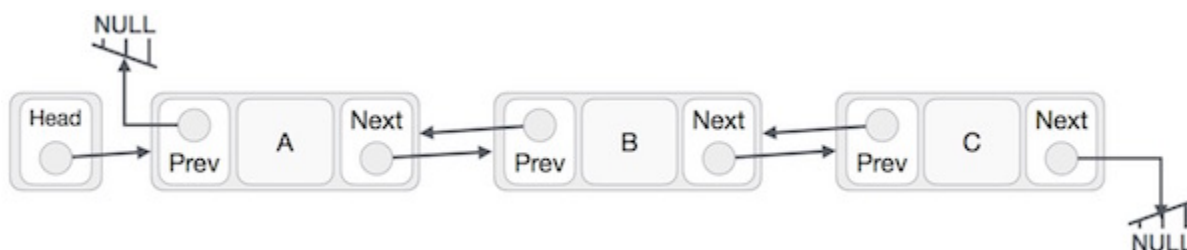


Data Structure - Doubly Linked List

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 the important terms to understand the concept 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 contains a link to the next link called Next.
- **Prev** – Each link of a linked list contains a link to the previous link called Prev.
- **LinkedList** – A Linked List contains the connection link to the first link called First and to the last link called Last.

Doubly Linked List Representation



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

- Doubly Linked List contains a link element called first and last.
- Each link carries a data field(s) and two link fields called next and prev.
- Each link is linked with its next link using its next link.
- Each link is linked with its previous link using its previous link.
- The last link carries a link as null to mark the end of the list.

Basic Operations

Following are the basic operations supported by a list.

- **Insertion** – Adds an element at the beginning of the list.
- **Deletion** – Deletes an element at the beginning of the list.
- **Insert Last** – Adds an element at the end of the list.

- **Delete Last** – Deletes an element from the end of the list.
- **Insert After** – Adds an element after an item of the list.
- **Delete** – Deletes an element from the list using the key.
- **Display forward** – Displays the complete list in a forward manner.
- **Display backward** – Displays the complete list in a backward manner.

Insertion Operation

Following code demonstrates the insertion operation at the beginning of a doubly linked list.

Example

```
//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 demonstrates the deletion operation at the beginning of a doubly linked list.

Example

```
//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 the End of an Operation

Following code demonstrates the insertion operation at the last position of a doubly linked list.

Example

```
//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](#) .