ECE 2574: Data Structures and Algorithms - Sorted Linked Lists

C. L. Wyatt

Today we will see how to adapt LinkedList into a sorted list implementation.

- Review (Single) Linked List methods
- The protected class scope and internal Nodes
- Reusing LinkedList via inheritance

Recall the Sorted List ADT

A number of objects, not necessarily distinct but of the same type, sorted by thier value.

```
+isEmpty(): boolean
+getLength(): integer
+insertSorted(newEntry: ItemType): void
+removeSorted(entry: ItemType): boolean
+remove(position: integer): boolean
+clear(): void
+getEntry(position: integer): ItemType
+getPosition(entry: ItemType: integer
```

These methods differed from the List ADT

- `+insertSorted(newEntry: ItemType): void`: insert the entry in order
- `+removeSorted(entry: ItemType): boolean`: remove first occurance
- `+getPosition(entry: ItemType: integer`: get position of first occurance or the negated position where it would be

Recall our interface definition

See code `abstract_sorted_list.h`

Reusing LinkedList

To implement insert we would like to have access to the internal node structure and the relevent members in a subclass, but we do not want them to be public.
The answer: make them protected.
See `linked_list.h` and `linked_list.txx`.

Now we can extend the LinkedList class to be a SortedLinkedList

See `sorted_linked_list.h` and `sorted_linked_list.txx`

Next Actions and Reminders

- Read CH pp. 373-378 (Queue ADT)
- There is a warmup for Monday!
- Program 3 is due 10/31.