4-2 Assignment: Hash Tables

Michael Wood

Southern New Hampshire University

CS-300: Analysis and Design

Professor Michael Rissover

9/29/2024

**Code Reflection:**

This program's objective is to load a CSV file containing bid information that the user may view and modify. Bids can be found, removed, and added by users. The software stores bids in a hash table. The assignment this time was a little bit different than the others. Key/value pairs are stored in a hash table. The value in this instance is the bid information, and the key that will be mapped to a list location or "index" is the bidID. After that, you can use the key to access the data kept in the hash table and carry out several operations. The modulo operator "%" and the table size are used to "hashed" or "mapped" the key to an index point. This is because the remaining portion of the calculation serves as the index position for the bid data storage. Additionally, each element will have a unique index created by the bidID, which is a unique number. Nevertheless, as the remainder for various hashed keys may occasionally be the same, collisions may occur. A technique called chaining is used to lessen collisions caused by key/value pairs being allocated to the same place.

I had issues with loading the CSV file eBid\_Monthly\_Sales. I kept getting Unhandled exception at 0x00007FFC98D4FE4C in HashTable.exe: in the CSVparser.cpp line 33. I searched high and low and could not find my mistake or how to fix this issue. My HashTable.cpp did not throw any errors, so I wasn’t sure where to go to fix this issue.

START PROGRAM

WHILE choice IS NOT equal to 9

EXECUTE

PRINT user menu:

1. Load Bids
2. Display All Bids
3. Find Bid
4. Remove Bid
5. Insert Bid
6. Exit

READ IN user input and store to variable choice for menu selection.

CHOICE 1 - (Load Bids)

SET variable ticks equal to clock() method. Stores starting clock tick.

INITIALIZE csv file

PARSE csv file and stores data into bid object

DEFINE vector to store collection of bids

RETURN bids

PRINT size of bids

CALCULATE elapsed time

Clock() – ticks

PRINT elapsed time

CHOICE 2 (Display All Bids)

INVOKE PrintAll() // Loops through hash table and prints all elements

BREAK;

CHOICE 3 - (Find Bid)

SET variable ticks equal to clock() method. Stores starting clock tick.

SET bid equal to Search() method and pass in bidkey as a parameter.

INVOKE Search()

Search() will do the following:

SET key equal to hashed bidID

SET new pointer called “node” equal to reference of nodes element at keyth position.

IF node IS NOT equal to null pointer AND node key IS NOT equal to UINT\_MAX AND node bidID compared to passed in bidID is a match

{

RETURN node bidID;

}

IF node IS EQUAL to null pointer OR node key IS equal to UINT\_MAX

{

RETURN bid;

}

WHILE node IS NOT equal to null pointer

{

IF node key IS NOT equal to UINT\_MAX AND node bidID compare bidID IS a match

{

RETURN node bidID;

}

SET node equal to next node;

}

RETURN bid;

}

CALCULATE elapsed time

IF bidId IS NOT empty

{

INVOKE displayBid() method and pass in bid;

}

ELSE

{

PRINT not found message;

}

PRINT clock ticks

PRINT clock ticks per second

BREAK;

CHOICE 4 - (Remove Bid)

INVOKE Remove() method with passed in bidId;

Remove() method does the following:

SET key equal to hashed bidID

ERASE node starting at the beginning with the hashed key

RESIZE nodes table

CHOICE 5 - (Insert Bid)

CREATE getBid() method as global method

GetBid() method obtains input from user.

SET bid equal to getBid() method

INVOKE Insert() method and pass in bid

{

Insert method does the following:

SET key to hashed bidID

IF the nodes keys at the keyth position IS equal to UINT\_MAX

{

CREATE new node

SET new node equal to Node with passed in bid and key

SET nodes at keyth position equal to new node

}

ELSE

{

CREATE new pointer currNode // to store current node

SET currNode equal to reference nodes at keyth position

CREATE new pointer newNode // to store NEW node

SET newNode equal to a new Node with passed in bid and key

WHILE next currNode IS NOT equal to NULL

{

POINT currNode to the next currNode

}

SET currNode next equal to the newNode

RESIZE nodes table

}

}

INVOKE displayBid() and pass in bid

BREAK;

}

}

ELSE

PRINT “Goodbye”

RETURN 0

}

END PROGRAM