Data structure

Lab task

Muhammad shaheer

20p-0480

Question

Implement the hashing technique Separate Chaining in c++.

The following functions for the algorithm should be implemented.

● Insert

● Delete

● Search

Note: Don’t use builtin list library to implement separate chaining

Code:

#include<iostream>

using namespace std;

 // maximum size of hash table

const int MAX\_SIZE = 10;

class node

{

public:

    int key;

    node\* next;

    node(int key)

    {

        this->key = key;

        next = NULL;

    }

};

class LinkedList

{

public:

    node\* head;

    LinkedList()

    {

        head = NULL;

    }

    // Insertion

    void insert(int key)

    {

        node\* newNode = new node(key);

        if (head == NULL)

        {

            head = newNode;

        }

        else

        {

            newNode->next = head;

            head = newNode;

        }

    }

    // Delettion

    void remove(int key)

    {

        if (head == NULL)

        {

            cout << "List is empty!" << endl;

            return;

        }

        if (head->key == key)

        {

            node\* temp = head;

            head = head->next;

            delete temp;

            return;

        }

        node\* prev = head;

        node\* curr = head->next;

        while (curr != NULL && curr->key != key)

        {

            prev = curr;

            curr = curr->next;

        }

        if (curr != NULL)

        {

            prev->next = curr->next;

            delete curr;

        }

        else

        {

            cout << "Key not found in the list!" << endl;

        }

    }

    // Search  in the linked list

    bool search(int key)

    {

        node\* curr = head;

        while (curr != NULL)

        {

            if (curr->key == key) {

                cout << "Key found in the list!" << endl;

                return true;

            }

            curr = curr->next;

        }

        cout << "Key not found in the list!" << endl;

        return false;

    }

    // Print the linked list

    void printList()

    {

        node\* curr = head;

        while (curr != NULL)

        {

            cout << curr->key << " ";

            curr = curr->next;

        }

        cout << endl;

    }

};

class Hash

{

public:

    LinkedList table[MAX\_SIZE];

    // Hash function

    int hash(int key)

    {

        return key % MAX\_SIZE;

    }

    // Insert   in the hash table

    void insert(int key)

    {

        int index = hash(key);

        table[index].insert(key);

    }

    // Delete from hash table

    void remove(int key)

    {

        int index = hash(key);

        table[index].remove(key);

    }

    // Search in the hash table

    bool search(int key)

    {

        int index = hash(key);

        return table[index].search(key);

    }

    // Print the hash table

    void printTable()

    {

        for (int i = 0; i < MAX\_SIZE; i++)

        {

            cout << "table[" << i << "]: ";

            table[i].printList();

        }

    }

};

int main()

{

    Hash h;

    // Insert keys

    h.insert(1);

    h.insert(2);

    h.insert(3);

    h.insert(4);

    h.insert(5);

    h.insert(6);

    h.insert(7);

    h.insert(8);

    h.insert(9);

    h.insert(10);

    // Print the hash table

    cout << "Hash Table:" << endl;

    h.printTable();

    cout << endl;

    // Delete keys

    h.remove(8);

    h.printTable();

    cout << endl;

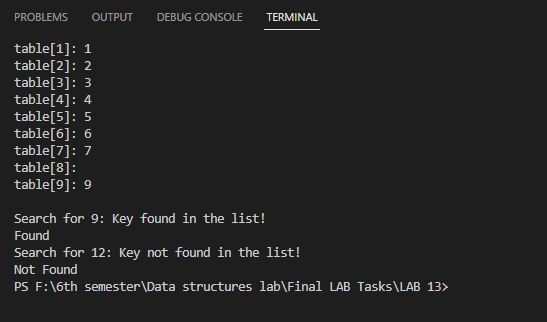
    cout << "Search for 9: " << (h.search(9) ? "Found" : "Not Found") << endl;

    cout << "Search for 12: " << (h.search(12) ? "Found" : "Not Found") << endl;

    return 0;

}

Output:



The end--------------------------------------------------------------------------------------------------------------------------------