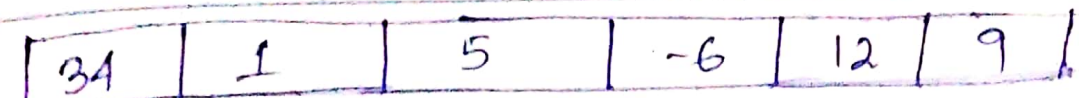
Data Structure

Arrays

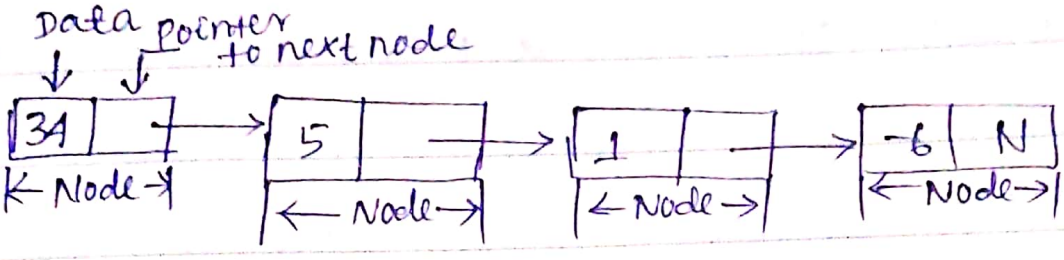
- Data structures are classified into two categories - linear and nonlinear.
- The elements in a linear data structure form a sequence, where as elements in a nonlinear data structure do not.
- There are two ways of representing linear data structures in memory - Array based list (simply called arrays) and linked list.
- In array the linear relationship between elements is established by storing its elements in sequential memory locations.
- In linked list the linear relationship is established through pointers or links.
- In a linked list each node contains the data and the address of the next node.
- Fig(1) and fig(2) shows the representation of an array and a linkedlist.

fig(1)



(a) Array of 6 integers

fig (2)



(b) Linked list of 4 integers

- Arrays are useful when the member of elements to be stored is fixed. They are easy to traverse, search and sort.
- on the other hand, linked lists are useful when number of data items in the collection is likely to vary.
- Linked lists are difficult to maintain as compared to an array.

Arrays

- An array is a finite collection of similar elements stored in ~~adjacent~~ adjacent memory location.
- An array containing n number of elements is referenced using an index that varies from 0 to n-1.

For Ex.

- The elements of an array arr[n] containing n elements are denoted by arr[0], arr[1], arr[2], - -- arr[n-1]; where 0 is the lower bound of the array, n-1 is the upper bound of the array and 0, 1, 2 etc. are indices of the array.
- A sample arrangement of array element is shown in below figure.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
34	1	5	-6	12	9

Elements in array with their indices

- There are several operations that can be performed on an array. These operations are listed below.

Operation	Description
Traversal	Processing each element in an array
Search	Finding the location of an element with a given value
Insertion	Adding a new element to an array
Deletion	Removing an element from an array
Sorting	Organizing the array elements in some order
Merging	Combining two arrays into a single array
Reversing	Reversing the elements of an array

Program - 1

* Implementation of various array operations.

```
#include <stdio.h>
#define MAX 5
void insert (int *, int pos, int num) ;
void del (int *, int pos) ;
void reverse (int *) ;
void display (int *) ;
void search (int *, int num) ;
int main()
{
int arr[5] ;
insert (arr, 1, 11) ;
insert (arr, 2, 12) ;
insert (arr, 3, 13) ;
insert (arr, 4, 14) ;
insert (arr, 5, 15) ;
printf ("Elements of Array :\n") ;
display (arr) ;
del (arr, 5) ;
del (arr, 2) ;
```

```
printf("After deletion:\n");
display(arr);
insert(arr, 2, 222);
insert(arr, 5, 555);
Printf("After insertion:\n");
display(arr);
reverse(arr);
Printf("After reversing:\n");
display(arr);
Search(arr, 222);
display(arr, 222);
search(arr, 666);
return 0;
}
/* insert an element num at given position
                                    POS */

void insert(int *arr, int pos, int num)
{
    /* Shift elements to right */
    int i;
for(i=MAX-1; i>=POS; i--)
arr[i] = arr[i-1];
    arr[i] = num;
}
```

```
/* deletes an element from the given position Pos */
void del (int *arr, int Pos)
{ /* SKIP to the desired position */
  int i;
  for (i=Pos; i<MAX; i++)
    arr[i-1] = arr[i];
  arr[i-1] = 0;
}
/* reverse the entire array */
void reverse (int * arr)
{ int i;
  for(i=0; i<MAX/2; i++)
    { int temp = arr[i];
      arr[i] = arr[MAX-1-i];
      arr[MAX-1-i] = temp;
    }
}
/* Searches array for a given element neem */
void search (int *arr, int neem)
{ int i;
  for(i=0; i<MAX; i++)
    { if(arr[i] == neem)
      {
```

```
        printf("Element %d is at %dth position\n",
                                        neem, i+1);
        return;
        }
    }
    printf("Element %d is absent\n", neem);
}
/* display Contents of a array */
    void display(int *arr)
    { int i;
      for(i=0; i<MAX; i++)
      { printf("%d\t", arr[i]);
        printf("\n");
    }
```

Output:

Elements of Array:

11 12 13 14 15

After deletion:

11 13 14 0 0

After insertion:

11 222 13 14 555

After reversing :

555 14 13 222 11

Element 222 is at 4th position

Element 666 is absent

- In this program we have created an array arr which contains 5 ints. We have passed the base address ~~to~~ of this array to functions insert(), del(), display(), reverse() and search() to perform different operations on the array.

- The insert() function takes two arguements, the position pos at which the new number has to be inserted and the number num that has to be inserted. In this function, firstly through a loop, we have shifted the numbers from the specified position, one place to the right of their existing position Then we have placed the number num at position pos.

- The del() function deletes the element present at the given position pos. For this we have shifted the numbers placed after the position from where the number is to be deleted, one place to the left of their existing positions. The number at position pos is then overwritten with 0.

- In reverse() function we have reversed the entire array by swapping the elements

arr[0] with arr[4], arr[1] with arr[3]

and so on.

Note that swapping should continue for MAX/2 times only, irrespective of whether MAX is odd or even.

- The search() function searches the array for the specified number. For this the comparision is carried out until either the list is exhausted or a match is found. If the match is not found then the function displays the relevant message.

- In the display( ) function, the entire array is traversed to display the elements of the array.