Southern New Hampshire University

Chris Trimmer

CS-300-T1159 DSA: Analysis and Design

Assignment 3-2: Linked List (Reflection)

22EW1 – 09/13/2022

The purpose of this document is to provide a code reflection of assignment 3-2 which is the linked list implementation of a course scheduler. Assignment 3-2 is focused on linked lists. Linked lists are one of the most popular and versatile data structures. They have constant time (O(1)) insertion at the front and back, and O(n) time for most other operations. Something that makes a linked list so popular, is the versatility of memory management. A linked list stores object data as a node, and each node carries a pointer to a memory address of its neighbor. Objects linked in this way can efficiently be stored anywhere in the heap – the memory does not need to be contiguous as it is with arrays. Furthermore, when inserting and deleting nodes, we can simply re-assign pointers instead of shifting nodes, which makes these operations much more efficient compared to insertions and deletions in arrays and vectors.

Code Reflection

Our code for this assignment includes the core operations that are part of a linked list: append, prepend, search, remove, tracking the list size, and printing the list. Each operation is coded as a function that performs the operation on the list of nodes. The append function is used to add a node to the back of the list. The prepend function is used to add a node to the front of the list. The search function takes a key as an argument. It walks through the list one node at a time, comparing the key to the bid id until it finds a match, or until no match is found. The remove function takes a key as an argument, and then manipulates pointers to disconnect the desired node, and then re-connecting the list, effectively deleting the node. We track the size of the list using a size variable that is incremented and decremented as we add or delete nodes from the list. Finally, the print function walks the list one node at a time and calls the displayBid method for each node.

Overall, this was a fun assignment. I didn't encounter any trouble or issues with implementing the linked list. I followed the steps outlined in our assignment and tested each step as I coded. This helped ensure that I found bugs and corrected any problems before moving to each subsequent step.

Note that I added a menu option to allow for prepending a node to the list. A customer can append to front and end of the list. I also modified the option for deleting a node. A customer can enter a bid id for the bid they want to delete. If the bid id is not in the list, the program will report this as output to the customer.

Chris Trimmer

CS-300-T1159 DSA: Analysis and Design

Assignment 3-2: Linked List (Pseudocode)

22EW1 – 09/13/2022

The purpose of this document is to provide pseudocode of the main functions used in the linked list code. Assignment 3-2 is focused on linked lists. Linked lists are one of the most popular and versatile data structures. They have constant time (O(1)) insertion at the front and back, and O(n) time for most other operations. Something that makes a linked list so popular, is the versatility of memory management. A linked list stores object data as a node, and each node carries a pointer to a memory address of its neighbor. Objects linked in this way can efficiently be stored anywhere in the heap – the memory does not need to be contiguous as it is with arrays. Furthermore, when inserting and deleting nodes, we can simply re-assign pointers instead of shifting nodes, which makes these operations much more efficient compared to insertions and deletions in arrays and vectors.

Pseudocode

Menu Loop

The menu loop is contained within main. The loop enables the user to continue making menu selections until they exit the program by choosing option 9.

```
Get user input
  While user input is not equal to 9
      Display menu options
          Switch (user input)
              Case 1:
                  Set up timer
                  Append Node to list
                  Display timer results
                  break
              Case 2:
                  Set up timer
                  Prepend Node to list
                  Display timer results
                  Break
              Case 3:
                  Start timer
                  Load the bids from the .csv file
                  Store results bids as objects in list
                  Display timer results
                  Break
              Case 4:
                  Start timer
                  Call Display function to print the list
                  Display timer results
                  Break
```

```
        Case 5:
            Start timer
            Get bid id as input from user
            **Call Search using the input from user**
            Display timer results
            Break
        Case 6:
            Start timer
            Get bid id as input from user
            **Call Remove using the input from user**
            Display timer results
            Break
        Default:
            Any input not valid for menu
            Break
```

// Linked list constructor

```
LinkedList::LinkedList LinkedList() {
  Set head to nullptr
  Set tail to nullptr
}
```

// Linked list destructor

```
LinkedList::LinkedList ~LinkedList() {
  Set pointer to head of list
  Create temp pointer

  Loop through the list until the current pointer to the head is null
    Set temp to the current pointer
    Set current pointer to the next pointer of current
    Release the memory of temp
}
```

// insert a node at the back of the list

```
Void LinkedList::Append(Bid bid) {

  Create newNode* with bid as argument to constructor

  If (head is empty) {

    Set head to newNode
    Set tail to newNode
  }
  Else {
    Set tail->next to newNode
    Set tail to newNode
  }

  Increment size by 1

} End Append function
```

// insert a node at the front of the list

```
Void LinkedList::Prepend(Bid bid) {

  If (list is empty) {

    // re-use code from Append function
    Call Append with bid as argument
    Return to caller
  }

  Create temporaryNode* with bid as argument to constructor

  Set temp->next to head
  Set head to the tempNode

  Increment size by 1

} end Prepend function
```

// print the list

```
void LinkedList::PrintList() {

  Set a pointer to the head of the list

  Loop through list until the current node is not nullptr
    Print the bid data
    Set current node pointer to the next node

}
```

Southern New Hampshire University

```
// remove a node from the list using bidId as key
Void LinkedList::Remove(string bidId) {

  // if the list is empty, simply return as there is nothing to remove
  If (head is null)
    Return

  // if bidId matches the head, set the next node as the head
  If (bidId is equal to head->bid.bidId) {

    Save head to a temp*
    Set head to head->next node
    Delete the temp node and set it to null

    Decrement size of list by 1
    Return to caller
  }

  // set up temp pointers that will track the previous and current nodes
  Set prev to this->head
  Set curr to this->head->next

  // handle case where tail needs to be removed
  If (bidId is equal to the tail->bid.bidId) {

    // walk the list with the pointers until the end
    While (curr->next is not equal to null) {
      Prev = prev->next
      Curr = curr->next
    }

    Set Prev->next to nullptr
    Set tail to prev

    Delete curr and set it to null

    Decrement size by 1

    return
  }

  // If node to remove is somewhere between first and last in list
  // traverse the list until we find it
  While (curr->next is not null) {
```

Southern New Hampshire University

```
    If (bidId is equal to curr->bid.bidId) {

      Set temp node to curr
      Set prev->next to temp->next
      Delete temp and set it to null
      Decrement size by 1
      Return
    }

    Else continue walking the list {
      Set prev to prev->next
      Set curr to curr->next
    }

  }

} end Remove function
```

// search function

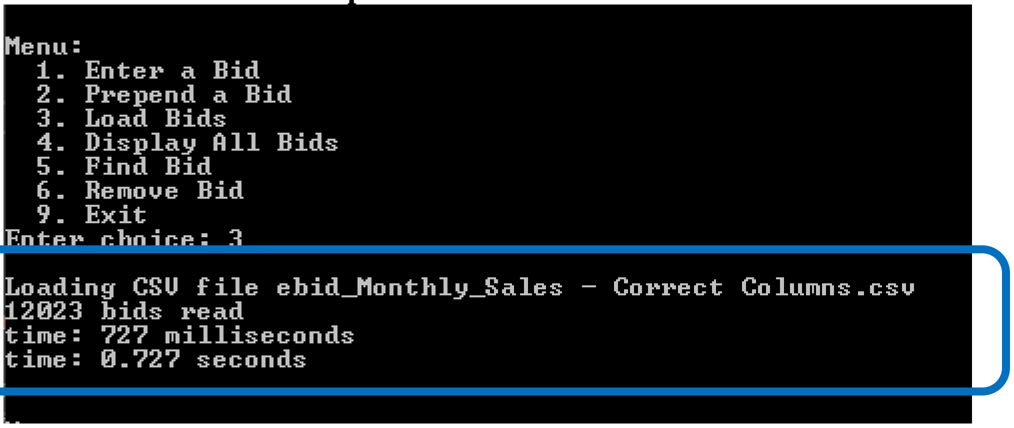
```
Void LinkedList::Search(string bidId) {

  if head is null, then return to caller

  Create currentNode equal to head

  // traverse the list looking for match to bidId
  While (currentNode not equal to null) {
    If (bidId is equal to currNode->bid.bidId) {
      Return currNode->bid
    }

    // continue traversing list
    Set currentNode equal to currNode->next
  }

} end of Search function
```

// function to return size of list

```
Int LinkedList::Size() {
  Return size of list
} end of Size function
```

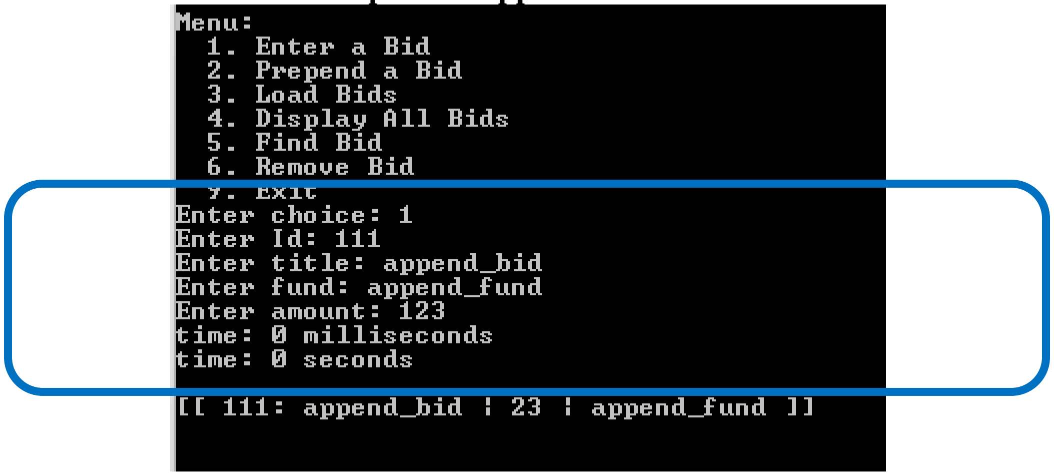
Screenshots

Speed of Load function



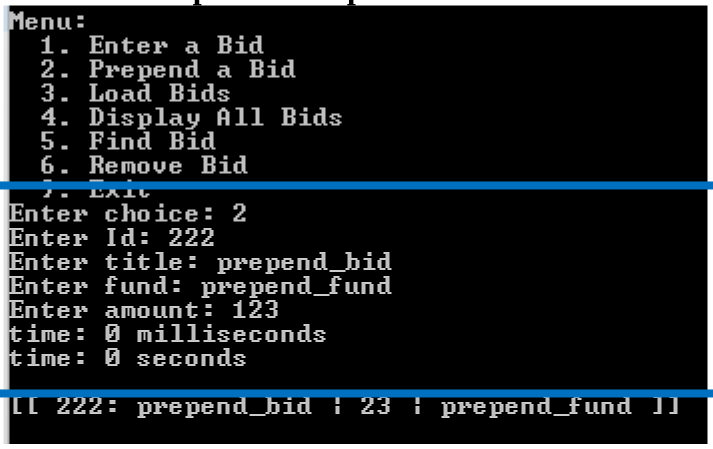
```
Menu:
  1. Enter a Bid
  2. Prepend a Bid
  3. Load Bids
  4. Display All Bids
  5. Find Bid
  6. Remove Bid
  9. Exit
Enter choice: 3

Loading CSV file ebid_Monthly_Sales - Correct Columns.csv
12023 bids read
time: 727 milliseconds
time: 0.727 seconds
```

Speed of Append function



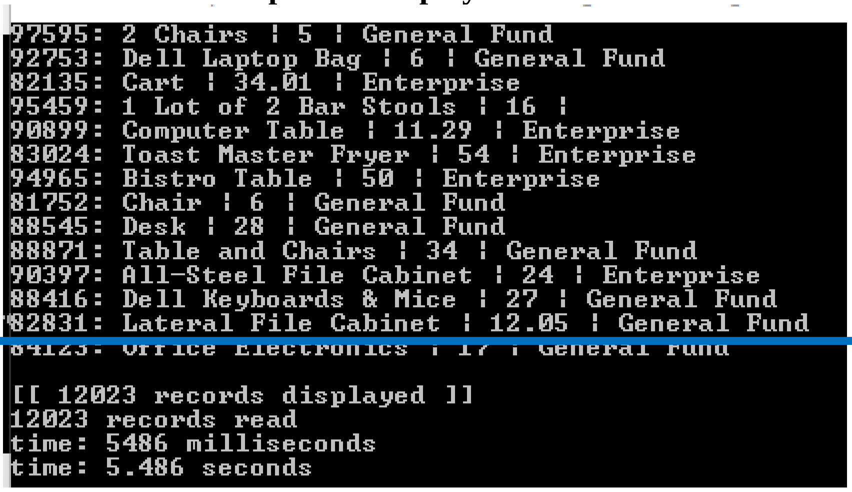
```
Menu:
  1. Enter a Bid
  2. Prepend a Bid
  3. Load Bids
  4. Display All Bids
  5. Find Bid
  6. Remove Bid
  9. Exit
Enter choice: 1
Enter Id: 111
Enter title: append_bid
Enter fund: append_fund
Enter amount: 123
time: 0 milliseconds
time: 0 seconds

[[ 111: append_bid | 23 | append_fund ]]
```

Speed of Prepend function



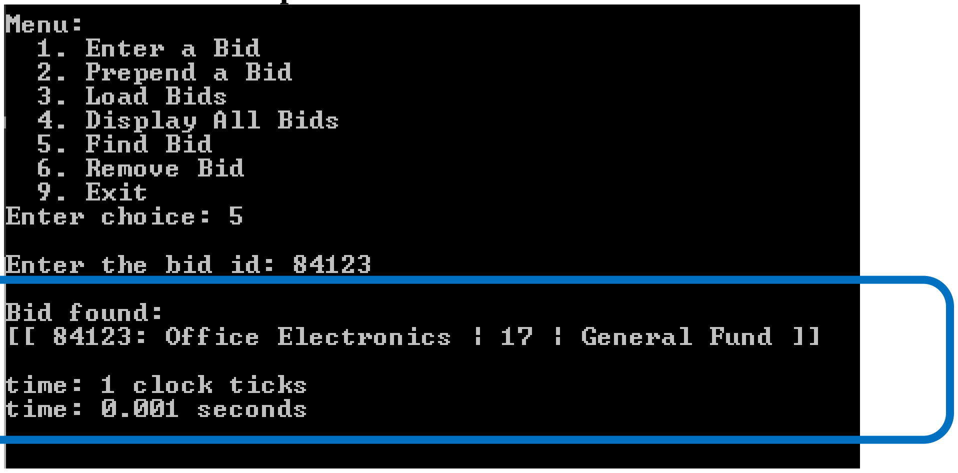
```
Menu:
  1. Enter a Bid
  2. Prepend a Bid
  3. Load Bids
  4. Display All Bids
  5. Find Bid
  6. Remove Bid
  7. Exit
Enter choice: 2
Enter Id: 222
Enter title: prepend_bid
Enter fund: prepend_fund
Enter amount: 123
time: 0 milliseconds
time: 0 seconds

[[ 222: prepend_bid | 23 | prepend_fund ]]
```

Speed of Display function



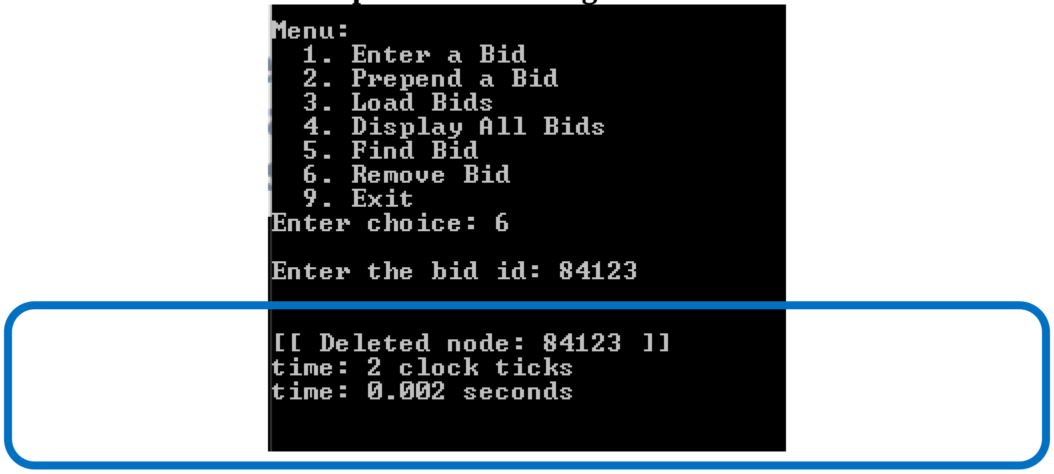
```
97595: 2 Chairs | 5 | General Fund
92753: Dell Laptop Bag | 6 | General Fund
82135: Cart | 34.01 | Enterprise
95459: 1 Lot of 2 Bar Stools | 16 |
90899: Computer Table | 11.29 | Enterprise
83024: Toast Master Fryer | 54 | Enterprise
94965: Bistro Table | 50 | Enterprise
81752: Chair | 6 | General Fund
88545: Desk | 28 | General Fund
88871: Table and Chairs | 34 | General Fund
90397: All-Steel File Cabinet | 24 | Enterprise
88416: Dell Keyboards & Mice | 27 | General Fund
82831: Lateral File Cabinet | 12.05 | General Fund
84123: Office Electronics | 17 | General Fund

[[ 12023 records displayed ]]
12023 records read
time: 5486 milliseconds
time: 5.486 seconds
```

Speed of Search function



```
Menu:
  1. Enter a Bid
  2. Prepend a Bid
  3. Load Bids
  4. Display All Bids
  5. Find Bid
  6. Remove Bid
  9. Exit
Enter choice: 5

Enter the bid id: 84123

Bid found:
[[ 84123: Office Electronics | 17 | General Fund ]]

time: 1 clock ticks
time: 0.001 seconds
```

Speed of Removing the node



```
Menu:
  1. Enter a Bid
  2. Prepend a Bid
  3. Load Bids
  4. Display All Bids
  5. Find Bid
  6. Remove Bid
  9. Exit
Enter choice: 6

Enter the bid id: 84123


[[ Deleted node: 84123 ]]
time: 2 clock ticks
time: 0.002 seconds
```