Alex Ko
Fyk211

Linear Hashing Project - Poll Monitor

Description

This is a C program that uses linear hashing to monitor votes across different people, and it provides functionalities such as quick look up, registering vote statutes, bulk vote, check the number of votes, and organize votes in a zip-based manner.

Design Choices

1. Hash Table
    a. My hash table is a dynamic array of buckets (LinkedLists), each of which can contain max *bucketentries* nodes. Bucketentreis is taken as an argument from the user when they first start the program.

2. LinkedList
    a. Each bucket is a LinkedList that has a *head* node, a pointer called *page* that points to the overflow page that follows, and a size integer representing the size of the current node (up to *bucketentries*).

3. Node
    a. Each LinkedList can contain Nodes. A Node have members such as pin, a pointer called *Next* that points to the next node, and another pointer called *data* that points to the Info struct.

4. Info
    a. I also created a struct called Info that stores the personal information of each person, including pin, first name, last name, zip, and their vote status (initially set to "N").

5. ZipList
    a. I also created another struct called ZipList that represents the people who voted yes organized by their zip code. A ZipList has a *head* node, a size integer. This ZipList can contain as many ZipNodes as possible

6. ZipNode
    a. A ZipNode is a struct that acts as each node in the ZipList. A ZipNode contains an integer zip, a pointer caled *next* that points to the next ZipNode, and another pointer called *ptr_list* that points to another LinkedList storing multiple pointers, which are also pointing to the Info of each person who voted yes in this zip code.

Additional Command

1. User can enter “p” to print out the table by bucket. It will print out each person’s info in each bucket.

Challenges Faced

1. One of the challenges was that I kept using the same pointer for a node while iterating through the list. This resulted in a problem where only the same memory location keeps getting overwritten and new values are not updated properly. I realized I have to assign it to another new pointer first, and iterate on that new pointer.