfter Insertion: ");

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

printf("%d ", arr[i]);

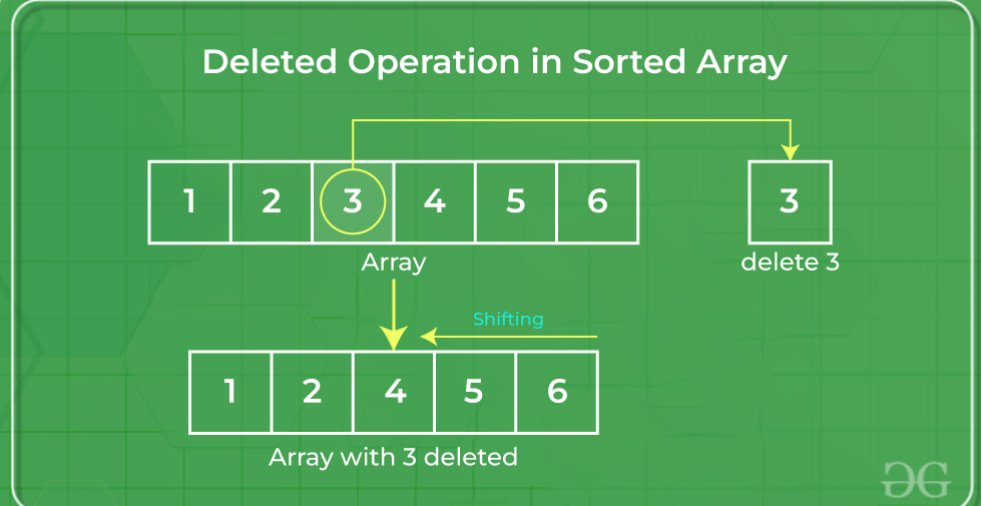
return 0;

}

## **Delete Operation**

## **How to Delete in a Sorted Array?**

In the delete operation, the element to be deleted is searched using binary search, and then the delete operation is performed followed by shifting the elements.



// C program to implement delete operation in a

// sorted array

#include <stdio.h>

// To search a key to be deleted

int binarySearch(int arr[], int low, int high, int key);

/\* Function to delete an element \*/

int deleteElement(int arr[], int n, int key)

{

// Find position of element to be deleted

int pos = binarySearch(arr, 0, n - 1, key);

if (pos == -1) {

printf("Element not found");

return n;

}

// Deleting element

int i;

for (i = pos; i < n - 1; i++)

arr[i] = arr[i + 1];

return n - 1;

}

int binarySearch(int arr[], int low, int high, int key)

{

if (high < low)

return -1;

int mid = (low + high) / 2;

if (key == arr[mid])

return mid;

if (key > arr[mid])

return binarySearch(arr, (mid + 1), high, key);

return binarySearch(arr, low, (mid - 1), key);

}

// Driver code

int main()

{

int i;

int arr[] = { 10, 20, 30, 40, 50 };

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

int key = 30;

printf("Array before deletion\n");

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

printf("%d ", arr[i]);

// Function call

n = deleteElement(arr, n, key);

printf("\n\nArray after deletion\n");

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

printf("%d ", arr[i]);

}