Gentian Hoxha

[Gentian.hoxha@snhu.edu](mailto:Gentian.hoxha@snhu.edu)

4/16/2024

**Reflection**

The provided C++ program is designed to manage a collection of bids using a hash table, facilitating efficient operations such as insertion, deletion, and searching. This hash table, which incorporates structured data for bid information and hash table nodes, effectively interacts with external data sources by reading bids from a CSV file.

Developing this hash table required leveraging modern C++ features like vectors, dynamic memory management, and robust exception handling. During development, several challenges arose, including designing an effective hash function to minimize collisions and prevent performance degradation. This was achieved by applying a modulo operation on keys derived from bid IDs to ensure uniform distribution across the table.

Another key challenge was collision handling. Chaining was implemented as a collision resolution strategy, necessitating careful management of dynamic node creation and linking. Additionally, memory management was critical to prevent memory leaks during dynamic additions or deletions. A destructor was implemented to ensure that allocated memory was properly freed when the hash table was destroyed.

Lastly, parsing CSV files introduced challenges related to data formatting and errors. The `CSVparser` library was integrated to robustly handle these files, and exception handling mechanisms were employed to manage runtime errors during file parsing.

In summary, the development of this hash table in C++ provided practical experience in handling real-world data, reinforcing theoretical concepts and demonstrating effective problem-solving through advanced programming techniques

**Pseudocode**

This is a smaller detailed pseudocode of the assignment since in the other assignment we were asked to be more explicit.

class HashTable

properties:

nodes (vector of Node)

setSize (unsigned int)

method initialize(size)

setSize = size

nodes.resize(setSize)

method hash(key)

return key % setSize

method insert(bid)

key = hash(convert bidId to integer)

if nodes at key is null

create new node with bid

else

find the end of the chain and add new node

method printAll()

for each node in nodes

if node has bid

display bid

method remove(bidId)

key = hash(convert bidId to integer)

locate the correct node in the chain and remove it

method search(bidId)

key = hash(convert bidId to integer)

find node in chain matching bidId

return bid if found, else return empty bid

main

create hashTable

while user does not choose to exit

if choice is load bids

read bids from CSV and insert into hashTable

else if choice is display all bids

print all bids in hashTable

else if choice is find bid

search for bid in hashTable and display

else if choice is remove bid

remove bid from hashTable

In summary, the development of this hash table in C++ provided practical experience in handling real-world data, reinforcing theoretical concepts and demonstrating effective problem-solving through advanced programming techniques.