Fundamentals of
Data Structures using C

List-Array Based Implementation

B.Bhuvaneswaran, AP (SG) / CSE

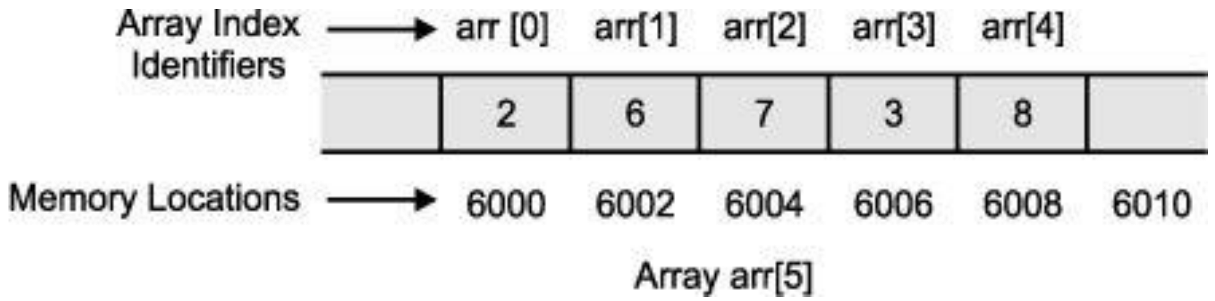
9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI
ENGINEERING COLLEGE

Introduction

- Array is a collection of specific number of data stored in consecutive memory locations.



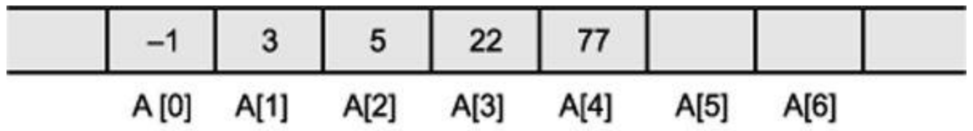
Operations Array

- Insert
- Delete
- Traversal
- Sorting
- Searching

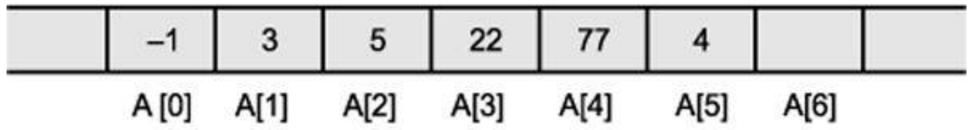
Insertion

- Insertion is the task of adding an element into an existing array.
- If an element is to be inserted at the end of the array, then it can be simply achieved by storing the new element one position to the right of the last element.
- Alternatively, if an element is required to be inserted at the middle, then this will require all the subsequent elements to be moved one place to the right.

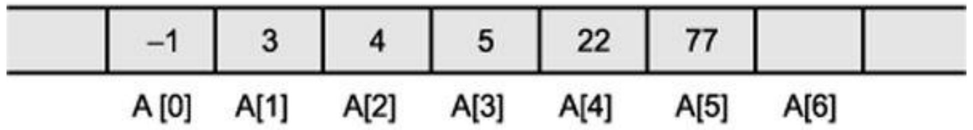
Example



Initial Array A



Array contents after element 4 is inserted at the end



Array contents after element 4 is inserted at index location 2

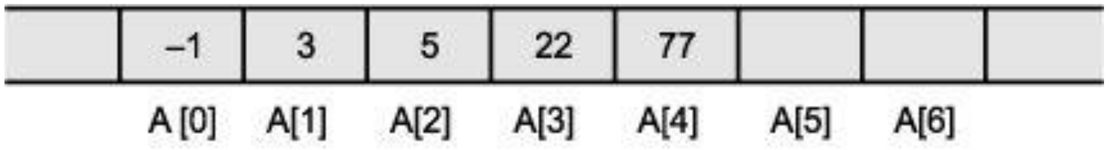
Routine-Insert an Element in an Array

```
void Insert(int a[], int p, int e)
{
        int i;
        for(i = n; i >= p; i--)
                a[i + 1] = a[i];
        a[p] = e;
        n = n + 1;
}
```

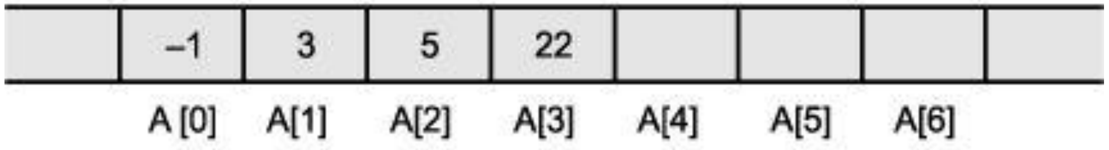
Deletion

- Deletion is the task of removing an element from the array.
- The deletion of element from the end is quite simple and can be achieved by mere updation of index identifier.
- However, to remove an element from the middle, one must move all the elements present to the right of the point of deletion, one position to the left.

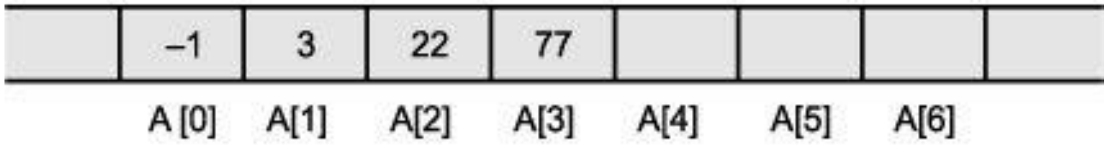
Example



Initial Array A



Array contents after element is deleted from the end



Array contents after an element is deleted from index location 2

Routine-Delete an Element from an Array

```
void Delete(int a[], int p)
{
        int i;
        for(i = p; i < n; i++)
                a[i] = a[i + 1];
        n = n - 1;
}
```

Traversal

- While working with arrays, it is often required to access the array elements; that is, reading values from the array.
- This is achieved with the help of array traversal.
- It involves visiting the array elements and storing or retrieving values from it.
- Some of the typical situations where array traversal may be required are:
  - Printing array elements
  - Searching an element in the array
  - Sorting an array

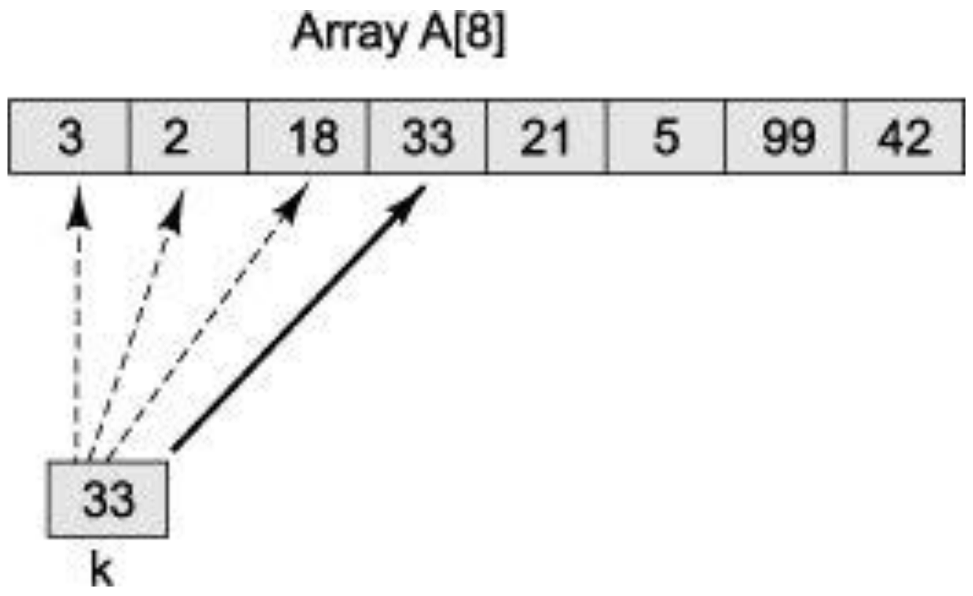
Routine-to Traverse an Array

```
void Traverse(int a[])
{
        int i;
        for(i = 0; i < n; i++)
                printf("%d\t", a[i]);
}
```

Searching

- Searching is the process of traversing an array to find out if a specific element is present in the array or not.
- If the search is successful, the index location of the element is returned.

Example



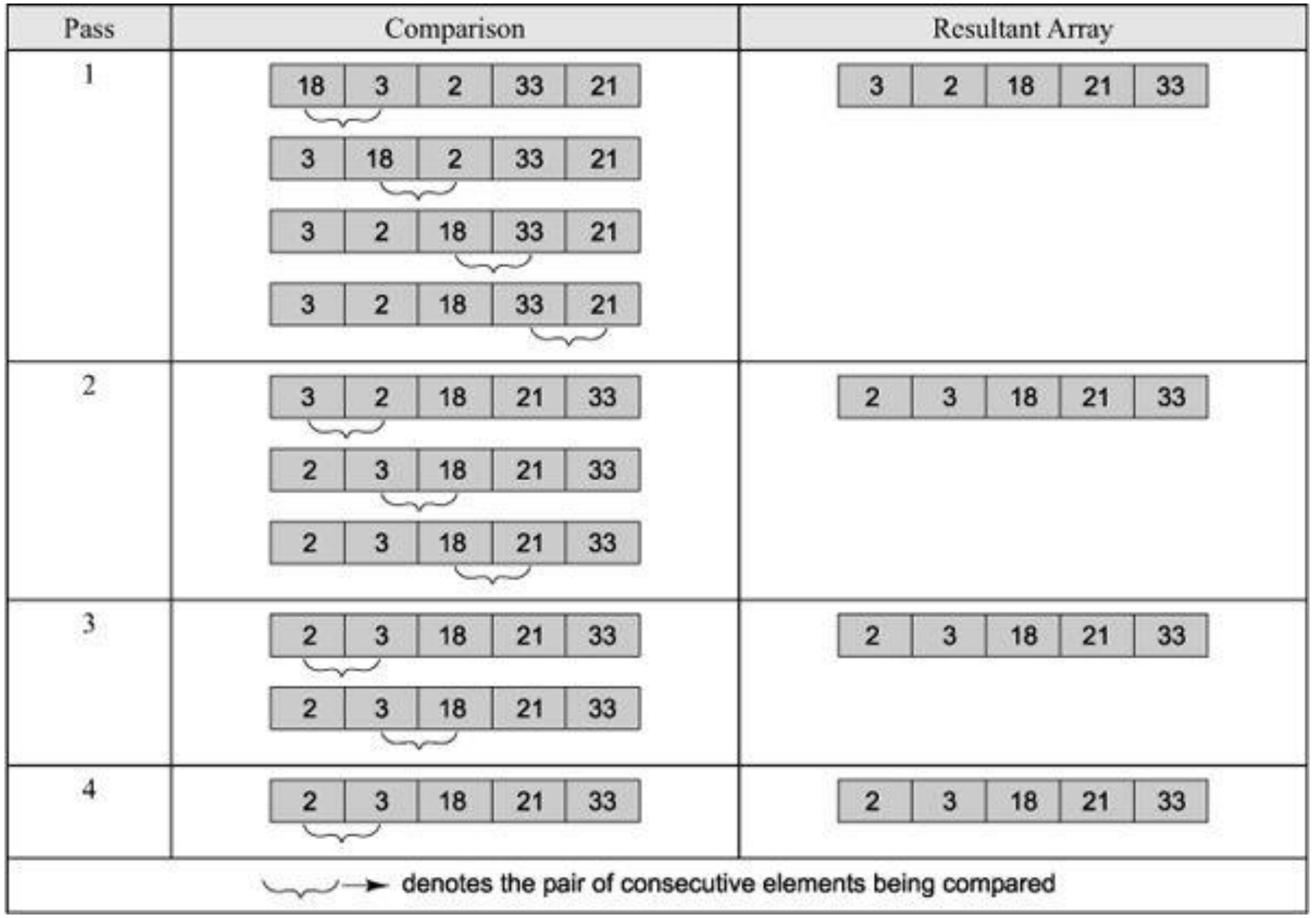
Routine-Search an element

```
void Search(int a[], int e)
{
        int i, flag = 0;
        for(i = 0; i < n; i++)
        {
                if(a[i] == e)
                {
                        flag = 1;
                        break;
                }
        }
        if(flag == 1)
                printf("Successful. Element %d is at location %d", e, i);
        else
                printf("Unsuccessful.");
}
```

Sorting

- The sorting operation arranges the elements of an array in a specific order or sequence.
- Sorting involves comparing the array elements with each other and shuffling them until all the elements are sorted.

Example



Pass	Comparison	Resultant Array
1	$\underbrace{18\ \ 3}\ \ 2\ \ 33\ \ 21$ $3\ \ \underbrace{18\ \ 2}\ \ 33\ \ 21$ $3\ \ 2\ \ \underbrace{18\ \ 33}\ \ 21$ $3\ \ 2\ \ 18\ \ \underbrace{33\ \ 21}$	3 2 18 21 33
2	$\underbrace{3\ \ 2}\ \ 18\ \ 21\ \ 33$ $2\ \ \underbrace{3\ \ 18}\ \ 21\ \ 33$ $2\ \ 3\ \ \underbrace{18\ \ 21}\ \ 33$	2 3 18 21 33
3	$\underbrace{2\ \ 3}\ \ 18\ \ 21\ \ 33$ $2\ \ \underbrace{3\ \ 18}\ \ 21\ \ 33$	2 3 18 21 33
4	$\underbrace{2\ \ 3}\ \ 18\ \ 21\ \ 33$	2 3 18 21 33

$\underbrace{\quad} \rightarrow$ denotes the pair of consecutive elements being compared

Routine-Sort the elements

```
void Sort(int a[])
{
        int i, j, t;
        for (i = 0; i < n - 1; i++)
        {
                for (j = 0; j < n - 1 - i; j++)
                {
                        if (a[j] > a[j + 1])
                        {
                                t = a[j];
                                a[j] = a[j + 1];
                                a[j + 1] = t;
                        }
                }
        }
}
```

Queries?

Thank You!