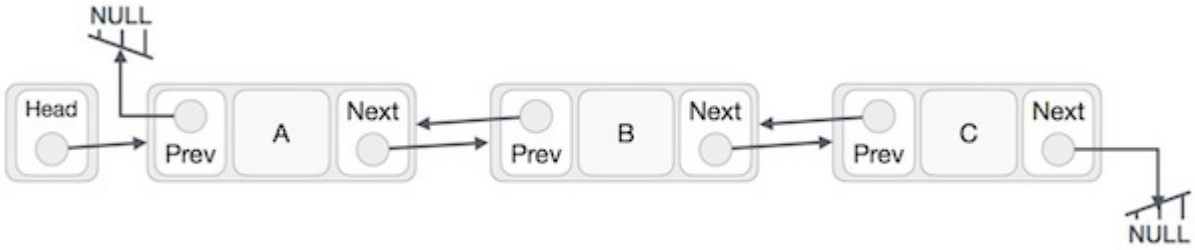
DATA STRUCTURE - DOUBLY LINKED LIST

Doubly Linked List is a variation of Linked list in which navigation is possible in both ways either forward and backward easily as compared to Single Linked List. Following are important terms to understand the concepts of doubly Linked List

- **Link** − Each Link of a linked list can store a data called an element.
- **Next** − Each Link of a linked list contain a link to next link called Next.
- **Prev** − Each Link of a linked list contain a link to previous link called Prev.
- **LinkedList** − A LinkedList contains the connection link to the first Link called First and to the last link called Last.

Doubly Linked List Representation



As per above shown illustration, following are the important points to be considered.

- Doubly LinkedList contains an link element called first and last.
- Each Link carries a data field*s* and a Link Field called next.
- Each Link is linked with its next link using its next link.
- Each Link is linked with its previous link using its prev link.
- Last Link carries a Link as null to mark the end of the list.

Basic Operations

Following are the basic operations supported by an list.

- **Insertion** − add an element at the beginning of the list.
- **Deletion** − delete an element at the beginning of the list.
- **Insert Last** − add an element in the end of the list.
- **Delete Last** − delete an element from the end of the list.
- **Insert After** − add an element after an item of the list.
- **Delete** − delete an element from the list using key.
- **Display forward** − displaying complete list in forward manner.
- **Display backward** − displaying complete list in backward manner.

Insertion Operation

Following code demonstrate insertion operation at beginning in a doubly linked list.

```
//insert link at the first location
void insertFirst(int key, int data) {
```

```
   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct node));
   link->key = key;
   link->data = data;

   if(isEmpty()) {
      //make it the last link
      last = link;
   }else {
      //update first prev link
      head->prev = link;
   }

   //point it to old first link
   link->next = head;

   //point first to new first link
   head = link;
}
```

Deletion Operation

Following code demonstrate deletion operation at beginning in a doubly linked list.

```
//delete first item
struct node* deleteFirst() {

   //save reference to first link
   struct node *tempLink = head;

   //if only one link
   if(head->next == NULL) {
      last = NULL;
   }else {
      head->next->prev = NULL;
   }

   head = head->next;

   //return the deleted link
   return tempLink;
}
```

Insertion at End Operation

Following code demonstrate insertion operation at last position in a doubly linked list.

```
//insert link at the last location
void insertLast(int key, int data) {

   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct node));
   link->key = key;
   link->data = data;

   if(isEmpty()) {
      //make it the last link
      last = link;
   }else {
      //make link a new last link
      last->next = link;
      //mark old last node as prev of new link
      link->prev = last;
   }

   //point last to new last node
   last = link;
```

```
}
```

To see the implementation in C programming language, please click here.