Ch. 17: Linked Lists
Part 1

CS 2308
Fall 2011

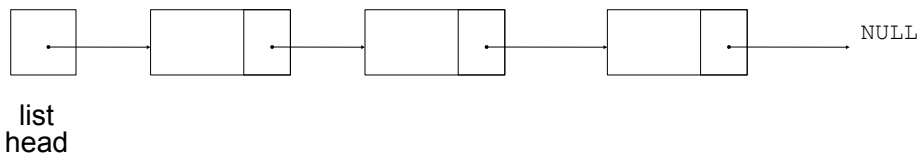
Jill Seaman

Lecture 16

Using content from textbook slides: Starting Out with C++, Gaddis, Pearson/Addison-Wesley

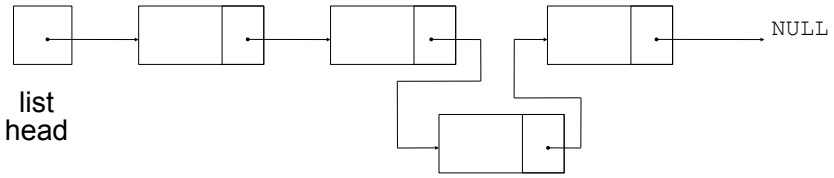
Introduction to Linked Lists

- A data structure representing a list
- A series of nodes chained together in sequence
  - Each node points to one other node.
- A separate pointer (the head) points to the first item in the list.
- The last element points to nothing (NULL)



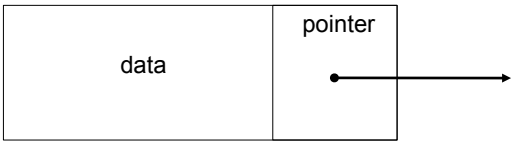
Introduction to Linked Lists

- The nodes are dynamically allocated
  - The list grows and shrinks as nodes are added/ removed.
- Linked lists can easily insert a node between other nodes
- Linked lists can easily delete a node from between other nodes



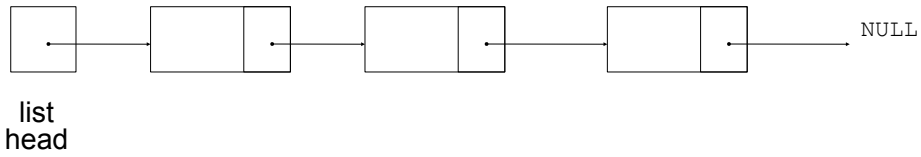
Node Organization

- Each node contains:
  - data field – may be organized as a structure, an object, etc.
  - a pointer – that can point to another node



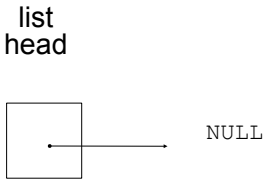
Linked List Organization

- A linked list contains 0 or more nodes
- The list head is a pointer that points to the first node.
- Each node points to the next node in the list.
- The last node points to NULL (address 0).



Empty List

- An empty list contains 0 nodes.
- The list head points to NULL (address 0)
- (There are no nodes, it's empty)



Declaring the Linked List data type

- We will be defining a class for a linked list data type that can store values of type double.
- The data type will describe the values (the lists) and operations over those values.
- In order to define the values we must:
  - define a data type for the nodes
  - define a pointer variable (head) that points to the first node in the list.

Declaring the Node data type

- Use a struct for the node type

```
struct ListNode {
   double value;
   ListNode *next;
};
```

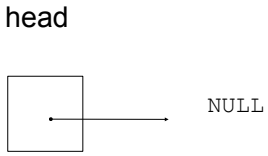
- (this is just a data type, no variables declared)
- next can hold the address of a ListNode.
  - it can also be NULL
  - “self-referential data structure”

Defining the Linked List variable

- Define a pointer for the head of the list:

```
ListNode *head = NULL;
```

- It must be initialized to NULL to signify the end of the list.
- Now we have an empty linked list:



Using NULL

- Equivalent to address 0
- Used to specify end of the list
- Use ONE of the following for NULL:

```
#include <iostream>
#include <cstddef>
```

- to test a pointer for NULL (these are equivalent):

```
while (p) ...   <==>  while (p != NULL) ...

if (!p) ...   <==>  if (p == NULL) ...
```

Linked List operations

- Basic operations:
  - create a new, empty list
  - append a node to the end of the list
  - insert a node within the list
  - delete a node
  - display the linked list
  - delete/destroy the list

Linked List class declaration

```
// file NumberList.h

#include <cstddef>   // for NULL
using namespace std;

class NumberList
{
   private:
      struct ListNode    // the node data type
      {
         double value;           // data
         struct ListNode *next;  // ptr to next node
      };
      ListNode *head;    // the list head

   public:
      NumberList();      // creates an empty list
      ~NumberList();

      void appendNode(double);
      void insertNode(double);
      void deleteNode(double);
      void displayList();
};
```

Linked List functions: constructor

- Constructor: sets up empty list

```
// file NumberList.cpp

#include "NumberList.h"


NumberList::NumberList()
{
   head = NULL;
}
```


Linked List functions: appendNode

- appendNode: adds new node to end of list
- Algorithm:

Create a new node and store the data in it
If the list has no nodes (it's empty)
  Make head point to the new node.
Else
  Find the last node in the list
  Make the last node point to the new node

Linked List functions: appendNode

- How to find the last node in the list?
- Algorithm:

  Make a pointer p point to the first element
  while (the node p points to) is not pointing to NULL
    make p point to (the node p points to) is pointing to

- In C++:

```
ListNode *p = head;
while ((*p).next != NULL)
   p = (*p).next;
```

<==>

```
ListNode *p = head;
while (p->next)
   p = p->next;
```

p=p->next is like i++

Linked List functions: appendNode

- appendNode: adds new node to end of list

```
void NumberList::appendNode(double num) {

   ListNode *newNode;  // To point to the new node

   // Create a new node and store the data in it
   newNode = new ListNode;
   newNode->value = num;
   newNode->next = NULL;


   // continued on next slide . . .
```

Linked List functions: appendNode

- appendNode: continued

```
   // If empty, make head point to new node
   if (!head)
      head = newNode;

   else {

      ListNode *nodePtr;  // To move through the list
      nodePtr = head;     // initialize to start of list

      // traverse list to find last node
      while (nodePtr->next)         //it's not last
         nodePtr = nodePtr->next;   //make it pt to next

      // now nodePtr pts to last node
      // make last node point to newNode
      nodePtr->next = newNode;
   }
}
```


Driver to demo NumberList

- ListDriver.cpp (no output)

```
//ListDriver.cpp: using NumberList

#include "NumberList.h"

int main() {

   // Define the list
   NumberList list;

   // Append some values to the list
   list.appendNode(2.5);
   list.appendNode(7.9);
   list.appendNode(12.6);
   return 0;
}
```

Traversing a Linked List

- Visit each node in a linked list, to
  - display contents, sum data, test data, etc.
- Basic process:

set a pointer to point to what head points to
while pointer is not NULL
  process data of current node
  go to the next node by setting the pointer to
    the pointer field of the current node
end while

Linked List functions: displayList

- displayList()

```
void NumberList::displayList() {
   ListNode *nodePtr;  //ptr to traverse the list

   // start nodePtr at the head of the list
   nodePtr = head;

   // while nodePtr pts to something (not NULL), continue
   while (nodePtr)
   {
      //Display the value in the current node
      cout << nodePtr->value << endl;

      //Move to the next node
      nodePtr = nodePtr->next;
   }

}
```

Driver to demo NumberList

- ListDriver.cpp

```
//ListDriver.cpp: using NumberList

#include "NumberList.h"

int main() {

   // Define the list
   NumberList list;

   // Append some values to the list
   list.appendNode(2.5);
   list.appendNode(7.9);
   list.appendNode(12.6);

   // Display the values in the list.
   list.displayList();
   return 0;
}
```

Output:
2.5
7.9
12.6