**G. H. RAISONI COLLEGE OF ENGG., NAGPUR**

**(An Autonomous Institute)**

**Department of Computer Science & Engg.**



**Date: 31-08-2021**

**Practical Subject: Data Structures and Algorithms in C.**

**Session: 2021-22**

**Student Details:**

| **Roll Number** | 01 |
| --- | --- |
| **Name** | Anand Suralkar |
| **Semester** | 9th |
| **Section** | A |
| **Batch** | CSE |

**Practical Details: Practical Number-1;**

| Practical Aim | Write a program in C/C++ to implement Linear Search and Binary Search |
| --- | --- |
| Theory & Syntax | What is a Linear Search?  A linear search, also known as a sequential search, is a method of finding an element within a list. It checks each element of the list sequentially until a match is found or the whole list has been searched.  A simple approach to implement a linear search is  Begin with the leftmost element of arr[] and one by one compare x with each element.  If x matches with an element then return the index.  If x does not match with any of the elements then return -1.  Binary Search is a searching algorithm for finding an element's position in a sorted array.  How Binary Search Works?  For a Binary search to work, it is mandatory for the target array to be sorted. We shall learn the process of binary search with a pictorial example. The following is our sorted array and let us assume that we need to the location of value 31 using .    First, we shall determine half of the array by using this formula −  mid = low + (high - low) / 2  Here it is, 0 + (9 - 0 ) / 2 = 4 (integer value of 4.5). So, 4 is the mid of the array.    Now we compare the value stored at location 4, with the value being ed, i.e. 31. We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.  We change our low to mid + 1 and find the new mid value again.  low = mid + 1  mid = low + (high - low) / 2  Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.  The value stored at location 7 is not a match, rather it is more than what we are looking for. So, the value must be in the lower part from this location.  Hence, we calculate the mid again. This time it is 5.  We compare the value stored at location 5 with our target value. We find that it is a match.  We conclude that the target value 31 is stored at location 5.  binary search halves the able items and thus reduces the count of comparisons to be made to very less numbers. |
| Program | #include<stdio.h>  int main(){  int binarySearch(int arr[], int l, int r, int x);  int linearSearch(int arr[],int size);  int arr[5]={-12,0,10,15,23};  int size =sizeof (arr);  linearSearch(arr,size);  int x = 10;  int result = binarySearch(arr, 0, size - 1, x);  (result == -1) ? printf("Element is not present in array")  : printf("Element is present at index %d",result);  }  int linearSearch(int arr[],int size)  {  int a;  printf("\nEnter integer to linear search in array ");  scanf("%d",&a);  for (int j=0;j<size;j++){  if(a==arr[j]){ printf("match found at position %d",j); break; }if(j==4){  printf("no match found");}  }  return 0;  }  void bubbleSort(int arr[], int n)  {  int i, j;  void swap(int \* i,int \*j);  bool swapped;  for (i = 0; i < n-1; i++)  {  swapped = false;  for (j = 0; j < n-i-1; j++)  {  if (arr[j] > arr[j+1])  {  swap(&arr[j], &arr[j+1]);  swapped = true;  }  }  if (swapped == false)  break;  }  }  void swap(int \* i,int \*j){    int temp=\*i;  \*i=\*j;  \*j=temp;  }  int binarySearch(int arr[], int l, int r, int x)  {  if (r >= l) {  int mid = l + (r - l) / 2;  if (arr[mid] == x)  return mid;  if (arr[mid] > x)  return binarySearch(arr, l, mid - 1, x);  return binarySearch(arr, mid + 1, r, x);  }  return -1;  } |
| Output |  |
| Conclusion | Implemented and Executed program in C/C++ to implement Linear Search and Binary Search |