Implementing Linked Lists

Lecture 17
Section 6.5

Robb T. Koether

Hampden-Sydney College

Wed, Feb 25, 2009
Outline

1. The Method
   - Test Preconditions
   - Create New Nodes
   - Locate Insertion Point
   - Draw “Before” Picture
   - Draw “After” Picture
   - Modify Pointers
   - Arrange Statements in Order
   - Consolidate the Cases
   - Combine Cases
   - Distinguish the Cases
   - Delete Old Nodes
   - Miscellaneous

2. Assignment
The method outlined here offers a reliable strategy for modifying a linked list.

As we go through the method, we will apply it to the problem of inserting a new element into a linked list.

**The insert() Prototype**

```cpp
void insert(int pos, const T& value);
```
The Method of Modifying a Linked List

Step 1

(1) Test any necessary pre-conditions.

Example (\texttt{insert()})

\begin{verbatim}
assert(pos >= 0 && pos <= size);
\end{verbatim}
The Method of Modifying a Linked List

Step 2

(2) Create any additional nodes and pointers that are needed for the task.

Example \( \text{insert()} \)

\[
\text{ListNode<T>* new_node} = \text{new ListNode<T>(value)};
\]
Step 3

(3) Use pointers to locate the position(s) in the list where the change will take place.

Example (insert())

```c++
LinkedListNode<T>* curr_node = head;
LinkedListNode<T>* prev_node = NULL;
for (int i = 0; i < pos; i++)
{
    prev_node = curr_node;
    curr_node = curr_node->next;
}
```
Now divide the task of modifying the list into distinct cases.

Begin with the most general case.

Work down to the least general case.

1. Insert into the middle of a non-empty list.
2. Insert at the head of a non-empty list.
3. Insert at the tail of a non-empty list.
4. Insert into an empty list.
Step 4

(4) Draw a picture of the structure before the modifications take place.

Example (insert())
Step 4

(4) Show any newly created nodes.

Example (**insert()**)
Step 4

(4) Label each relevant pointer.

Example (\texttt{insert()})

```
prev_node->next
prev_node
curr_node
new_node
new_node
prev_node->next
curr_node
```
The Method

Step 5

(5) Draw a picture of the structure after the modifications take place.

Example (insert())

```
prev_node->next
prev_node
curr_node
new_node
```
Step 5

(5) Draw a picture of the structure after the modifications take place.

Example (insert())

```
prev_node->next
prev_node
curr_node
new_node
new_node
prev_node->next
curr_node
```

```
The Method

Step 6

(6) For the pointers which were modified, write the assignment statements that will modify them.

Example (insert())

```
prev_node->next = new_node;
new_node->next = curr_node;
```
Step 7

(7) Arrange the assignment statements in the correct order.

Example (insert())

// Case 1
prev_node->next = new_node;
new_node->next = curr_node;

// Case 2
head = new_node;
new_node->next = curr_node;

// Case 3
prev_node->next = new_node;
new_node->next = NULL;

// Case 4
head = new_node;
new_node->next = NULL;
Step 8

(8) Consolidate the cases. Determine what code is common to all cases. Write the common code either before or after dividing into cases, as appropriate.

Example (\texttt{insert()})

Replace

\begin{verbatim}
new_node->next = NULL;
\end{verbatim}

with

\begin{verbatim}
new_node->next = curr_node;
\end{verbatim}

Then the line is common to all four cases.
The Method

Step 9

(9) Combine cases that use the same code into a single case.

Example (`insert()`)

- Cases 1 and 3 are identical.
- Cases 2 and 4 are identical.
The Method

**Step 10**

(10) Distinguish the cases. Find conditions that are unique to each case. Write the if statements and the code to handle the separate cases.

**Example (insert())**

- Based on the values of the pointers,
  - In cases 1 and 3, `prev_node != NULL`.
  - In cases 2 and 4, `prev_node == NULL`.

- Based on the values of indices,
  - In cases 1 and 3, `pos > 0`.
  - In cases 2 and 4, `pos == 0`. 
The Method

Example (insert())

```c
if (prev_node == NULL)
    head = new_node;
else
    prev_node->next = new_node;
```
Step 11

(11) Delete any old nodes.

Example

In this example, there are no nodes to be deleted.
Step 12

(12) Write any other statements necessary to complete the task.

Example `insert()`

`mSize++;`
Example (insert())

```cpp
template <class T>
void LinkedList<T>::insert(int pos, const T& value) {
    // Test validity of parameters
    assert(pos >= 0 && pos <= size);
    // Create a new node
    LinkedListNode<T>* new_node = new LinkedListNode<T>(value);
    // Locate insertion point
    LinkedListNode<T>* curr_node = head;
    LinkedListNode<T>* prev_node = NULL;
    for (int i = 0; i < pos; i++) {
        prev_node = curr_node;
        curr_node = curr_node->next;
    }
    // Modify pointers to insert new node
    new_node->next = curr_node;
    if (prev_node == NULL)
        head = new_node;
    else
        prev_node->next = new_node;
    // Update the size
    mSize++;
    return;
}
```
The **LinkedList Class**

- **Download** `linkedlistnode.h`
- **Download** `linkedlist.h`
- **Download and run** `ListTest.cpp`
Homework

- Read Section 6.5, pages 295 - 300.