Doubly Linked Lists

Lecture 19
Section 11.3

Robb T. Koether

Hampden-Sydney College

Mon, Mar 2, 2009
Outline

1. Doubly Linked Lists
2. Doubly Linked List Nodes
3. Inserting and Deleting
4. Assignment
Definition (Doubly Linked List)

A **doubly linked list** is a linked list in which each node has two links: a forward link to its successor node and a backward link to its predecessor node.
### Doubly LinkedListNode Data Members

- **T data** - The value stored in the node.
- **DoublyLinkedListNode* prev** - A pointer to the previous node.
- **DoublyLinkedListNode* next** - A pointer to the next node.

A doubly linked list must use doubly linked list nodes.
## Doubly Linked List Data Members

<table>
<thead>
<tr>
<th>Data Member Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>int mSize</strong></td>
<td>Number of elements in the list.</td>
</tr>
<tr>
<td><em><em>DoublyLinkedListNode</em> head</em>*</td>
<td>Pointer to the first node.</td>
</tr>
<tr>
<td><em><em>DoublyLinkedListNode</em> tail</em>*</td>
<td>Pointer to the last node.</td>
</tr>
</tbody>
</table>
Chasing Pointers

- We can move both forwards and backwards in the list.
- When chasing pointers, it is not necessary to keep a pointer to the previous node.
Validity Requirements

- \( mSize \geq 0 \).
- If \( mSize = 0 \), then \( \text{head} == \text{NULL} \) and \( \text{tail} == \text{NULL} \).
- If \( mSize > 0 \), then \( \text{head} != \text{NULL} \) and \( \text{tail} != \text{NULL} \).
- If \( mSize = 1 \), then \( \text{head} == \text{tail} \).
- If \( mSize \geq 1 \), then
  - The next pointer in node \( mSize - 1 \) is NULL.
  - The prev pointer in node 0 is NULL.
Validity Requirements

- If \( mSize \geq 2 \), then
- For every \( i \) from 1 to \( mSize - 2 \), the next pointer in node \( i \) is not NULL.
- For every \( i \) from 1 to \( mSize - 1 \), the prev pointer in node \( i \) is not NULL.
- The prev pointer of node 1 equals head.
- The next pointer of node \( mSize - 2 \) equals tail.
Validity Requirements

- If \( m\text{Size} \geq 3 \), then

- For every \( i \) from 1 to \( m\text{Size} - 2 \), the next pointer of node \( i - 1 \) equals the prev pointer of node \( i + 1 \).
Inserting and Deleting in a Doubly Linked List

- Apply the 12-step method to
  - Insert a node into a doubly linked list.
  - Delete a node from a doubly linked list.
The **DoublyLinkedList** Class

- Download `doublylinkedlistnode.h`
- Download `doublylinkedlist.h`
- Download and run `ListTest.cpp`
Assignment

Homework

- Read Section 11.3, pages 602 - 604.