Linked Lists
Lecture 16
Section 6.3

Robb T. Koether
Hampden-Sydney College
Mon, Feb 23, 2009
Outline

1. The LinkedList Class
2. The LinkedListNode Class
3. Random Access vs. Sequential Access
4. Assignment
The name of the linked list class is `LinkedList`.

In a linked list, each node is allocated dynamically when the element is added to the list.

A linked list always uses exactly the amount of memory it needs.

A linked list is more efficient than an array list in some ways and less efficient in others.
**LinkedList Data Members**

- **int mSize** - The number of elements in the list.
- **LinkedListNode* head** - A pointer to the first node (which contains the first element).
We create a separate class, `LinkedListNode`, as a data type that stores a single element of a LinkedList object.

Each `LinkedListNode` contains a pointer that links it to the next node.
**LinkedListNode Data Members**

- **T data** - The list element $a_i$.
- **LinkedListNode* next** - A pointer to the node containing list element $a_{i+1}$ or NULL if $a_i$ is the last element.
**LinkedListNode Public Member Functions**

- `T value() const;`
  Returns a copy of the data member of the node.

- `T& value();`
  Returns a reference to the data member of the node.

- `LinkedListNode* nextNode();`
  Returns a copy of the pointer to the next node.
LinkedListNode Member Functions

LinkedListNode Private Member Functions

- LinkedListNode(const T& value = T());
  Constructs a linked list node containing the specified value.

- It can be invoked only by friend classes.
- The only friend class (for now) will be the LinkedList class.
The **LinkedListNode** Class

**Download** linkedlistnode.h.
Validity Requirements of a Linked List

- mSize \( \geq 0 \).
- \textbf{If } mSize = 0, \textbf{then} head == NULL.
- \textbf{If } mSize > 0, \textbf{then} head != NULL.
- \textbf{For every } i \textbf{ from } 0 \textbf{ to } mSize - 2, \textbf{in node } i, next != NULL.
- \textbf{In node } mSize - 1, next == NULL.
Random Access vs. Sequential Access

- A linked list allows sequential access, but not random access.
- To find the list element in position $n$, we must begin at the head of the list and move sequentially through positions $1, 2, \ldots, n - 1$ before reaching position $n$.
- An array allows random, or direct, access.
Random Access vs. Sequential Access

- Sequential access is much slower than random access if the list elements are accessed randomly.
- However, sequential access may be faster than random access if the list elements are accessed sequentially.
The Technique of Chasing Pointers

Chasing Pointers

```cpp
LinkedListNode* node = head;
for (int i = 0; i < pos; i++)
    node = node->next;
```

- Technique of chasing pointers to locate position `pos`.
Random Access vs. Sequential Access

```java
for (int i = 0; i < list.size(); i++)
    list[i] = 0;
```

How would the performance of the above `for` loop differ between the `ArrayList` and `LinkedList` implementations?
The LinkedList Class

- Download `linkedlist.h`.
- Download and run `ListTest.cpp`. 
Assignment

Homework

- Read Section 6.4, pages 287 - 294.