Daniel Loranger

CS-300

Week 4 assignment

* Code Reflection – Having just learned about the linked lists, this work went quite smoothly as I have also recently had practice with dealing with modulo bucketing, so everything was quite clear what was needed. The use of an table for listing the modulo sorted entries was also clear.

1 public declared function (Size()) was removed from the class as it was not used in the code anywhere and could lead to errors for users down the road.

* Pseudo-code

For brevity, existing static functions for testing are not detailed, only the functions that are edited are covered with below pseudo code. Exception is the main function which is shown for flow understanding inclusively.

* + Declare variables
  + Define class including constructor / destructor
    - Constructor (fixme 1)
      * Update Table size (from default value )
    - Destructor (fixme 2)
      * Erase the vector node beginning
  + HashTable::hash(int) (fixme 3)
    - Calculate a hash key from the input int and the tableSize variable;
  + HashTable::Insert (bid) (fixme 4)
    - Retrieve the bid.bidID hashed value as the key
    - If node is empty
      * assign bid as the first entry in the table at index (key)
    - Else (node is not empty)
      * Iterate thru the list to find the end of the list
      * Append the new entry to the tail of the list
  + HashTable::PrintAll() (fixme 5)
    - For i less than tableSize
      * If node key != UNIT\_MAX (empty)
        + Print “Key “
        + Print key,bidID,title,amount,fund, endl
      * Note: the formatting was not explicitly specified, formatting was assumed to be word wrapped in the guidelines. As such, simplest formatting output was used.
  + HashTable::Remove(bidID) (fixme 6)
    - Handle empty entry
    - If head is the desired entry
      * head is also the tail
        + flush the key to null values
      * head is not the tail
        + update next pointer
    - else (head is not the desired entry)
      * start at the list beginning
      * while not a match( looking ahead 1 position for a matched bidID)
        + if end of list

return

* + - * (match found)
        + If node.next is the tail

Set current node to be the tail

* + - * + Else

Set current node.next = node.next.next to skip the found node.

* + HashTable::Search (string) (fixme 7)
    - Calculate the key for the input string
    - Create a cursor for the node at the given key
    - if cursor is NULL
      * return empty bid
    - else cursor is NOT null
      * while bidID != “”
        + if successfully found

return the current bid

* + - * + increment pointer
    - return empty bid (never found a match)
* Static functions (unchanged)
  + displayBid()
  + loadBids()
  + strToDouble
  + Main ()
    - Handle command line arguments
    - declare an empty time place holder
    - declare an empty hashtable
    - declare an empty bid
    - declare an empty bidtable
    - while loop
      * display user menu
      * handle user selection (case struct)