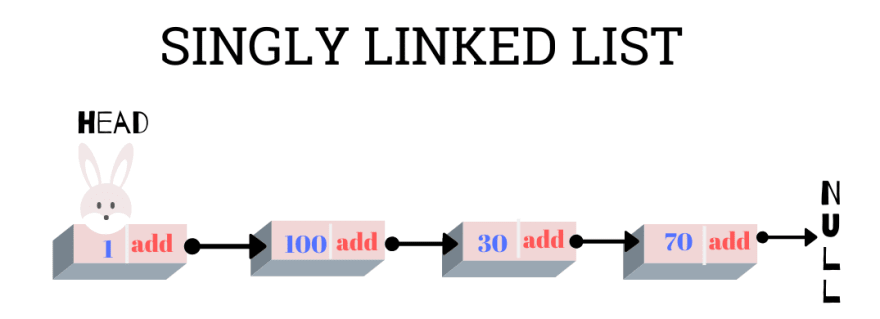
Joshua Hale

July 11, 2024

# CS-300 **3-2 Assignment: Linked Lists**



**Code Reflection**

The primary objective of this assignment was to implement a singly linked list in C++ to manage a collection of bids from a CSV file. The implementation involved adding logic for basic linked list operations, including appending, prepending, printing, removing, and searching for nodes. The provided skeleton code was augmented with methods that facilitated these operations. One of the main challenges was ensuring proper handling of edge cases such as empty lists, removal of head or tail nodes, and maintaining the integrity of the list after each operation. These challenges were addressed by careful pointer manipulation and thorough testing of each function. Additionally, the task required integrating a CSV parser to load bid data from a file, which was accomplished by incorporating the ‘CSVparser’ library. Overall, the project enhanced understanding of linked list data structures and their practical applications in handling dynamic datasets.

**Pseudocode**

**Initialize Linked List**

* Initialize LinkedList:
  + Set head to nullptr
  + Set tail to nullptr
  + Set size to 0

**Append a Bid**

* Append(bid):
  + Create a new node with the given bid
  + If the list is empty (head is nullptr):
    - Set head and tail to the new node
  + Else:
    - Set tail.next to the new node
    - Update tail to the new node
  + Increment size

**Prepend a Bid**

* Prepend(bid):
  + Create a new node with the given bid
  + If the list is empty (head is nullptr):
    - Set head and tail to the new node
  + Else:
    - Set newNode.next to head
    - Update head to the new node
  + Increment size

**Print All Bids**

* PrintList():
  + Set current to head
  + While current is not nullptr:
    - Print current.bid (bid ID, title, amount, fund)
    - Move current to the next node (current = current.next)

**Remove a Bid**

* Remove(bidId):
  + If the list is empty (head is nullptr), return
  + If head.bid.bidId equals bidId:
    - Set temp to head
    - Update head to head.next
    - Delete temp
    - Decrement size
    - Return
  + Set current to head
  + While current.next is not nullptr and current.next.bid.bidId does not equal bidId:
    - Move current to the next node (current = current.next)
  + If current.next is nullptr, return
  + Set temp to current.next
  + Update current.next to temp.next
  + If temp is tail, update tail to current
  + Delete temp
  + Decrement size

**Search for a Bid**

* Search(bidId):
  + Set current to head
  + While current is not nullptr:
    - If current.bid.bidId equals bidId, return current.bid
    - Move current to the next node (current = current.next)
  + Return an empty Bid if not found

**Load Bids from CSV**

* LoadBids(csvPath):
  + Open the CSV file at csvPath
  + For each row in the CSV file:
    - Create a Bid object with the row data (bid ID, title, fund, amount)
    - Append the Bid object to the linked list

This pseudocode captures the logic needed to implement a singly linked list with the required functionalities to manage bids, ensuring a functional and comprehensive implementation.

*Image Reference:*

[*https://dev.to/kauresss/linked-lists-data-structure-and-operations-26md*](https://dev.to/kauresss/linked-lists-data-structure-and-operations-26md)