6.9 DOUBLY LINKED LIST

So far you have studied singly linked lists and their variation circular linked lists. One of the most striking disadvantages of these lists is that the inability to traverse the list in the backward direction. In most of the real world applications it is necessary to traverse the list both in forward direction and backward direction. The most appropriate data structure for such an application is a **doubly linked list.**

A doubly linked list is one in which all nodes are linked together by multiple number of links which help in accessing both the successor node and predecessor node from the given node position. It provides bi-directional traversing.

Each node in a doubly linked list has two link fields. These are used to point to the successor node and predecessor nodes. Figure 6.18 shows the structure of a node in the doubly linked list.

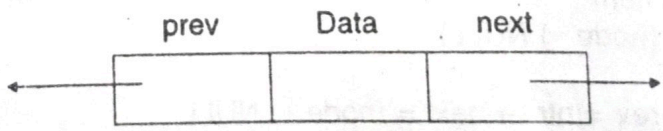

Fig. 6.18. A node in doubly linked list.

The LEFT link points to the predecessor node and the RIGHT link points to the successor node. The structure definition for the above node on C is as follows.

```
struct node
{     int num ;
      struct node *prev ;
      struct node *next ;
} ;
typedef struct node NODE ;
```