CSE 114 – Data Structures and Algorithms Lab

Lecture 1

Experiment No: **01**

Experiment Title: **Array: Traverse, Insertion, Deletion; Searching: Linear Search, Binary Search**

1. Array: Traverse

Given an integer array of size **N**, the task is to traverse and print the elements in the array.

Examples:

Input: *arr[] = {2, -1, 5, 6, 0, -3}*

Output: *2 -1 5 6 0 -3*

Input: *arr[] = {4, 0, -2, -9, -7, 1}*

Output: *4 0 -2 -9 -7 1*

Pseudocode

1. Start a loop from **0** to **N-1**, where **N** is the size of array.
2. Access every element of array
3. Print the elements.

C program to traverse the array

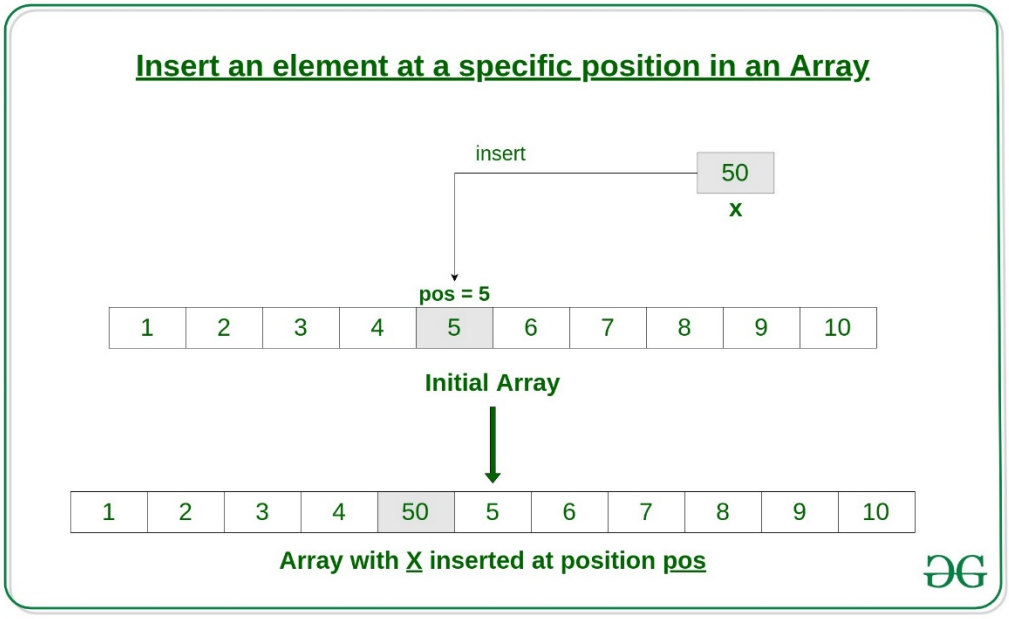
```
#include <stdio.h>
void printArray(int* arr, int n)
{
        int i;
        printf("Array: ");
        for (i = 0; i < n; i++) {
                printf("%d ", arr[i]);
        }
        printf("\n");
}

int main()
{
        int arr[] = { 2, -1, 5, 6, 0, -3 };
        int n = sizeof(arr) / sizeof(arr[0]);
        printArray(arr, n);
```

```
    return 0;
}
```

2. Array: Insertion

Given an integer array of size **N**, the task is to traverse and print the elements in the array.

Examples:



Steps to solve the problem:

1. First get the element to be inserted, say x
2. Then get the position at which this element is to be inserted, say pos
3. Then shift the array elements from this position to one position forward (towards right), and do this for all the other elements next to pos.
4. Insert the element x now at the position pos, as this is now empty.

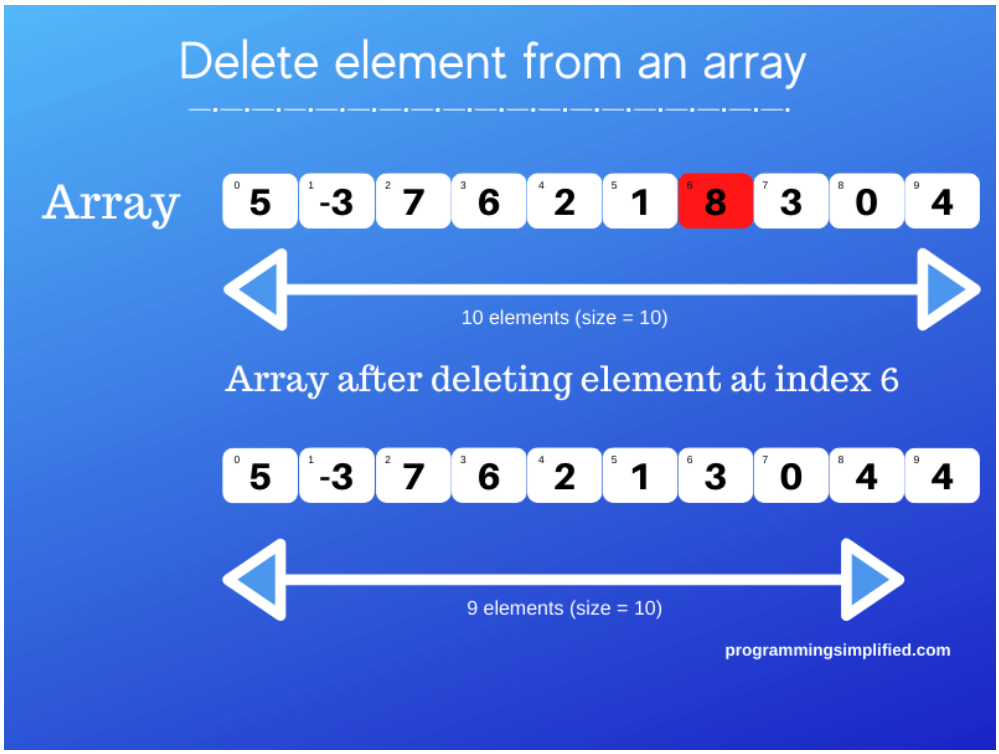
// C Program to Insert an element at a specific position in an Array

```
#include <stdio.h>

int main()

{
```

```
int arr[100];
int i, x, pos, n = 10; // initial array of size 10
for (i = 0; i < 10; i++)
        arr[i] = i + 1;
// print the original array
for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
printf("\n");
// element to be inserted
x = 50;
// position at which element is to be inserted
pos = 5;
// increase the size by 1
n++;
// shift elements forward
for (i = n - 1; i >= pos; i--)
        arr[i] = arr[i - 1];
// insert x at pos
arr[pos - 1] = x;
// print the updated array
for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
printf("\n");
```

```
    return 0;
}
```

3. Array: Deletion



Step by step descriptive logic to remove element from array.

1. Move to the specified location which you want to remove in given array.
2. Copy the next element to the current element of array.
3. Repeat above steps till last element of array.
4. Finally decrement the size of array by one.

C program to delete an element from array at specified position

```
#include <stdio.h>
#define MAX_SIZE 100
int main()
```

```
{
    int arr[MAX_SIZE];
    int i, size, pos;
    /* Input size and element in array */
    printf("Enter size of the array : ");
    scanf("%d", &size);
    printf("Enter elements in array : ");
    for(i=0; i<size; i++)
    {
        scanf("%d", &arr[i]);
    }
    /* Input element position to delete */
    printf("Enter the element position to delete : ");
    scanf("%d", &pos);
    /* Invalid delete position */
    if(pos < 0 || pos > size)
    {
        printf("Invalid position! Please enter position between 1 to %d", size);
    }
    else
    {
        /* Copy next element value to current element */
        for(i=pos-1; i<size-1; i++)
        {
            arr[i] = arr[i + 1];
        }
        /* Decrement array size by 1 */
        size--;
        /* Print array after deletion */
        printf("\nElements of array after delete are : ");
        for(i=0; i<size; i++)
```

```
        {
            printf("%d\t", arr[i]);
        }
    }
    return 0;
}
```

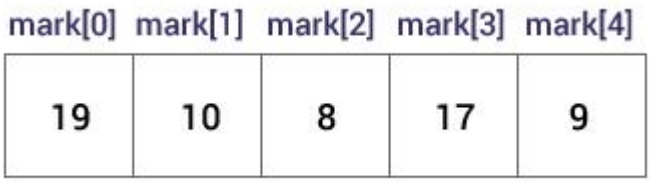
Linear Search

Problem Statement: Given an array of numbers and a key value. You need to find the key from the given array using linear search algorithm. Array could be any size and need to take input from keyboard.

Sample Input:

Given Array:

mark[5] = {19, 10, 8, 17, 9}



Key = 8

Sample Output: Print “Yes” if key=8 found in mark[2], Otherwise Print “No”.

Pseudo code:

```
procedurelinear_search(list, value)

for each item in the list

if match item == value

return the item's location

end if

end for
```

```
end procedure
```

// C Program to implement the linear search

```
#include <stdio.h>
int search(int array[], int n, int x) {
  // Going through array sequencially
  for (int i = 0; i < n; i++)
    if (array[i] == x)
      return i;
  return -1;
}

int main() {
  int array[] = {2, 4, 0, 1, 9};
  int x = 1;
  int n = sizeof(array) / sizeof(array[0]);
  int result = search(array, n, x);
  (result == -1) ? printf("Element not found") : printf("Element found at index: %d", result);
}
```

Binary Search

Problem Statement: Given a sorted list of a [] of n elements, search a given element x in list.

a. Search a sorted list by repeatedly dividing the search interval in half. Begin with an interval covering the whole list.

b. If the search key is less than the item in the middle item, then narrow the interval to the lower half. Otherwise narrow it to the upper half.

c. Repeat the procedure until the value is found or the interval is empty.

Sample Input:

Consider a sorted list a[] with 9 elements and the search key is 31.

0	1	2	3	4	5	6	7	8
11	23	31	33	65	68	71	89	100

Sample Output: Let the search key = 31. First low = 0, high = 8, mid = (low + high) = 4

a[mid] = 65 is the centre element, but 65 > 31.

So now high = mid - 1= 4 - 1 = 3, low = 0, mid = (0 + 3) / 2 = 1 10

a[mid] = a[1] = 23, but 23 < 31. Again low = mid +1 = 1 +1 =2, high = 3, mid = (2 + 3) /2 = 2

a[mid] = a[2] = 31 which is the search key, so the search is successful.

Pseudo code:

```
Algorithm binsrch (a[], n, x)
{ // a[1:n] is an array of n elements
    low = 1;
    high = n;
    while (low < high)
      do { mid = (low + high)/2 ;
     if (x < a[mid])
        then high = mid – 1;
    else if (x > a[mid])
        then low = mid + 1;
      else return mid;
    }
    return 0;
    }
```

C program to implement the binary search

```
#include <stdio.h>
int main()
```

```
{
int i, low, high, mid, n, key, array[100];
printf("Enter number of elementsn");
scanf("%d",&n);
printf("Enter %d integersn", n);
for(i = 0; i < n; i++)
scanf("%d",&array[i]);
printf("Enter value to findn");
scanf("%d", &key);
low = 0;
high = n - 1;
mid = (low+high)/2;
while (low <= high) {
if(array[mid] < key)
low = mid + 1;
else if (array[mid] == key) {
printf("%d found at location %d.n", key, mid+1);
break;
}
else
high = mid - 1;
mid = (low + high)/2;
}
if(low > high)
printf("Not found! %d isn't present in the list.n", key);
return 0;
}
```