Program Documentation

CIS22C Team 3 – Cupertino Restaurants

Christina Sok
Yenni Chu
James Agua

> Individual Assignments:

Christina Sok

- Main
- Hash.h and Hash.cpp
- ListHead.h and ListHead.cpp
- CollisionTable.h and CollisionTable.cpp
- RestaurantInfo.h and RestaurantInfo.cpp
- Test Plan
- Presentation Outline
- Output (Based on Demonstration Test Plan)
- Demonstration Test Plan
- Input file

Yenni Chu

- BinaryNode.h and BinaryNode.cpp
- BinarySearchTree.h and BinarySearchTree.cpp (all fcns. Exept save to output file function)
- Output (Based on Demonstration Test Plan)
- Demonstration Test Plan

James Agua

- Output file + save to output function (From BST)
- BinaryNode/BST comments/definitions
- Documentation Description

> Note:

- Output: commented at end of main.cpp file
- Input file: “Restaurants.txt”
- Output file: “RestaurantsOutfile.txt”

> Descriptions:

Main:

Main contains the calls to invoke the program and its core routines that allow processing the data into the structure. Main loads the file to be processed, calls the routines to get the users inputs and saves the file on exit. Other functions in the file main.cpp are the utilities that allow the user to interact with the program. The program gets the users input and tests for validity. If the input is a valid command the input will be handed to a operations manager to launch a specific action. This process will continue until the user enters Q for quit. A menu will be presented to the user to allow (s)he to make one of several choices available to the program. The choices are insert data, delete data, search by street number or by restaurant name, list the data by hash sequence or by name sequence, print the indented BST, print the hash statistics, save the BST to file, quit and show the menu. Each of these choices has a manager defined within main except for the print and save options. Once the user quits, the updated restaurant list will be automatically saved to an output file.

Hash: (Hash Array)

The purpose of the hash array is to store a pointer to an object (a restaurant) using a hash key as an index. The array will also keep count of the total number of restaurants stored in the array.

Each index in the hash array holds a restaurant object, a count of the total of restaurants in that table, and a pointer to a collision table linked-list.

The index is calculated using the restaurant's unique key, it's address number. (See Hash: Hash Key for more information).

The hash array allows the user to search for restaurants and delete restaurants with an address number.

Hash: Hash Key

The hash key is derived from the address number of the restaurant where each number is added to the subsequent number. A sub-sum is also created during this process. The sub-sum is the sum of the first integer with the third integer.

If the sum of the integers is a single digit and is divisible by two, it's respective hash index number will be that sum appended to it's sum.

If the sum is not divisible by two and the sub-sum is a single digit number, the respective hash index number is the sum appended with its sub-sum.

Otherwise (if the sum is a single digit, is not divisible by two, and it's sub-sum is not a single digit), the respective hash index number is the sub-sum append to the sub-sum.

Finally, if the sum is a two-digit number, the respective hash index number is the sum of the sum and the sub sum divided by two.

Collision Table:

The collision table is a linked single linked-list that is used to store pointers to objects (restaurants) when the hash of the address of the restaurant happens to have calculated to an index in the hash array that is occupied. The algorithm to test for a populated hash node determines if a list node in the collision table should be created and if so if it is a new list member or part of an existing list. The nodes are added to the collision table sequentially.

Binary Node:

The binary node is the primitive object of the binary search tree. The node has pointers for left and right children as well as a pointer to the object restaurant.

Binary Search Tree:

The binary search tree is a BST structure that provides methods for managing the binary search tree as well as maintaining the proper structure required to properly define the tree as a BST.

The BST allows the user to search by name and delete by name (as opposed to the hash where you can only search and delete by number). This gives the user more options.

The BST also has functions to print an indented list. If the user wants to save to file or at the end of the program when the restaurants are automatically saved, the save to output function is called from the BST.

Restaurants:

A restaurant is the primary object of this application. A restaurant has attributes for the name, street number, street name and type and provides access for its data. The restaurant name is it's secondary key (for BST) while it's street number is the primary key (for hash array).

List Head:

The list head class manages the relationship between the two primary data structures (hash array and binary search tree) the list head also contains the count of the objects and the size of the hash array.