**PROGRAM-10**

**AIM –** **Write an algorithm and program to implement hashing with chaining Using linked list for chaining and use division method for creating hashing function.**

**ALGORITHM-**

Hash Method

int hashCode(int key){

return key % SIZE;

}

## Search Operation

struct DataItem \*search(int key) {

//get the hash

int hashIndex = hashCode(key);

//move in array until an empty

while(hashArray[hashIndex] != NULL) {

if(hashArray[hashIndex]->key == key)

return hashArray[hashIndex];

//go to next cell

++hashIndex;

//wrap around the table

hashIndex %= SIZE;

}

return NULL;

}

## Insert Operation

void insert(int key,int data) {

struct DataItem \*item = (struct DataItem\*) malloc(sizeof(struct DataItem));

item->data = data;

item->key = key;

//get the hash

int hashIndex = hashCode(key);

//move in array until an empty or deleted cell

while(hashArray[hashIndex] != NULL && hashArray[hashIndex]->key != -1) {

//go to next cell

++hashIndex;

//wrap around the table

hashIndex %= SIZE;

}

hashArray[hashIndex] = item;

}

## Delete Operation

struct DataItem\* delete(struct DataItem\* item) {

int key = item->key;

//get the hash

int hashIndex = hashCode(key);

//move in array until an empty

while(hashArray[hashIndex] !=NULL) {

if(hashArray[hashIndex]->key == key) {

struct DataItem\* temp = hashArray[hashIndex];

//assign a dummy item at deleted position

hashArray[hashIndex] = dummyItem;

return temp;

}

//go to next cell

++hashIndex;

//wrap around the table

hashIndex %= SIZE;

}

return NULL;

}

**SOURCE CODE-**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include<conio.h>

struct hash \*hashTable = NULL;

int eleCount = 0;

struct node {

int key;

struct node \*next;

};

struct hash {

struct node \*head;

int count;

};

struct node \* createNode(int key)

{

struct node \*newnode;

newnode = (struct node \*)malloc(sizeof(struct node));

newnode->key = key;

newnode->next = NULL;

return newnode;

}

void insertToHash(int key)

{

int hashIndex = key % eleCount;

struct node \*newnode = createNode(key);

if (!hashTable[hashIndex].head)

{

hashTable[hashIndex].head = newnode;

hashTable[hashIndex].count = 1;

return;

}

newnode->next = (hashTable[hashIndex].head);

hashTable[hashIndex].head = newnode;

hashTable[hashIndex].count++;

return;

}

void deleteFromHash(int key)

{

int hashIndex = key % eleCount, flag = 0;

struct node \*temp, \*myNode;

myNode = hashTable[hashIndex].head;

if (!myNode)

{

printf("Given data is not present in hash Table!!\n");

return;

}

temp = myNode;

while (myNode != NULL)

{

if (myNode->key == key)

{

flag = 1;

if (myNode == hashTable[hashIndex].head)

hashTable[hashIndex].head = myNode->next;

else

temp->next = myNode->next;

hashTable[hashIndex].count--;

free(myNode);

break;

}

temp = myNode;

myNode = myNode->next;

}

if (flag)

printf("Data deleted successfully from Hash Table\n");

else

printf("Given data is not present in hash Table!!!!\n");

return;

}

void searchInHash(int key)

{

int hashIndex = key % eleCount, flag = 0;

struct node \*myNode;

myNode = hashTable[hashIndex].head;

if (!myNode)

{

printf("Search element unavailable in hash table\n");

return;

}

while (myNode != NULL)

{

if (myNode->key == key)

{

printf("KEY FOUND : %d\n", myNode->key);

flag = 1;

break;

}

myNode = myNode->next;

}

if (!flag)

printf("Search element unavailable in hash table\n");

return;

}

void display()

{

struct node \*myNode;

int i;

for (i = 0; i < eleCount; i++)

{

if (hashTable[i].count == 0)

continue;

myNode = hashTable[i].head;

if (!myNode)

continue;

printf("\nData at index %d in Hash Table:\n", i);

while (myNode != NULL)

{

printf("%-12d", myNode->key);

myNode = myNode->next;

}

}

return;

}

int main()

{

int n,key,ch;

printf("Enter the number of elements:");

scanf("%d", &n);

eleCount = n;

hashTable = (struct hash \*)calloc(n, sizeof (struct hash));

while (1)

{

printf("\n1. Insertion\t2. Deletion\n");

printf("3. Searching\t4. Display\n5. Exit\n");

printf("Enter your choice:");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter the key value:");

scanf("%d", &key);

getchar();

insertToHash(key);

break;

case 2:

printf("Enter the key to perform deletion:");

scanf("%d", &key);

deleteFromHash(key);

break;

case 3:

printf("Enter the key to search:");

scanf("%d", &key);

searchInHash(key);

break;

case 4:

display();

break;

case 5:

exit(0);

default:

printf("wrong option!!\n");

break;

}

}

return 0;

}

**OUTPUT-**



