ECE250-Project 2

Bingjian Du, b23du

Jan 12th, 2020

1. Overview of Classes

Class: user

Description: this is a basic unit of information.

Member variables: string number; string caller; user \*next\_caller; user \*prev\_caller;

Member functions:

user(string number, string name);

user();

~user();

string get\_number();

string get\_caller();

void set\_number(string number);

void set\_caller(string caller);

void set\_next(user \*n);

void set\_prev(user \*n);

user \*get\_next();

user \*get\_prev();

bool operator==(const user &a); compare user

bool operator!=(const user &a); compare user

Class: chain

Description: this is a linked\_list.

Member variables: user \*head, user \*tail

Member functions:

chain();

~chain();

void set\_head(user \*n);

void set\_tail(user \*n);

user \*get\_head();

user \*get\_tail();

Class: chain\_table

Description: this is an ordered hash table

Member variables: vector<chain> data, long long m

Member functions: (functions are consistent with the corresponding command)

chain\_table();

~chain\_table();

vector<chain> get\_data();

void define(string k);

void insert(string number, string name);

bool search(string number);

int delete\_data(string number);

void print(string i);

long long get\_size();

Class: open\_table

Description: this is an open-addressing hash table

Member variables: vector<user> data; long long m; long long exists;

Member functions: ( functions are consistent with the corresponding command)

open\_table()=default;

~open\_table();

vector<user> get\_data();

void define(string k);

int insert(string number, string name);

int search(string number);

int delete\_data(string number);

Class diagrams

|  |  |
| --- | --- |
| user | chain |
| string number;  string caller;  user \*next\_caller;  user \*prev\_caller; | user \*head,\*tail; |
| user(string number, string name);  user();  ~user();  string get\_number();  string get\_caller();  void set\_number(string number);  void set\_caller(string caller);  void set\_next(user \*n);  void set\_prev(user \*n);  user \*get\_next();  user \*get\_prev();  bool operator==(const user &a);  bool operator!=(const user &a); | chain();  ~chain();  void set\_head(user \*n);  void set\_tail(user \*n);  user \*get\_head();  user \*get\_tail(); |
| Chain\_table | Open\_table |
| vector<chain> data;  long long m; | vector<user> data;  long long m;  long long exists; |
| chain\_table();  ~chain\_table();  vector<chain> get\_data();  void define(string k);  void insert(string number,string name);  bool search(string number);  int delete\_data(string number);  void print(string i);  long long get\_size(); | open\_table()=default;  ~open\_table();  vector<user> get\_data();  void define(string k);  int insert(string number,string name);  int search(string number);  int delete\_data(string number); |

1. Constructors/Destructor

Class user: the first constructor is for creating an empty space for future use. The second constructor create a new user with a name and number. The = != operators are overwritten in this class for comparison

Class chain: the destructor delete users one by one to ensure there is no memory leak.

Other constructors and destructors are kept as default.

1. Test Cases

There are 2 cases I tested in addition to the example tests.

Test1: insert the number from large to small

Test2: test “insert-search-delete-search” works as expected

1. Performance

It is uniform hashing which means in each slot the number of elements is O(1), so the expected average runtime of insert, search and delete should be constant O(1).