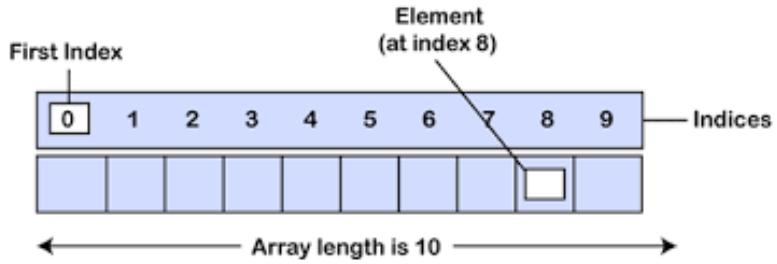
**UNIT I – INTRODUCTION**

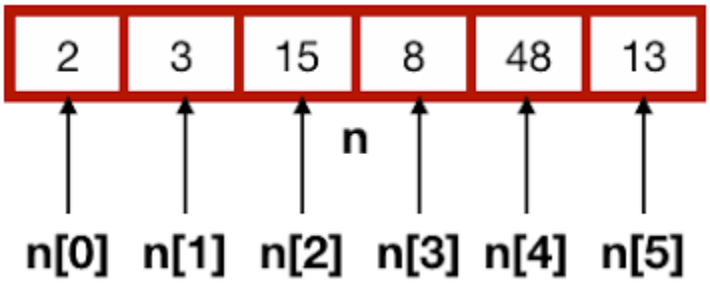
Data structures - Abstract data types - Primitive data structures- – Performance analysis – Space complexity – Time complexity – Asymptotic notations – Performance measurement – Array as an abstract data type – Polynomial as an abstract data type – Sparse matrix abstract data type – String abstract data type.

---

## ARRAY AS AN ABSTRACT DATA TYPE

Array is a collection of a specific number of the same type of data stored in consecutive memory locations. Array is a static data structure i.e., the memory should be allocated in advance and the size is fixed. This will waste the memory space when used space is less than the allocated space. Insertion and Deletion operation are expensive as it requires more data movements Find and Print list operations takes constant time.




The basic operations performed on a list of elements are

a. Creation of List.

b. Insertion of data in the List

c. Deletion of data from the List

d. Display all data's in the List

e. Searching for a data in the list

### Declaration of Array:

```
#define maxsize 10
```

```
int list[maxsize], n ;
```

## Create Operation:

Create operation is used to create the list with „ n „ number of elements .If n exceeds the array's maxsize, then elements cannot be inserted into the list. Otherwise the array elements are stored in the consecutive array locations (i.e.) list [0], list [1] and so on.

```
void create ()
{
        int i;
        printf("\nEnter the number of elements to be added in the list:\t");
        scanf("%d",&n);
        printf("\nEnter the array elements:\t");
        for(i=0;i<n;i++)
                scanf("%d",&list[i]);
}
```

If n=6, the output of creation is as follows:

list[6]

## Insert Operation:

Insert operation is used to insert an element at a particular position in the existing list. Inserting the element in the last position of an array is easy. But inserting the element at a particular position in an array is quite difficult since it involves all the subsequent elements to be shifted one position to the right.

## Routine to insert an element in the array:

```
void insert()
{
        int i,data,pos;
        printf("\nEnter the data to be inserted:\t");
        scanf("%d",&data);
        printf("\nEnter the position at which element to be inserted:\t");
        scanf("%d",&pos);
        if (pos>=n)
                printf (“Array overflow”);
        for(i = n-1 ; i >= pos-1 ; i--)
                list[i+1] = list[i];
        list[pos-1] = data;
```

```
        n=n+1;
        display();
}
```

Consider an array with 5 elements [max elements = 10]

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 | | | | | |

If data 15 is to be inserted in the 2nd position then 50 has to be moved to next index position, 40 has to be moved to 50 position, 30 has to be moved to 40 position and 20 has to be moved to 30 position.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | | 20 | 30 | 40 | 50 | | | | |

After these four data movements, 15 is inserted in the 2nd position of the array.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 15 | 20 | 30 | 40 | 50 | | | | |

## Deletion Operation:

Deletion is the process of removing an element from the array at any position. Deleting an element from the end is easy. If an element is to be deleted from any particular position ,it requires all subsequent elements from that position to be shifted one position towards left.

## Routine to delete an element in the array:

```
void delete( )
{
        int i, pos ;
        printf("\nEnter the position of the data to be deleted:\t");
        scanf("%d",&pos);
```

```
        printf("\nThe data deleted is:\t %d", list[pos-1]);
        for(i=pos-1;i<n-1;i++)
                list[i]=list[i+1];
        n=n-1;
        display();
}
```

Consider an array with 5 elements [max elements = 10]

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 | | | | | |

If data 20 is to be deleted from the array, then 30 has to be moved to data 20 position, 40 has to be moved to data 30 position and 50 has to be moved to data 40 position.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 | | | | | |

After these 3 data movements, data 20 is deleted from the 2nd position of the array.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 30 | 40 | 50 | | | | | | |

## Display Operation/Traversing a list:

Traversal is the process of visiting the elements in an array. Display( ) operation is used to display all the elements stored in the list. The elements are stored from the index 0 to n - 1. Using a for loop, the elements in the list are viewed.

## Routine to traverse/display elements of the array:

```
void display( )
{
        int i;
        printf("\n**********Elements in the array**********\n");
        for(i=0;i<n;i++)
                printf("%d\t",list[i]);
}
```

## Search Operation:

Search( ) operation is used to determine whether a particular element is present in the list or not. Input the search element to be checked in the list.

## Routine to search an element in the array:

```
void search( )
{
	int search,i,count = 0;
	printf("\nEnter the element to be searched:\t");
	scanf("%d",&search);
	for(i=0;i<n;i++)
	{
		if(search == list[i])
		count++;
	}
	if(count==0)
		printf("\nElement not present in the list");
	else printf("\nElement present in the list");
}
```

## Program for array implementation of List

```
#include<stdio.h>
#include<conio.h>
#define maxsize 10
int list[maxsize],n;
void create();
void insert();
void delete();
void display();
void search();
void main()
{
	int choice;
	clrscr();
	do
```

```
	{
		printf("\n Array Implementation of List\n");
		printf("\t1.create\n");
		printf("\t2.Insert\n");
		printf("\t3.Delete\n");
		printf("\t4.Display\n");
		printf("\t5.Search\n");
		printf("\t6.Exit\n");
		printf("\nEnter your choice:\t");
		scanf("%d",&choice);
		switch(choice)
		{
		case 1:
			create();
			break;
		case 2:
			insert();
			break;
		case 3:
			delete();
			break;
		case 4:
			display();
			break;
		case 5:
			search();
			break;
		case 6:
			exit(1);
		default:
			printf("\nEnter option between 1 - 6\n");
			break;
		}
	}while(choice<7);
}
	void create ()
	{
```

```
int i;
printf("\nEnter the number of elements to be added in the list:\t");
scanf("%d",&n);
printf("\nEnter the array elements:\t");
for(i=0;i<n;i++)
        scanf("%d",&list[i]);
}
void insert()
{
int i,data,pos;
printf("\nEnter the data to be inserted:\t");
scanf("%d",&data);
printf("\nEnter the position at which element to be inserted:\t");
scanf("%d",&pos);
if (pos==n)
        printf (“Array overflow”);
for(i = n-1 ; i >= pos-1 ; i--)
        list[i+1] = list[i];
list[pos-1] = data;
n=n+1;
display();
}
void delete( )
{
int i, pos ;
printf("\nEnter the position of the data to be deleted:\t");
scanf("%d",&pos);
printf("\nThe data deleted is:\t %d", list[pos-1]);
for(i=pos-1;i<n-1;i++)
        list[i]=list[i+1];
n=n-1;
display();
}
void search( )
{
int search,i,count = 0;
printf("\nEnter the element to be searched:\t");
```

```
scanf("%d",&search);
for(i=0;i<n;i++)
{
        if(search == list[i])
        count++;
}
if(count==0)
        printf("\nElement not present in the list");
else printf("\nElement present in the list");
}
void display( )
{
int i;
printf("\n**********Elements in the array**********\n");
for(i=0;i<n;i++)
        printf("%d\t",list[i]);
}
```

---