Hiep Ha

CS-300

Professor Nathan Lebel

Sept. 23, 2024

3-2 Assignment: Linked Lists

The purpose of this program is to manage bid information using a linked list, allowing users to add, search, and remove bids. The linked list structure is chosen because it makes adding and removing items easy.

Key challenges included handling memory properly when deleting nodes to avoid memory leaks and correctly adjusting pointers when inserting or removing bids. To solve these issues, I used a loop to safely delete each node in the destructor and carefully updated pointers to keep the list structure intact during changes.

**Task 1: Create an Internal Structure**

* **Define a Bid structure with the following fields**:
* bidId (string)
* title (string)
* fund (string)
* amount (double, initialized to 0.0)
* **Inside LinkedList class, define housekeeping variables**:
  + head (Node pointer, initialized to null)
  + tail (Node pointer, initialized to null)
  + size (integer, initialized to 0)

**Task 2: Initialize Housekeeping Variables**

* **In LinkedList constructor**:
* Set head = null
* Set tail = null
* Set size = 0

**Task 3: Append Logic**

* **Function Append(bid)**:
* Create a new node with bid data.
* If head is null (list is empty):
* Set head and tail to the new node.
* Else:
* Set tail's next pointer to the new node.
* Set tail to the new node.
* Increment size by 1.

**Task 4: Prepend Logic**

* **Function Prepend(bid)**:
  + Create a new node with bid data.
  + If head is null (list is empty):
    - Set head and tail to the new node.
  + Else:
    - Set the new node’s next pointer to the current head.
    - Set head to the new node.
  + Increment size by 1.

**Task 5: Print Logic**

* **Function PrintList()**:
  + Set current node to head.
  + While current node is not null:
    - Print current node's bidId, title, amount, and fund.
    - Move current to the next node.

**Task 6: Remove Logic**

* **Function Remove(bidId)**:
  + If head is null, return (list is empty).
  + If head's bidId matches bidId:
    - Set a temporary node to head.
    - Set head to head's next node.
    - Delete the temporary node.
    - Decrease size by 1.
    - Return.
  + Set current node to head.
  + While current's next node is not null:
    - If current's next node's bidId matches bidId:
      * Set a temporary node to current's next node.
      * Set current's next to the temporary node's next.
      * Delete the temporary node.
      * Decrease size by 1.
      * Return.
    - Move current to the next node.

**Task 7: Search Logic**

* **Function Search(bidId)**:
  + Set current node to head.
  + While current node is not null:
    - If current node's bidId matches bidId, return the current node's bid.
    - Move current to the next node.
  + If no match is found, return an empty bid.