**LAB # 07**

**Doubly Linked List implementation**

**Object**

Implementing doubly linked list and associated operations.

**Theory**

* A doubly linked list/2-way LL is often more convenient.
* Nodes store:
  + element
  + link to the previous node
  + link to the next node
* Special trailer and header nodes

|  |  |
| --- | --- |
| **Public Class Name: DNode** |  |
| **Member Variables** | **Responsibilities** |
| Protected <Object> element | public Object getElement()  Returns element from the node |
| Private DNode nextlink | public DNode getNext()  Returns link of next node in the LL |
| Private DNode previouslink | public setElement(Object newElem)  Set element part of the node to new element |
| **Constructors** | public setNext(DNode newNextlink)  Set nextlink part of node to newnextlink |
| public DNode() | public DNode getPrevious()  Returns link of previous node in the LL |
|  | public setPrevious(DNode newpreviouslink)  Set previouslink part of node to newpreviouslink |

|  |  |
| --- | --- |
| **Public Class Name: DoublyList** |  |
| **Member Variables** |  |
| Protected Node STARTNODE , Protected Node ENDNODE  STARTNODE points to first node in SLL. ENDNODE points to last node. Empty SLL has STARTNODE =ENDNODE= NULL | Public insertBefore(Object item, Node locptr)  Insert an item in Doubly Linked List before node referred by locp |
| Protected N  N is the no of elements in the SLL. N is 0 if SLL is empty | Public <Object> delete(Node locptr)  Delete node from the Doubly LL and returns the element from deleted node |
| **Constructors** | Public <Node> search(item)  Searches item in the Doubly LL and returns link of the node if searh successful |
| DoublyList()  Construct to create empty LL where STARTNODE and ENDNODE=NULL | Public int size()  Returns no of elements in SLL |
| **Responsibilities** | Public <boolean> isEmpty()  Returns TRUE if SLL is empty else FALSE |
| Public insertAfter(Object item, Node locptr)  Insert an item in Doubly Linked List after node referred by locptr |  |

**Algorithm to insert node at particular location**:

insertAfter(locptr,item):

Create a new node new

new.setElement(item)

if STARTNODE:=NULL then {insert new node in empty DLL}

STARTNODE:=new, ENDNODE:=new

else if locptr=NULL then {insert new node at the beginning of DLL}

new.setNextLink(STARTNODE)

STARTNODE.setPrevLink(new)

STARTNODE:=new

Else

new.setPrevLink(locptr) {link new to its predecessor}

new.setNextLink(locptr.getNextLink()) {link new to its successor}

locptr.setNextLink(new) {link locptr to its new successor}

if locptr:=ENDNODE then {if new node becomes the end node of DLL}

ENDNODE:=new

else

(new.getNextLink()).setPrevLink(new) {link locptr’s old successor to new}

N=N+1

return v {the position for the element e}

**Algorithm to delete node:**

delete(locptr, item):

Precond: DLL is not empty

item = locptr.element {a temporary variable to hold the return value}

If locptr=STARTNODE then {delete at beginning of DLL}

STARTNODE:=locptr.getNextLink()

Else if STARTNODE=ENDNODE=locptr then

STARTNODE:=NULL, ENDNODE:=NULL {delete last node}

Else

(locptr.getPrevLink()).setNextLink(locptr.getNextLink()) {linking out p}

(locptr.getNextLink()).setPrevLink(locptr.getPrev())

if locptr=ENDNODE then {delete node at end of DLL}

ENDNODE:=locptr.getPrevlink()

locptr.setPrevLink(null) {invalidating the position p}

locptr.setNextLink(null)

N:=N-1

return item

**Algorithm to search element:**

1. Set PTR := START, LOC:= NULL

2. Repeat Steps 3-4 while PTR ≠ NULL

3. if ptr.getElement() = ITEM then

4. Set Loc:=PTR and return

Else

5. Set PTR:= ptr.getNextLink()

[end of if]

[end of loop]

7. Return LOC

**Lab Task**

Write a program which includes these 3 classes

1. Create a class DNode with all the methods defined above.
2. Create a class DoublyLL to implement all the given operations.
3. Write a Demo class to create
   * Unsorted Linked List and
   * Sorted linked list