**WORKSHEET 1.1**

**1. Aim:**

Write a Program to Implement Linear Search and Binary Search.

**2. Problem Description:**

The main objective of this experiment is to demonstrate the working of Linear Search and Binary Search in an array.

**For Linear Search:**

**3.1. Algorithm**

Linear Search ( Array A, Value x)

Step 1: Set i to 1

Step 2: if i> n then go to step 7

Step 3: if A[i] = x then go to step 6

Step 4: Set i to i + 1

Step 5: Go to Step 2

Step 6: Print Element x Found at index i and go to step 8

Step 7: Print element not found

Step 8: Exit

**3.2. Computational Complexity:-**

**For Linear Search :**

Best Case :- O(1), because the element might be present at the 1st index.

Worst Case :- O(n), because the element might be present at the last index

Average Case :- O(n)

**3.3. Pseudo Code :-**

procedure linear\_search (list, value)

for each item in the list

if match item == value

return the item's location

end if

end for

end procedure

**3.4. Source Code:**

#include<stdio.h>

#include<stdlib.h>

int c=0;

int linearsearch(int \*, int,int);

int linearsearch(int \*a,int b,int n)

{

int i;

for(i=0;i<=n-1;i++)

{

if(\*(a+i)==b)

{

c=1;

break;

}

}

return i;

}

int main()

{

int \*a,i,n,m;

printf("Enter the size of an array: ");

scanf("%d",&n);

a=(int\*)malloc(n\*sizeof(int));

printf("Enter the elements of the array: ");

for(i=0;i<=n-1;i++)

{

scanf("%d",(a+i));

}

printf("Enter the number to be search: ");

scanf("%d",&m);

i=0;

i=linearsearch(a,m,n);

if(c==0)

printf("The number is not in the list\n");

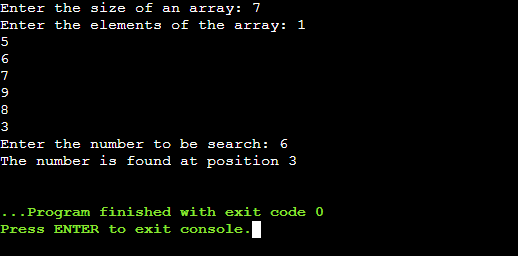
else

printf("The number is found at position %d\n",i+1);

return 0;

}

* 1. **Screenshot of Output:**

****

**For Binary Search:**

**4.1. Algorithm**

**Step 1:** [INITIALIZE] SET BEG = lower\_bound  
END = upper\_bound, POS = - 1

**Step 2:** Repeat Steps 3 and 4 while BEG <=END

**Step 3:** SET MID = (BEG + END)/2

**Step 4:** IF A[MID] = VAL  
SET POS = MID  
PRINT POS  
Go to Step 6  
ELSE IF A[MID] > VAL  
SET END = MID - 1  
ELSE  
SET BEG = MID + 1  
[END OF IF]  
[END OF LOOP]

**Step 5:** IF POS = -1  
PRINT "VALUE IS NOT PRESENT IN THE ARRAY"  
[END OF IF]

**Step 6:** EXIT

**4.2. Computational Complexity:-**

Best Case :- O(1), because the element might be present at the 1st index.

Worst Case :- O(log n), because the element might be present at the last index

Average Case :- O(log n)

**4.3. Pseudo Code :-**

Procedurebinary\_search

A ← sorted array

n ← size of array

x ←value to be searched

SetlowerBound=1

SetupperBound= n

while x not found

ifupperBound<lowerBound

EXIT: x does notexists.

setmidPoint=lowerBound+(upperBound-lowerBound)/2

if A[midPoint]< x

setlowerBound=midPoint+1

if A[midPoint]> x

setupperBound=midPoint-1

if A[midPoint]= x

EXIT: x found at location midPoint

endwhile

end procedure

**4.4. Source Code:**

#include <iostream>

using namespace std;

int main()

{

int a[50] = { 10,20,30,40,50,90,110 }, i, x, beg=0, end=7, mid, n = 7, flag=0;

cout<< "THIS IS BINARY SEARCH\n\n";

cout<< "Array: ";

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

{

cout<< a[i]<<" ";

}

cout<< "\nEnter the element you want to search: ";

cin>> x;

while (beg <= end)

{

mid = (beg + end) / 2;

if (a[mid] == x)

{

cout<< "Element found at position " << mid + 1 << "\n\n";

return 0;

}

else

{

if (x > a[mid])

{

beg = mid + 1;

}

else

end = mid - 1;

}

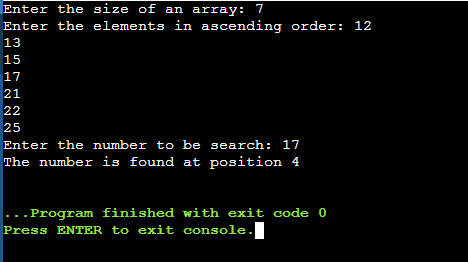
}

cout<< "Element not found!\n\n";

return 0;

}

* 1. **Screenshot of Output:**



1. **Learning & Outcomes:**

* Learned about the linear search Algorithm, how it works, How much is the time complexity.
* Learned about the Binary Search Algorithm, how it works and how much is the time complexity.
* Learned to create dynamic array using pointers.