Experiment 4
Implementation of Linear Search Algorithm

Date: 06-09-2020

Aim: To find an element in a give array using the linear search algorithm

Data Structures Used: Arrays
Operations Used : Comparison

Algorithm:

Input: An integer Array A[L...U] //L and U are the lower and upper bounds of the array. An integer ‘q’ which is to be searched

Output: An integer value KEY which is the index of the element ‘q’, -1 if the element doesn’t exist in the array

Steps:

```
Step 1 : Start
Step 2 : i ← L
Step 3 : while i <= U
            Step 1: if(A[i] = q)
                        Step 1: End while
            Step 2: End if
Step 4 : End while
Step 5: KEY ← i
Step 5 : if KEY=n
            Step 1: Print “The element q is not in the array”
Step 6 : else
            Step 1:  Print “The element q is at position KEY+1 and at index KEY+1”
Step 7 : End if
Step 8 : Stop
```

Description of the Algorithm:
The Linear search will search the array for the element q in the array starting from the first element until the element to be searched is found or the last element is reached. If the control variable reaches the last element and it is not the required element then the program will print an error message. If the element is found then the program will return the posion and the index value of the element in the array
The Best case time complexity is O(1)
The Worst case time complexity is O(n)

Result: The Program was successfully compiled and the required output was obtained

Program:

```
/*Implementation of linear Search*/

#include<stdio.h>

/*Linear search funtion:
 * Takes in an array A and a value query to search for in the array
 * Returns the position of the element or prints an error message if the element
is not found
 */
int linearSearch(int* A,int n,int query){
    int KEY;
    int i;
    for(i=0;i<n;i++){
        if(A[i]==query){
            break;
        }
    }
    if(i==n){
        return -1;
    }
    else{
        return i;
    }
}

void getValues(int* A, int n){
    for(int i = 0; i<n; i++){
        scanf("%d%*c",A+i);
    }
}

void main(){
    int n;
    int arr[100];
    int elem;
    printf("Enter the size of the array: ");
    scanf("%d%*c",&n);
    printf("Enter the elements of the array: ");
    getValues(arr,n);
    printf("Enter the element to be searched: ");
    scanf("%d%*c",&elem);

    int KEY = linearSearch(arr,n,elem);
    if(KEY<0){
        printf("%d doesn't exist in the array\n",elem);
    }
    else{
        printf("First occurrence of %d is in position %d and index %d\
n",elem,KEY+1,KEY);
    }

}
```

Sample Input 1:
```
5
59 57 41 32 81
32
```

Sample Output 1:
```
Enter the size of the array: 5
Enter the elements of the array: 59 57 41 32 81
Enter the element to be searched: 32
First occurrence of 32 is in position 4 and index 3
```

Sample Input 2:
```
5
59 57 41 32 81
69
```

Sample Output 2:
```
Enter the size of the array: 5
Enter the elements of the array: 59 57 41 32 81
Enter the element to be searched: 69
69 doesn't exist in the array
```

Experiment 5
Implementation of Binary Search Algorithm

Date: 06-09-2020

Aim: To find an element in a give array using the Binary search algorithm

Data Structures Used: Arrays
Operations Used : Comparison

Algorithm:

Input: A sorted integer Array A[L...U] //L and U are the lower and upper bounds of the array. An integer ‘q’ which is to be searched

Output: An integer value KEY which is the index of the element ‘q’, -1 if the element doesn’t exist in the array

Steps:

```
Step 1 : Start
Step 2 : beg ← L                        // Variable initially pointing to the first index
Step 3 : last ← U                              // Variable initially pointing to the second index
Step 4 : KEY ←  -1
Step 5 : while beg<=last
         Step 1 : mid ← (last+beg)/2
         Step 2 : if A[mid] > q
                    Step 1: last ← mid-1
         Step 3 : else if A[mid] < q
                    Step 1: beg ← mid+1
         Step 4 : else
                   Step 1: KEY ← mid
         Step 5 :End If
Step 6 : if KEY = -1
         Step 1 : Print “The element is not in the array”
Step 7 : else
         Step 1 : Print “The element is at index KEY and position KEY+1”
Step 8 : End If
Step 9 : Stop
```

Description of the Algorithm:
The Binary search checks if the middle element is the element to be searched (q), if it is not then it checks if the middle element is greater than q if it is then it searches only the lower half of the array, hence it works only on sorted arrays. If the middle element is smaller than q it checks the upper half of the array.
The Best case time complexity is O(1). When the element to be searched is the middle element.
The Worst case time complexity is O(log(n)).

Result: The Program was successfully compiled and the required output was obtained.

Program:

```
/* Implementation of binary search
 */
#include<stdio.h>

int binarySearch(int *A,int n, int elem){
    int beg = 0;
    int last = n-1;
    int mid;
    while(beg<=last){
        mid = (last+beg)/2;
        if(A[mid]>elem){
            last=mid-1;
        }
        else if(A[mid]<elem){
            beg = mid+1;
        }
        else{
            return mid;
        }
    }
    return -1;
}

void inputArray(int *A, int n){
    for(int i = 0;i<n;i++){
        scanf("%d%*c",A+i);
    }
}

void main(){
    int n,elem;
    int arr[100];
    printf("Enter the number of elements in the array: ");
    scanf("%d%*c",&n);
    printf("Enter the elements of the array: ");
    inputArray(arr,n);

    printf("Enter the element to search for: ");
    scanf("%d%*c",&elem);

    int KEY = binarySearch(arr,n,elem);
    if(KEY<0){
        printf("%d is not in the array\n",elem);
    }
    else{
        printf("%d is found at position %d and at index %d in the array\
n",elem,KEY+1,KEY);
    }
}
```

Sample Input 1:

```
5
32 41 57 59 81
81
```

Sample Output 1:

```
Enter the number of elements in the array: 5
Enter the elements of the array: 32 41 57 59 81
Enter the element to search for: 81
81 is found at position 5 and at index 4 in the array
```

Sample Input 2:

```
5
32 41 57 59 81
23
```

Sample Output 2:

```
Enter the number of elements in the array: 5
Enter the elements of the array: 32 41 57 59 81
Enter the element to search for: 23
23 is not in the array
```