Implementing Linked Lists

Lecture 17
Sections 17.1 - 17.3

Robb T. Koether

Hampden-Sydney College

Mon, Feb 27, 2017

1. Modifying a Linked List
   - Test Preconditions
   - Create New Nodes
   - Locate “Action” 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

Outline

Modifying a Linked List

The `insert()` Prototype

```
void insert(int pos, const T& value);
```

- 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.

Outline

The Method of Modifying a Linked List

Step 1

```
assert(pos >= 0 && pos <= m_size);
```

Test any necessary pre-conditions.

Outline

The Method of Modifying a Linked List

Step 2

```
LinkedListNode<T>* new_node
    = new LinkedListNode<T>(value);
```

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

Outline

The Method

Step 3

```
LinkedListNode<T>* succ = head;
LinkedListNode<T>* pred = NULL;
for (int i = 0; i < pos; i++)
{
    pred = succ;
    succ = succ->m_next;
}
```

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

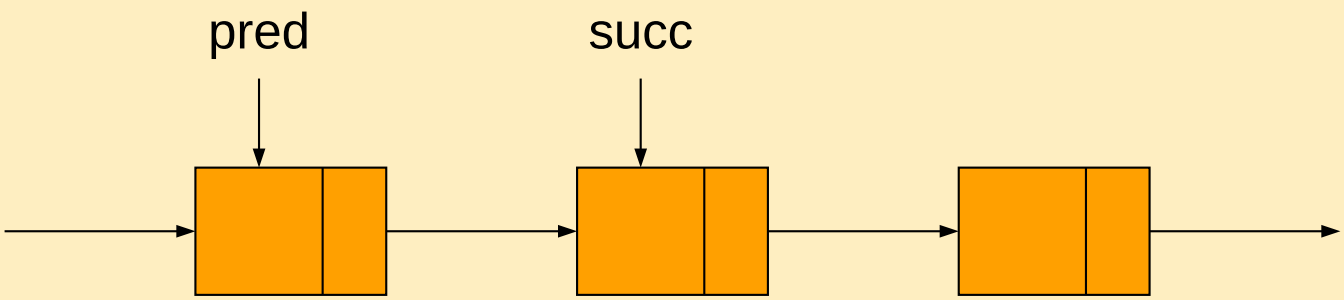
The Method

- 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.

Outline

The Method

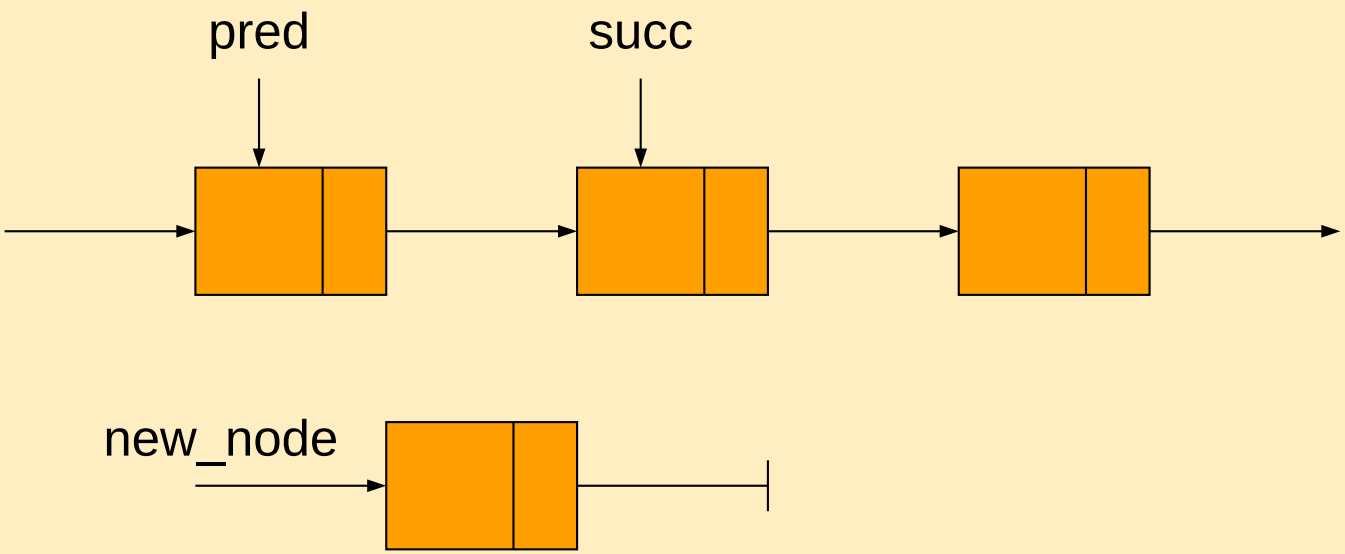
Step 4



Draw a picture of the structure before the modifications take place.

The Method

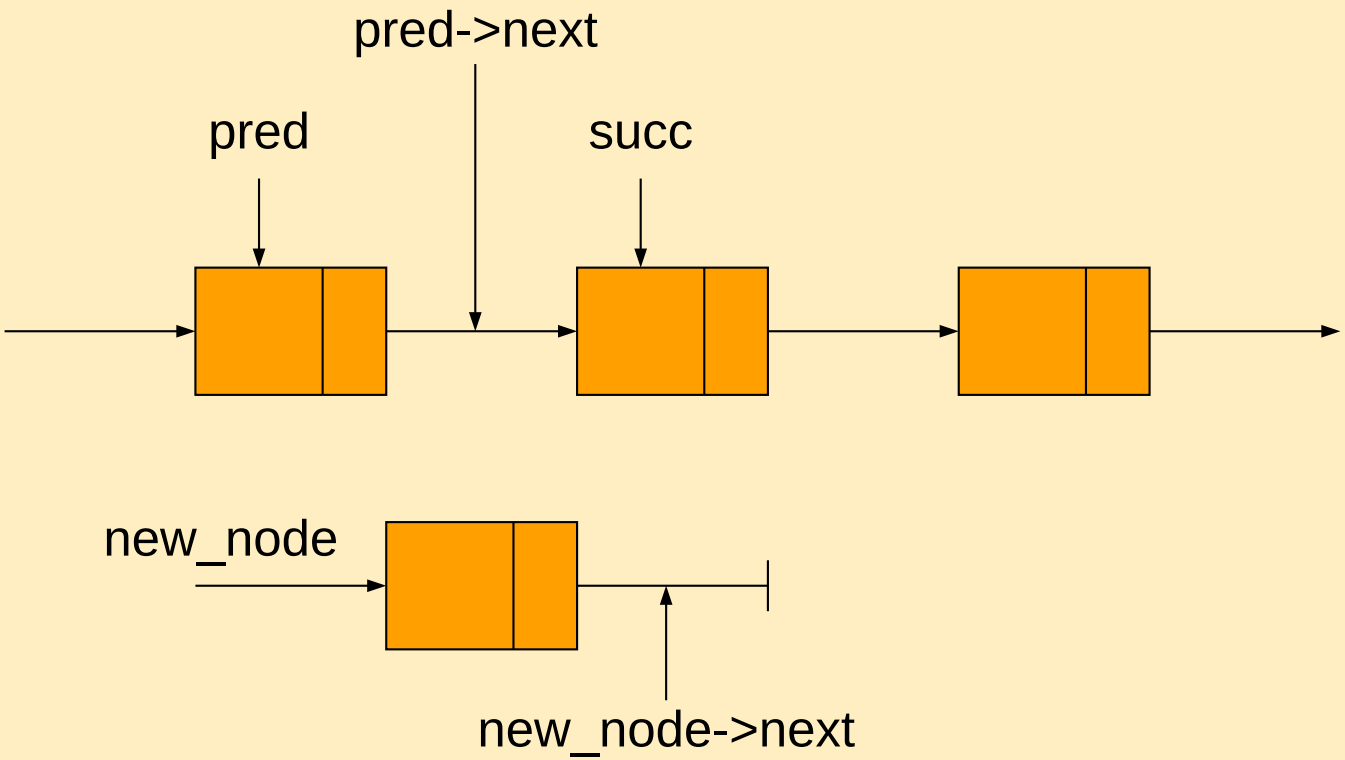
Step 4



Show any newly created nodes.

The Method

Step 4

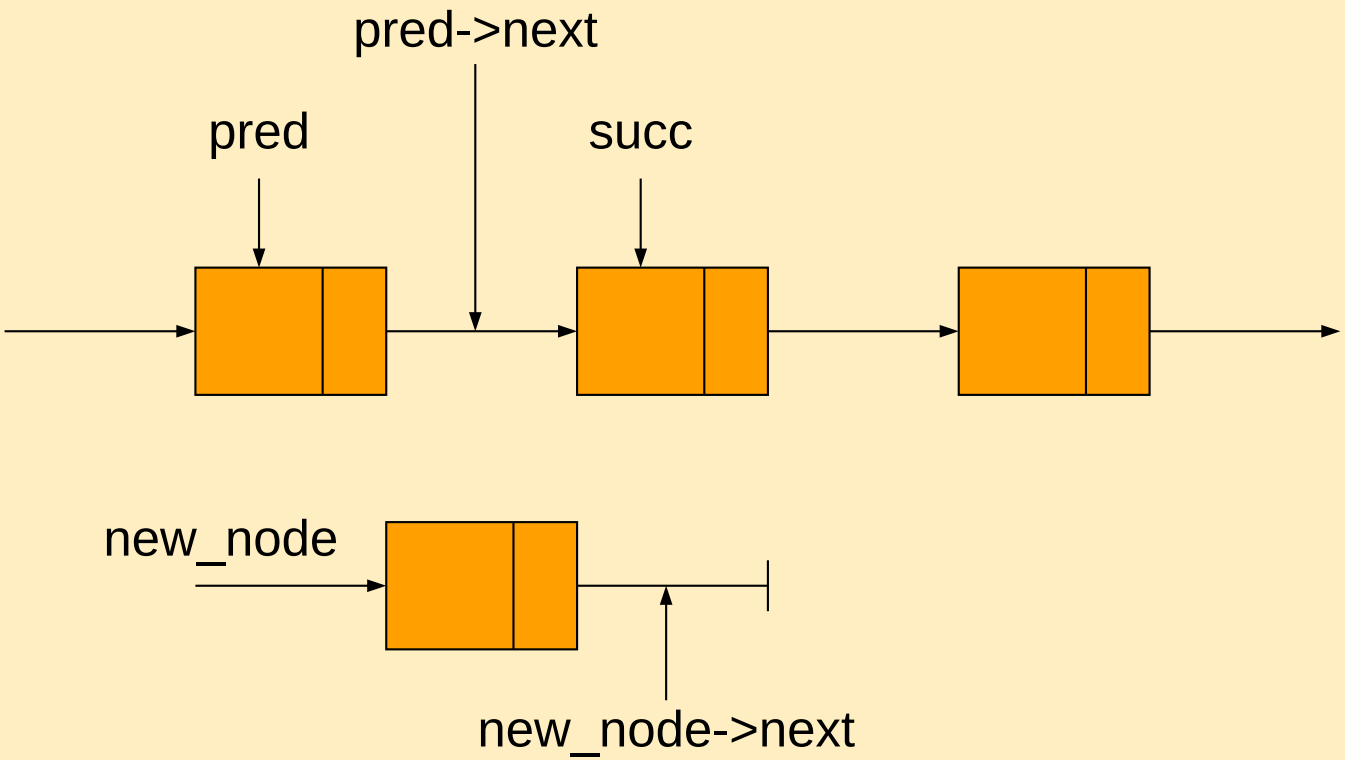


Label each relevant pointer.

Outline

The Method

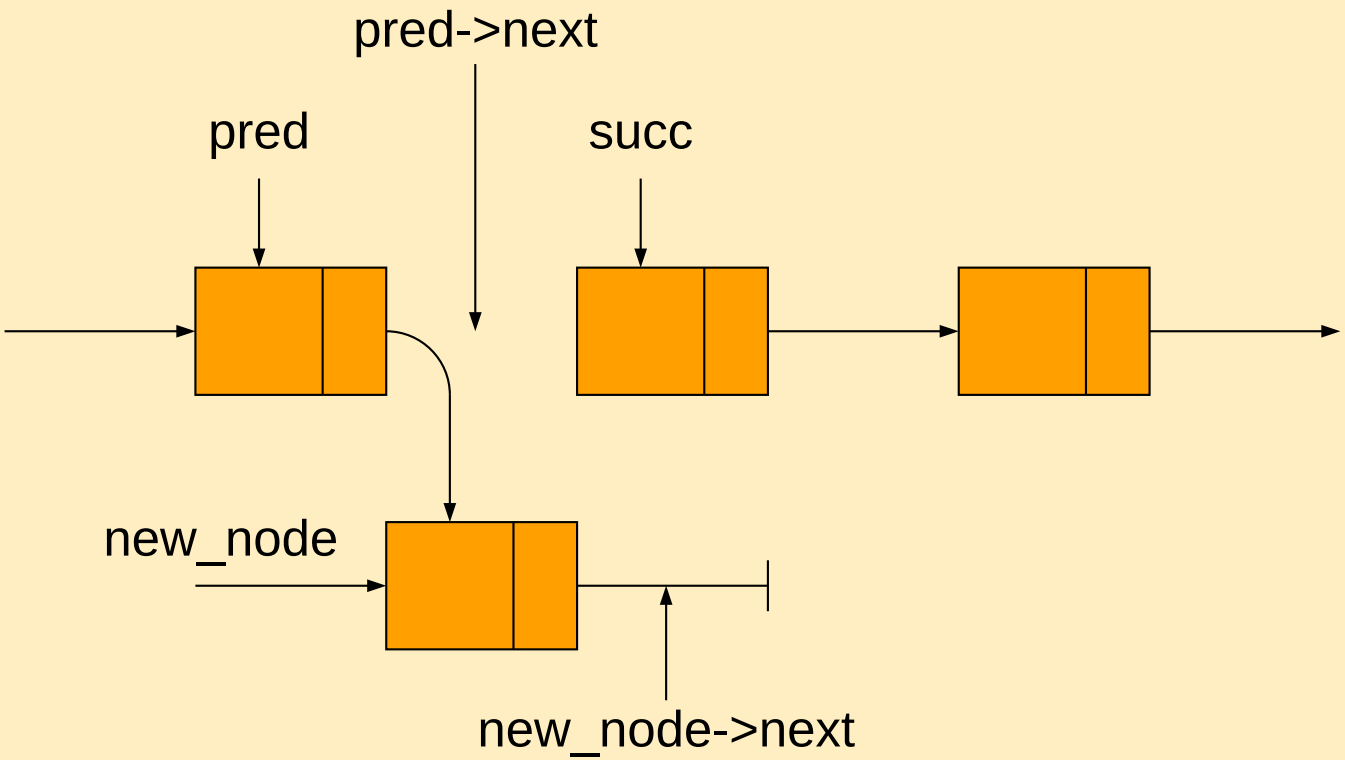
Step 5



Draw a picture of the structure after the modifications take place.

The Method

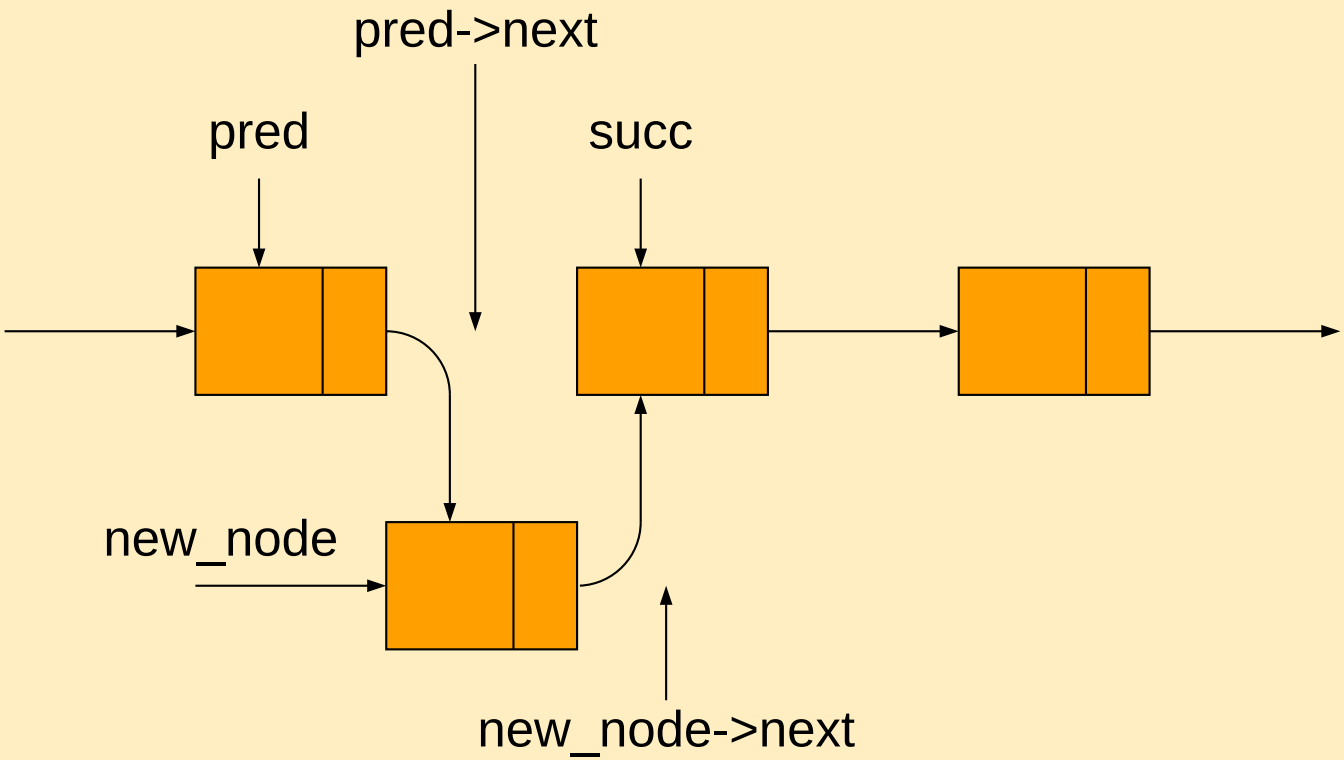
Step 5



Draw a picture of the structure after the modifications take place.

The Method

Step 5



Draw a picture of the structure after the modifications take place.

Outline

The Method

Step 6

```
pred->m_next = new_node;
new_node->m_next = succ;
```

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

Outline

Arrange the Statements in Order

Step 7

```
pred->m_next = new_node;
new_node->m_next = succ;
```

Arrange the statements it correct order.

The Other Cases

- Now apply Steps 4 - 7 to the other three cases:
  - Insertion at the head.
  - Insertion at the tail.
  - Insertion into an empty list (head and tail).

The Method

Step 7

```
//  Case 1
    pred->m_next = new_node;
    new_node->m_next = succ;
//  Case 2
    head = new_node;
    new_node->m_next = succ;
//  Case 3
    pred->m_next = new_node;
    new_node->m_next = NULL;
//  Case 4
    head = new_node;
    new_node->m_next = NULL;
```

Arrange the assignment statements in the correct order.

Outline

The Method

Step 8

Replace

```
new_node->m_next = NULL;
```

with

```
new_node->m_next = succ;
```

Then the line is common to all four cases.

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

Outline

The Method

Step 9

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

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

Outline

The Method

Step 10

- Based on the values of the pointers,
  - In cases 1 and 3, `pred != NULL`.
  - In cases 2 and 4, `pred == NULL`.
- Based on the values of indices,
  - In cases 1 and 3, `pos > 0`.
  - In cases 2 and 4, `pos == 0`.

- Distinguish the cases.
- Find conditions that are unique to each case.
- Write the **`if`** statements and the code to handle the separate cases.

The Method

Step 10

```
if (pred == NULL)
    head = new_node;
else
    pred->m_next = new_node;
```

Outline

The Method

Step 11

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

Delete any old nodes.

Outline

The Method

Step 12

```
m_size++;
```

Write any other statements necessary to complete the task.

The Method

The `insert()` Function

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

The `LinkedList` Class

The `LinkedList` Class

- `linkedlistnode.h`.
- `linkedlist.h`.
- `List Test.cpp`.

Outline

Assignment

Assignment

- Read Sections 17.1 - 17.3.