**Linear Search**

#include <bits/stdc++.h>

using namespace std;

int search(int arr[], int N, int x)

{

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

if (arr[i] == x)

return i;

return -1;

}

// Driver code

int main()

{

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

int x = 10;

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

// Function call

int result = search(arr, N, x);

(result == -1)

? cout << "Element is not present in array"

: cout << "Element is present at index " << result;

return 0;

}

**Iterative Binary Search**

#include <bits/stdc++.h>

using namespace std;

// An iterative binary search function.

int binarySearch(int arr[], int l, int r, int x)

{

while (l <= r) {

int m = l + (r - l) / 2;

// Check if x is present at mid

if (arr[m] == x)

return m;

// If x greater, ignore left half

if (arr[m] < x)

l = m + 1;

// If x is smaller, ignore right half

else

r = m - 1;

}

// If we reach here, then element was not present

return -1;

}

int main()

{

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

int x = 10;

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

int result = binarySearch(arr, 0, n - 1, x);

(result == -1)

? cout << "Element is not present in array"

: cout << "Element is present at index " << result;

return 0;

**Recursive Binary Search**

#include <bits/stdc++.h>

using namespace std;

// A recursive binary search function. It returns

// location of x in given array arr[l..r] is present,

// otherwise -1

int binarySearch(int arr[], int l, int r, int x)

{

if (r >= l) {

int mid = l + (r - l) / 2;

// If the element is present at the middle

// itself

if (arr[mid] == x)

return mid;

// If element is smaller than mid, then

// it can only be present in left subarray

if (arr[mid] > x)

return binarySearch(arr, l, mid - 1, x);

// Else the element can only be present

// in right subarray

return binarySearch(arr, mid + 1, r, x);

}

// We reach here when element is not

// present in array

return -1;

}

int main()

{

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

int x = 10;

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

int result = binarySearch(arr, 0, n - 1, x);

(result == -1)

? cout << "Element is not present in array"

: cout << "Element is present at index " << result;

return 0;

}