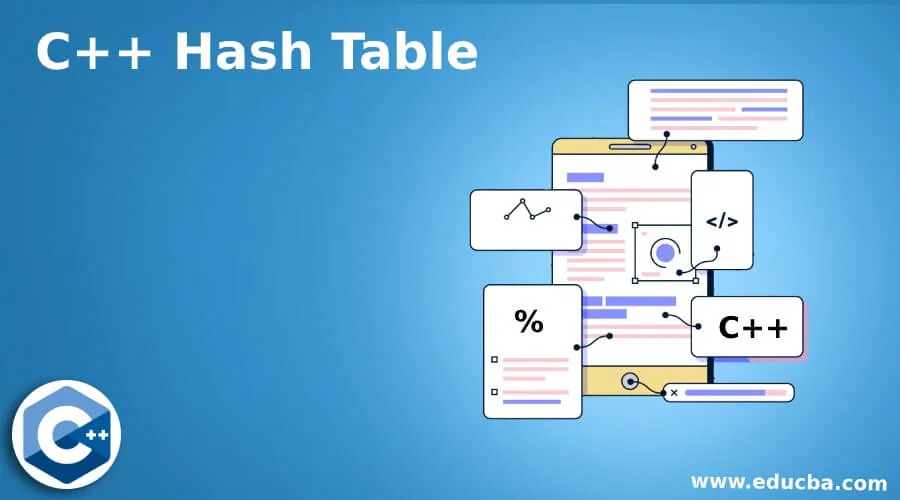
Joshua Hale

July 11, 2024

CS300

4-2 Assignment: Hash Tables



**Code Reflection**

**Purpose of the Code**

The primary purpose of this project is to implement a hash table with chaining to manage bids from property auctions. The hash table efficiently handles the storage, retrieval, and deletion of bid records using unique bid IDs. This project includes functionalities to load bids from a CSV file, display all bids, find a specific bid, and remove a bid.

**Techniques Implemented**

1. **Hash Table with Chaining**: The hash table uses chaining to handle collisions. Each bucket in the hash table is a linked list that stores nodes containing bid data.
2. **CSV Parsing**: The CSV parser reads auction bid data from a CSV file and populates the hash table.
3. **Menu-Driven Program**: The program provides a user-friendly menu to load bids, display all bids, find a bid, and remove a bid.

**Challenges Encountered and Approaches to Overcome**

1. **Handling Collisions**: Ensuring that the hash table handles collisions correctly was challenging. This was overcome by implementing linked lists for each bucket, allowing multiple entries to be stored at the same hash index.
2. **CSV Parsing**: Reading and correctly parsing a large CSV file with various data types required careful handling of string operations and error checking.
3. **Memory Management**: Ensuring that all dynamically allocated memory is properly managed and freed was critical to avoid memory leaks. This was handled by implementing a destructor to clean up the linked lists.

**Pseudocode**

**Purpose**

This program helps manage bids from property auctions. It can load bids from a file, show all bids, find a specific bid, and remove a bid.

**Steps**

1. **Initialize Program**

* Start the program and set up a place to store bids.

1. **Menu Options**

* Show a menu with options: Load Bids, Show All Bids, Find a Bid, Remove a Bid, Exit.

1. **Load Bids**
2. If the user chooses "Load Bids":
   * + Ask for the file containing the bids.
     + Read each bid from the file.
     + Store each bid in the program's storage.
3. **Show All Bids**
4. If the user chooses "Show All Bids":
   * + Go through all stored bids.
     + Display each bid's details (ID, title, amount, fund).
5. **Find a Bid**
6. If the user chooses "Find a Bid":

* Ask for the bid ID to search for.
* Look through the stored bids.
* If the bid is found, display its details.
* If not found, display a message saying the bid is not found.

1. **Remove a Bid**
2. If the user chooses "Remove a Bid":

* Ask for the bid ID to remove.
* Look through the stored bids.
* If the bid is found, remove it from storage.
* If not found, display a message saying the bid is not found.

1. **Exit Program**
   * If the user chooses "Exit":
     + End the program.

**Menu Process**

1. Show menu options.
2. Get user's choice.
3. Depending on the choice, perform the corresponding action:

* Load bids from a file.
* Show all stored bids.
* Find and show a specific bid.
* Remove a specific bid.
* Exit the program.

**Example Workflow**

1. **User starts the program**.
2. **Menu is displayed**:

1. Load Bids

2. Show All Bids

3. Find a Bid

4. Remove a Bid

9. Exit

Enter choice: \_

1. **User chooses "Load Bids"**:

* Program asks for the file name.
* Program reads bids from the file and stores them.

1. **User chooses "Show All Bids"**:

* Program displays all stored bids.

1. **User chooses "Find a Bid"**:

* Program asks for the bid ID.
* Program searches for the bid and displays it if found.

1. **User chooses "Remove a Bid"**:

* Program asks for the bid ID.
* Program searches for the bid and removes it if found.

1. **User chooses "Exit"**:

* Program ends.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Specifications and Correctness**

The source code meets the specifications by implementing the hash table with the following functionalities:

1. **Load Bids**: Reads and inserts bid data from a CSV file into the hash table.
2. **Display All Bids**: Iterates through the hash table and prints all stored bids.
3. **Find Bid**: Searches for a specific bid by ID and displays it.
4. **Remove Bid**: Removes a bid from the hash table by ID.

The code produces the correct output as per the provided data and problem definition. It is fully functional and executes without errors, handling edge cases such as non-existent bids and empty buckets.

**Annotation and Documentation**

The code includes comments explaining the purpose of each section and the approach taken:

* **Global Definitions**: Constants and forward declarations.
* **Bid Structure**: Definition of the Bid structure to hold bid information.
* **HashTable Class**:
  + **Private Members**: Definitions for node structure and hash table storage.
  + **Public Methods**: Constructors, destructor, and methods for inserting, printing, removing, and searching bids.
* **Helper Functions**:
  + **displayBid**: Function to display bid information.
  + **loadBids**: Function to load bids from a CSV file.
  + **strToDouble**: Function to convert strings to doubles.

**Modular and Reusable**

The code is modular and reusable, adhering to the single responsibility principle:

* **HashTable Class**: Encapsulates all functionalities related to the hash table.
* **Helper Functions**: Separate functions for displaying bids and loading bids from a CSV file.
* **Main Function**: Manages the program flow and user interactions.

**Readability**

The code is readable and follows these guidelines:

* **Consistent Whitespace and Indentation**: Properly formatted to separate distinct parts of the code.
* **Explicit Variable Names**: Clear and consistent naming conventions.
* **Organized Structure**: Logical separation of responsibilities and grouping related code into blocks.

*Image Reference:*

<https://www.educba.com/c-plus-plus-hash-table/>