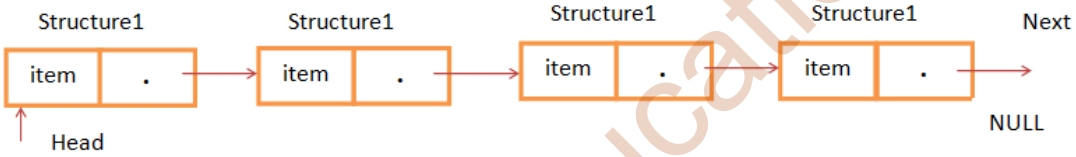
Linked Lists

What is a Linked List?

Linked list is a data structure that is used for storing collections of data. It is a very common data structure to create tree, graph and other abstract data types. Linked list has the following properties.

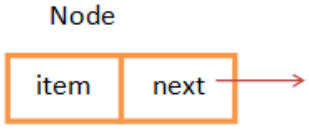
- Linked list is series of nodes: Linked list comprise of series of nodes in which each node have link to next node to form chain shown in below figure.
- Each successive elements in node connected by pointers.
- Last element points to NULL.
- List can grow and shrink during execution of program.
- It doesn't waste memory but takes extra memory for pointers.



- Here each structure of list is called as node, it consists of two fields. One is item or data field and other is containing address of next field.
- A linked list is a collection of structures ordered not by their physical placement in memory (like an array) but by logical links that are stored as part of the data in structure itself.
- The link is in the form of a pointer to another structure this can be represented as:

```
struct node{
        int item;
        struct node*  next;
};
```

Here the item could be of any complex data type like int, float etc.



- **Self refrential structures:** Structures which contain a member field that points to the same structure type are called self refrential structures.
- The structure may contain more than one item with different data types. However, one of the item must be pointer of the type tag name.

Structtagname{

typemember1;

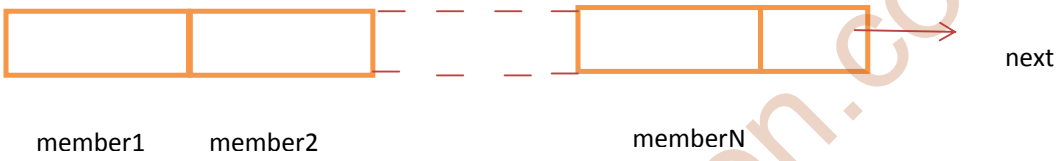
typemember2;

……

typememberN;

structtagname* next;

};



Concept of Linking:

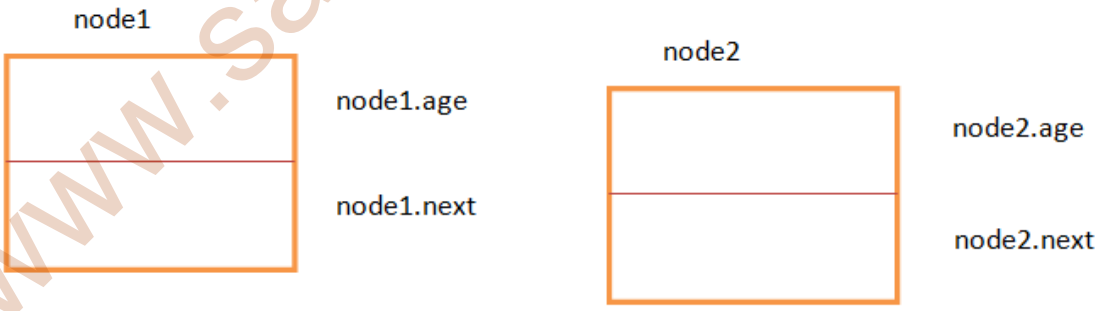
Let us consider an example to illustrate concept of linking. Define a structure as follows.

```
struct node{
        float age ;
        struct node*  next;
};
```

Let us assume list contains two nodes node1and node2. They are of type struct**linked_list** and are defined as follows**:**

structlinked_list node1,node2;

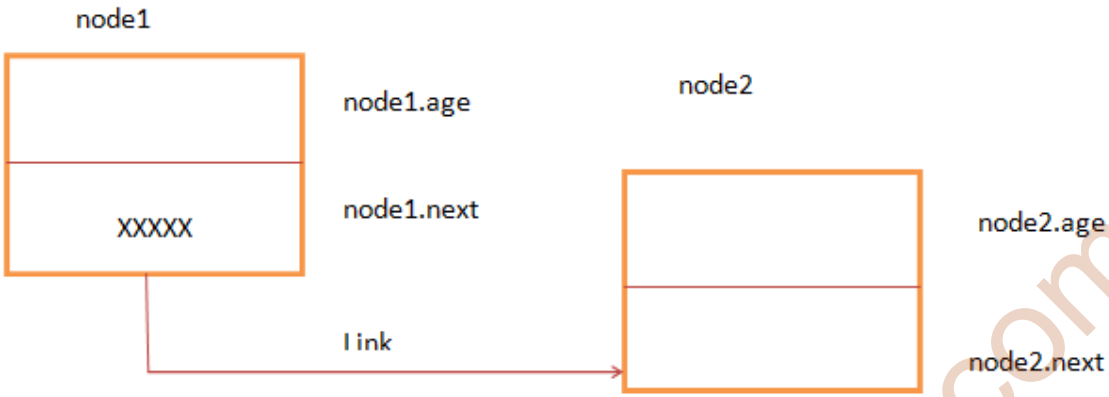
This statement creates space for two nodes each containing two empty fields as shown below.



The next pointer of node1 can be made point to node2 by statement

node1.next=&node2;

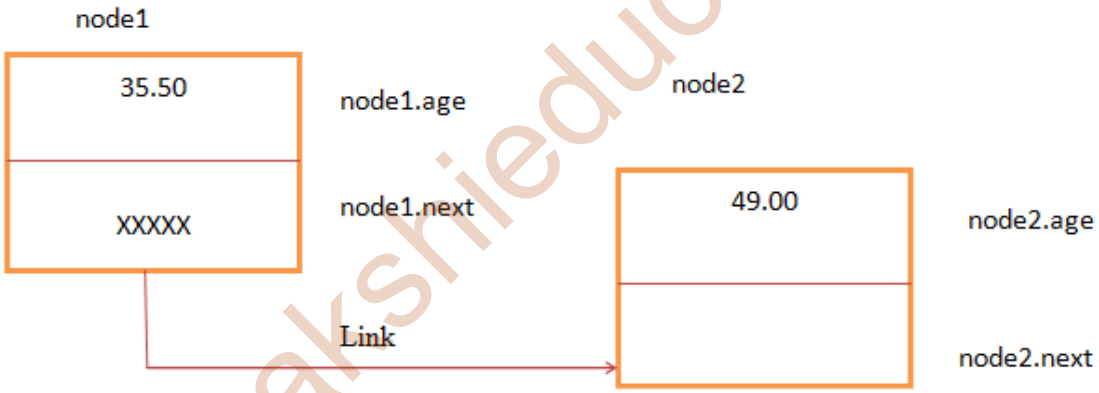
www.sakshieducation.com



"XXXX" is the address of node2 where the value of the variable node2.age will be stored. Let us assign values to the field age.

node1.age=35.50;

node2.age=49.00;



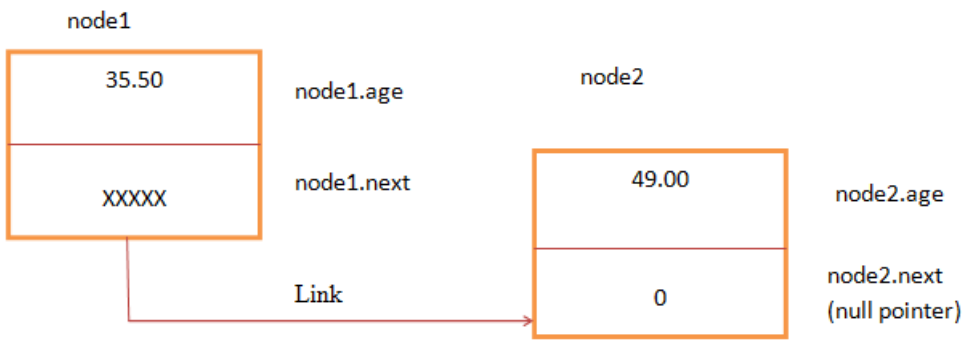
We may continue this process to create a linked list of any number of values. For example

node2.next=&node3

In our two-node list, the end of the list is marked as follows.**node2.next=0;**

The final linked list containing two nodes is shown below.

www.sakshieducation.com



Linked List Operations:

Main operations:

- Insert: inserts an element into the list.
- Delete: Removes and returns the specified position element from the list.

Auxiliary operations:

Delete list: removes all elements from the list.

Count: Returns number of elements in the list.

Find nth from the end of the list etc...

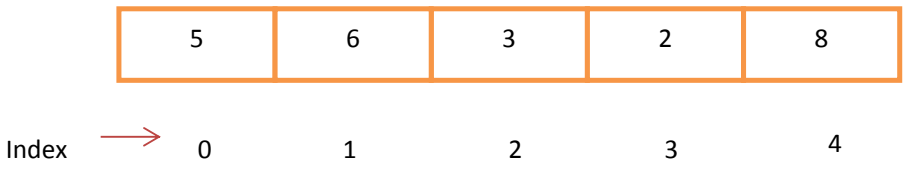
Why Linked Lists?

There are many data structures to do same thing as linked list. Before discussing this we should know difference between arrays and linked lists. Both are used to store collections of data. But, usage is different. That means in which cases arrays are suitable and in which cases linked lists are suitable.

Arrays Overview: We have studied earlier; an array is used to store data in consecutive memory locations. Arrays are considered as a good example for simple data structure.

- Arrays can be used for random access of fixed amount of data.
- We can also use arrays to create linear data structures, such as stack and queue.

The array elements can be accessed in a constant time by using the index of the particular element as the subscript.



www.sakshieducation.com

Why Constant Time is taken for Accessing Array Elements?

To access an array element, address of an element is computed as an offset from the base address of the array. To get the address of the next element one multiplication is needed to compute what is supposed to add to the base address. So, first the size of an element of that data type is calculate and then it is multiplied with the index of the element to get the value to be added to the base address.

This process takes one addition and one multiplication. Since, these two operations take constant time; we can say that array elements can be accessed in constant time.

Advantages & Disadvantages of Arrays:

Advantages:

1. Simple and easy to use.
2. We can access an array element faster (constant access).

Disadvantages:

1. The array size is fixed. Before we are going to use, we have to specify the size of an array.
2. One block allocation: The memory allocation for an array is done at beginning itself. Sometimes we may not get memory for complete array if array size is big.
3. Complex position-based insertion: To insert an element at given position, we may need to perform shift operation. To insert an element at beginning of an array expensive (to perform shift operation).

Linked Lists:

Linked list is a dynamic data structure and it has both advantages and disadvantages.

Advantages:

1. The primary advantage is the list can grow or shrink during execution of program at constant time.
2. It doesn't waste memory space. It just uses the memory for to specify the number nodes to be used. Because, it is necessary to define.
3. The important advantage of list is it provides flexibility to rearrange items efficiently

Disadvantages:

1. The major limitation is accessing of list items is little time consuming. Because in array we can access element in constant time i.e. O(1). In linked list takes O(n)to access an element in the worst case.
   Another reason to advantage of array access time is special locality in memory. Arrays are defined in consecutive memory blocks and so any array element will be physically near its neighbours.

www.sakshieducation.com

2. Linked list will use more storage than an array with the same number of items. This is because each item has an additional link field.
- If we deal with fixed length list, it would be better to use an array rather than a linked list.

Comparison of Linked Lists with Arrays:

Parameter	Linked List	Array
Indexing	O(n)	O(1)
Insertion/deletion at beginning	O(1)	-
Insertion/deletion at ending	O(n)	-
Insertion/deletion at middle	O(n)	-
Wasted space	O(n)	0

Types of Linked Lists: There are different types of lined lists.

1. Singly Linked Lists
2. Two-way or Doubly Linked Lists
3. Circular Linked Lists
4. Circular Doubly Linked Lists.

www.sakshieducation.com