Laboratory 2 – LFTC

Documentation

The HashTable is an implementation of a simple hash table data structure that supports basic operations such as adding, finding, and deleting elements. It is designed to store and retrieve elements of generic type T. This implementation uses coalesced chaining to handle collisions, meaning that when a collision appears, we search for the first available position from the left.

for the HashTable we have:

```
template<class T>
struct HashEntry {
   T token;
   int next = -1;
};
```

- The HashEntry structure represents an entry in the hash table.
- It holds a generic type T representing the element (in this case, a string token).
- The next field is an index pointing to the next colliding element within the hash table.

```
template<class T>
class HashTable {
private:
   vector<HashEntry<T>> table;
   int size = 103;
   int hashFn(T element);
   int findNextPositionAvailable();
public:
   HashTable();
   HashTable(HashTable &hTable);
   int addValue(T element);
   int findPosition(T element);
   int containsValue(T element);
   int deleteValue(T element);
};
```

- The CoalescedHashTable class is a generic hash table implementation using the coalesced hashing technique.
- It includes functions for adding, finding, checking existence, and deleting elements.
- Collisions are resolved by linking colliding elements within the hash table using the next field.

---

int HashTable<T>::hashFn(T element) {}

- Computes the hash value for the given element using the sum of ASCII values of characters

---

Int HashTable<T>::findNextPositionAvailable () {}

- Finds the next available position in the hashtable

Int HashTable<T>::addValue(T element) {}

- Adds a value to the hash table, handling collisions with chaining

---

Int HashTable<T>::findPosition(T element) {}

- Finds the position of an element in the hash table

Int HashTable<T>::contrainsValue(T element) {}

- Checks if the hashtable contains a specific value

Int HashTable<T>::deleteValue(T element) {}

- Deletes a value from the hashtable

For the SymbolTable:

```
Class SymbolTable {
private:
        HashTable<string> hashTable;
Public:
        SymbolTable();
        Int findPosition(string token);
        Int deleteToken(string token);
        Int addToken(string token);
};
```

- The SymbolTable class is a specialization of the HashTable for storing and managing string tokens

---

```
SymbolTable::SymbolTable() {}
```

- Initializes a new SymbolTable instance

```
Int SymbolTable::findPosition(String token) {}
```

- Finds the position of a token in the symbol table

```
Int SymbolTable::deleteToken(string token) {}
```

- Deletes a token from the symbol table

```
Int SymbolTable::addToken(string token) {}
```

- Adds a token to the symbol table