**Experiment No.10**

**Title:** Implementation of Binary Search

**Problem Statement:** Write a C++ program to search an element in an array using Binary search

**Algorithm:**

**Step 1** **:** Compare x with the middle element of the sorted array

**Step 2** **:** If x matches with middle element, we return the mid index

**Step 3 :** Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.

**Step 4:** Else (x is smaller) recur for the left half.

**Code:**

#include <iostream>

using namespace std;

// If found, location of x in the array is returned.

// otherwise -1 is returned.

int binarySearch(int arr[], int first, int last, int x)

{

if (last >= first) {

int mid = first + (last- first) / 2;

// If the element is present at the middle of the array

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, first, mid - 1, x);

// Else the element can only be present

// in right subarray

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

}

// If the element is not found.

return -1;

}

int main()

{

int arr[20],n,x,i;

cout<<"How many elements?";

cin>>n;

cout<<"\nEnter elements of the array\n";

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

cin>>arr[i];

cout<<"\nEnter element to search:";

cin>>x;

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

if (result == -1)

cout << "Element is not present in array.";

else

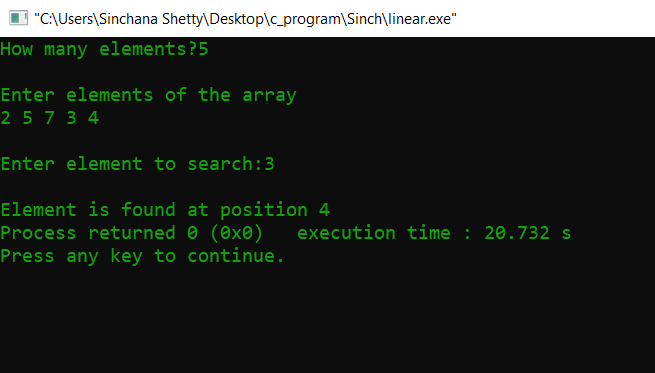
cout << "Element is present at index " <<

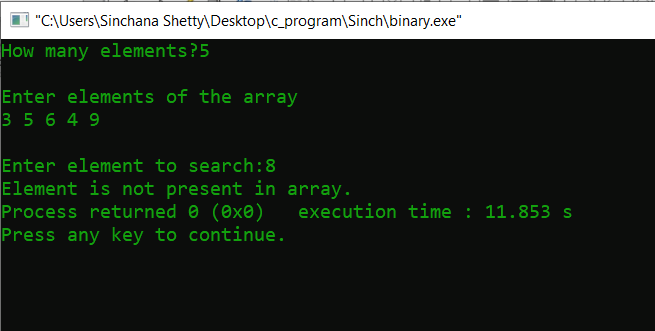
result;

return 0;

}

**Results:**





**Analysis(Limitations):**

A binary search however, cut down your search to half as soon as you find middle of a sorted list.

1. The middle element is looked to check if it is greater than or less than the value to be searched.
2. Accordingly, search is done to either half of the given list