

Linked Lists: Adding to Tail

```
class List; // forward declaration
class ListNode {
    friend class List; // List to access private data
private:
    string sName;
    ListNode* pInNext;
public:
    ListNode() : sName(""), pInNext(NULL) {}
    ListNode(string& rsData) : sName(rsData), pInNext(NULL) {}
};
```

```
class List {
private:
    int nNumItems;
    ListNode* pInHead;
public:
    List() : nNumItems(0), pInHead(NULL) {}
    void AddAtTail(string& rsNewData);
    void OutputAllRecords(ostream& ros);
};

void List::AddToTail(string& rsNewData)
{
    ListNode* pInTmp = new ListNode(rsNewData);
    nNumItems++;
    if (pInHead == NULL) // empty list
        pInHead = pInTmp;
    else { // find the last node in list
        ListNode* pInPrevious = NULL;
        ListNode* pInCurrent = pInHead;
        while (pInCurrent != NULL) {
            pInPrevious = pInCurrent;
            pInCurrent = pInCurrent->pInNext;
        }
        pInPrevious->pInNext = pInTmp;
    }
}
```

```
void List::OutputAllRecords(ostream& ros)
{
    ListNode* pInCurrent = pInHead;
    while (pInCurrent != NULL) {
        ros << pInCurrent->sName << ' ';
        pInCurrent = pInCurrent->pInNext;
    }
}
```

```
void main()
{
    List lPeople;
    char cMore;
    do {
        string sNewName;
        cout << "Enter a name:";
        cin >> sNewName;
        lPeople.AddAtTail(sNewName);
        cout << "More? (y/n):";
        cin >> cMore;
    } while (cMore == 'y');
    cout << "\nList of People\n\n";
    lPeople.OutputAllRecords(cout);
}
```

```
class List; // forward declaration
class ListNode {
    friend class List; // List to access private data
private:
    string sName;
    ListNode* plnNext;
public:
    ListNode() : sName(""), plnNext(NULL) {}
    ListNode(string& rsData) : sName(rsData), plnNext(NULL) {}
};
```

```
class List {
private:
    int nNumItems;
    ListNode* plnHead;
    ListNode* plnTail
public:
    List() : nNumItems(0), plnHead(NULL), plnTail(NULL) {}
    void AddAtTail(string& rsNewData);
    void OutputAllRecords(ostream& ros);
};
```

```
void List::AddToTail(string& rsNewData)
{
    ListNode* plnTmp = new ListNode(rsNewData);
    nNumItems++;
    if (plnHead == NULL) // empty list
        plnHead = plnTail = plnTmp;
    else // find the last node in list
        plnTail = plnTail->plnNext = plnTmp;
}
```