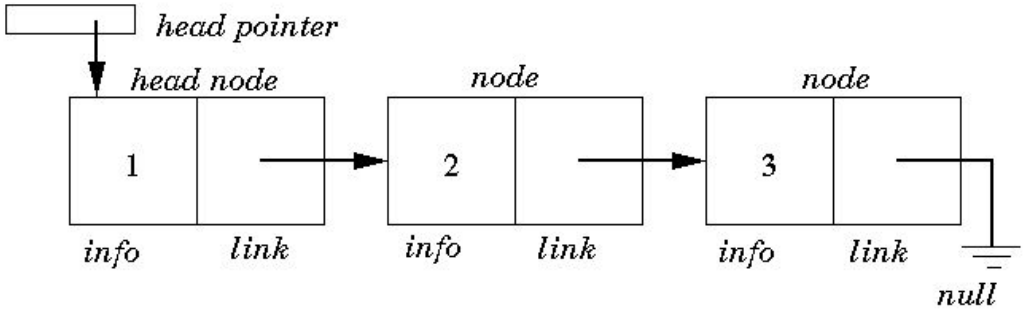
C++ IX

Linked Lists

Linked Lists: What are They?

- It is a data structure: a structured way of storing data
  - Preserves order
- It is a series of linked nodes, each node containing information and a link to the next node



A Linked List

Linked Lists in C++

- Can use either a struct or a class for the nodes
- Use a pointer for next: putting the actual object there will cause . . . ???
  - Think of the difference between how Java and C++ reference objects
- The –> means get the object the pointer is pointing to and access the function/member

```
1  struct node{
2      int x;
3      node *next;
4  }
5
6  int main(){
7      node *root;
8      root = new node;
9
10     root->next = 0;
11     root->x = 5;
12 }
```

Traversing a Linked List

- Use a separate runner node to keep reference to the root node
- Advance the node as seen on right
- Adding a node to the tail requires traversing the list

```
node *root;
node *runner;

root = new node;

root->next = 0;
root->x = 5;

runner = root;

if(runner != 0){
    while (runner->next != 0){
        runner = runner->next;
    }
}
runner->next = new node;
runner = runner->next;
runner->next = 0;
runner-> = 42;
```

Printing a Linked List

Advance through the nodes and print the contents

```
if(runner != 0){
    cout << runner->x << " ";
    while (runner->next != 0){
        cout << runner->x << " ";
        runner = runner->next;
    }
    cout << endl;
}
```

Deleting a Linked List

- You have to traverse the list and delete the nodes individually
- Different from Java: just delete the head and let garbage collection handle it
- When your program ends, your OS will typically get rid of things on your heap

```
43 void deleteList ( Node **head){
44     if (! *head)
45     return;
46
47     Node *curr1 = *head, *curr2 = (*head)->next;
48     while ( ! curr2){
49         delete curr1;
50         curr1 = curr2;
51         curr2 = curr2 -> next;
52     }
53
54     delete curr1;
55     *head = 0;
56 }
```