FULL SAIL UNIVERSITY

Data Structures and Algorithms

Lab 6: unordered_map

Overview

The **unordered_map** class is the C++ implementation of a Dictionary (hashmap). The implementation is similar to the one written in the previous lab, with some key differences to make it more flexible (but more complex under the hood).

Instead of using a fixed-size dynamic array, the **unordered_map** uses a vector. This allows for the number of buckets to be expanded if the existing buckets get too full. This requires all of the keys to be rehashed, which is handled internally. It also uses the **std::pair** included in the STL.

In this lab, the **unordered_map** will be used to store information for the board game, Scrabble. In this game, players create words out of letter tiles that each have a point value associated with them. There is a file containing all of the valid words in Scrabble, and you will be calculating the value for each of those words.

Things to Review

- Text file I/O
- ASCII values

New Topics

- unordered_map
- pair

Data Members

mLetterValues	Array of 26 values, representing the individual values for each letter
mScrabbleMap	An unordered_map that stores the words as the key, and the total points as the value

Methods

PopulateLetterValues

- Copy the values of the passed-in array into the **mLetterValues** data member
    - Since the methods are explicitly written for scrabble, you can assume that this is an array of 26 elements

GetLetterValue

- Returns the value of a particular letter
- The parameter received will **always** be an upper-case letter
    - Will need to offset this into the array element range and return that value from the array

GetWordValue

- Returns the total value for a word
- *Use a previous method to help calculate this*

CreatePair

- Creates and returns an **std::pair** with the word as the key, and the total for that word as the value

LoadWords

- The string passed in contains the name of a text file that contains all of the words for Scrabble
    - There is one word per line, and each word is already in all upper-case letters
- Read each word in the file, and create a pair that stores the word and its score

- Add each pair to the unordered_map data member

FindValueInMap

- Finds a word in the map and returns the value associated with it
- If the word is not found, return -1
- Use the **find** method of the unordered_map to ensure O(1) complexity