DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

EAST WEST UNIVERSITY

(SPRING-2021)

Group Members

Mridul Ranjan Karmakar

ID: 2018-3-60-021

Md. Asif Imtiyaj Chowdhury

ID: 2019-3-60-115

PROJECT REPORT

Course: CSE 207
Section: 1

Instructor: Dr. Maheen Islam

Associate Professor

Introduction

A **hash table** is a data structure that efficiently stores and retrieves data from memory. There are many ways to construct a hash table. One is Open Hashing (Close Addressing) and another is Closed Hashing (Open Addressing). We will be using Open Hashing (chaining method) using the Divisor Method. Because Open Hashing contains chaining and we are implementing chained hash table.

An object is stored in a hash table by associating it with a key. Input a data to hash table, we use divisor method for finding hash key. In most cases the hashing process computes a number. It is done for faster inputing data and access to element. The efficiency of mapping depends on the efficiency of the hash method used. In a hash table, data is stored in link List by help of array indexing, where each data value has its own unique node address but multiple data value has same array index because of chaining hash. Access of data becomes very fast if we know the index of the desired data.

Let a hash function H(x) maps the value *x* at the key index x%10 in an array. For example if the list of values is [11, 12, 13, 14, 15] it will stored at position {1, 2, 3, 4, 5} in the hash table.

Solution &problem description

Description of problem:

To implement a hash table it has to be input data in an certain position and for inputting multiple data there were many collision because of the same array index (hash key) so there we used chaining to solve this process.

Data Structure Used: Array & Link List

Features:

1. Insert data in the hash table
2. Search data if it exist or not
3. How Many Data have entry the hash table from the starting
4. Delete any data from the hash table efficiently without any memory waste.
5. Display the whole hash table with the desing.
6. User input system

The algorithm and all process are shown below:

The funtions() we used in the chained Hash table:

1. Intial()
2. Insert()
3. Delete()
4. Search()
5. Print()

Step 01:

#include<stdio.h>

#include<stdlib.h>

#define size 7

These are the Library Function and define the array size of 7.

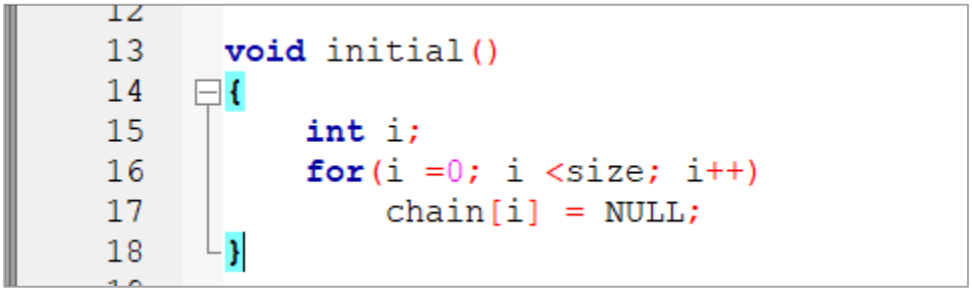
“

```
struct node
{
   int data;
   struct node *next;
};
```

”

Here,we declear the struct class for create node and store the node data and node address.

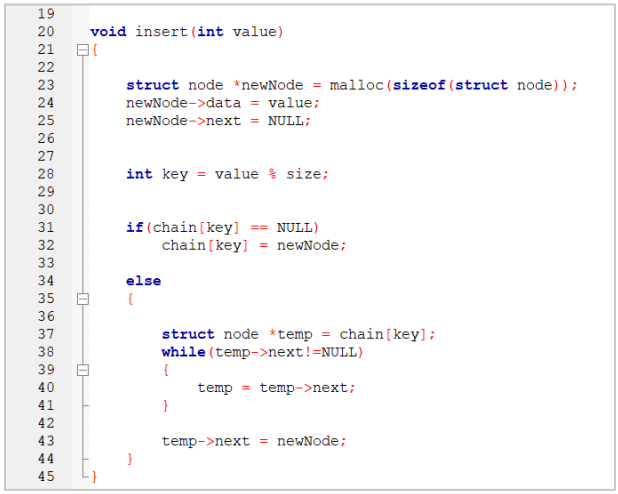
Step 2:



```
void initial()
{
    int i;
    for(i =0; i <size; i++)
        chain[i] = NULL;
}
```

By initial function we assing all the array index by “NULL” .

Step 3:



```
void insert(int value)
{

    struct node *newNode = malloc(sizeof(struct node));
    newNode->data = value;
    newNode->next = NULL;


    int key = value % size;


    if(chain[key] == NULL)
        chain[key] = newNode;

    else
    {

        struct node *temp = chain[key];
        while(temp->next!=NULL)
        {
            temp = temp->next;
        }

        temp->next = newNode;
    }
}
```

Here, in insert ()

-> pass the integer value as a parameter(int vlaue)

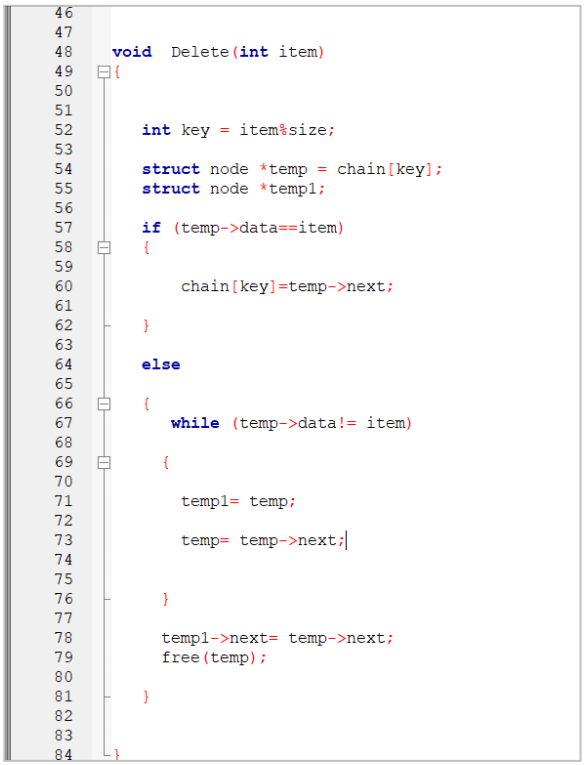
-> Create a node by malloc()

-> Find the Hash key ,MODs the value by array size (value%size)

-> if condition check if array index is NULL then input data in newnode at that index address.

-> else{} check that if key index is not NULL then find the last node and store the new node to the last position of the chain.

Step 4:



```
46
47
48   void  Delete(int item)
49   {
50
51
52      int key = item%size;
53
54      struct node *temp = chain[key];
55      struct node *temp1;
56
57      if (temp->data==item)
58      {
59
60          chain[key]=temp->next;
61
62      }
63
64      else
65
66      {
67         while (temp->data!= item)
68
69        {
70
71          temp1= temp;
72
73          temp= temp->next;
74
75
76        }
77
78        temp1->next= temp->next;
79        free(temp);
80
81      }
82
83
84   }
```

-> In delete() we pass the value as

Parameter (int item).

-> then find the key where the item index will be.

-> if ()condtion check if first data is match with item so replace with the next node data.

-> In else () the while(Loop) go to the item until its found then it cut the link and linkup with the previous node and next node of the deletion node.

-> free(the deletion node)

Step 5:

```
85
86  void search(int item)
87  {
88
89      int key = item%size;
90
91      struct node *temp = chain[key];
92
93
94      if (chain[key]==NULL)
95      {
96          printf("Value Is Not Match Found ");
97
98      }
99
100
101     else
102     {
103
104
105       while (temp->next!= NULL)
106
107      {
108         if (temp->data==item)
109
110         {
111
112             printf("Your Value Is Match Found In The Hash Table");
113
114             break;
115
```

```
119
120         temp=temp->next;
121
122     }
123
124
125       if (temp->data== item && temp->next==NULL)
126
127       {
128            printf(" Your Value Is  Match Found in The Hash Table")
129
130
131       }
132
133
134        if (temp->data!= item)
135
136       {
137            printf("Value Is Not Match Found ");
138
139
140       }
141
142
143   }
144
145
146 }
147
```

Here in search function()

->Search (Passing a item by parameter)

-> *temp pointer point the index of the key item

-> if condition() check if index value NULL then Item not found.

-> in else() the while(traverse the whole linklist)

-> if (Found item) then print match Found and break the process.

-> next if (compare with the last node of the link)

-> Last if(check if any of the node does not have the search item).

Step 6:

```
147
148
149   void print()
150   {
151       int i;
152
153       for(i=0;i<size;i++)
154       {
155           struct node *temp = chain[i];
156           printf("chain[%d]-->",i);
157           while(temp!=NULL)
158           {
159               printf("%d -->",temp->data);
160               temp = temp->next;
161           }
162           printf("NULL\n");
163       }
164   }
165
```

Here in Print ()

->for(Traverse the whole array) print the array

->while(traverse the each linklist and print the data)

-> print the all last node next position by NULL

Step 7:

```
165
166  int main()
167  {
168
169
170      initial();
171
172      while (1)
173      {
174          printf("\n     MENU    \n");
175          printf("\n1:Create Hash Table");
176          printf("\n2:Display The Hash Table");
177          printf("\n3:How Many Data Entry In The Hash Table");
178          printf("\n4:Delete a Data From The Hash Table");
179          printf("\n5:Search an Data From The Hash Table");
180
181          int choice,m,x,y;
182
183          printf("\n\nEnter Your Choice:");
184          scanf("%d",&choice);
```

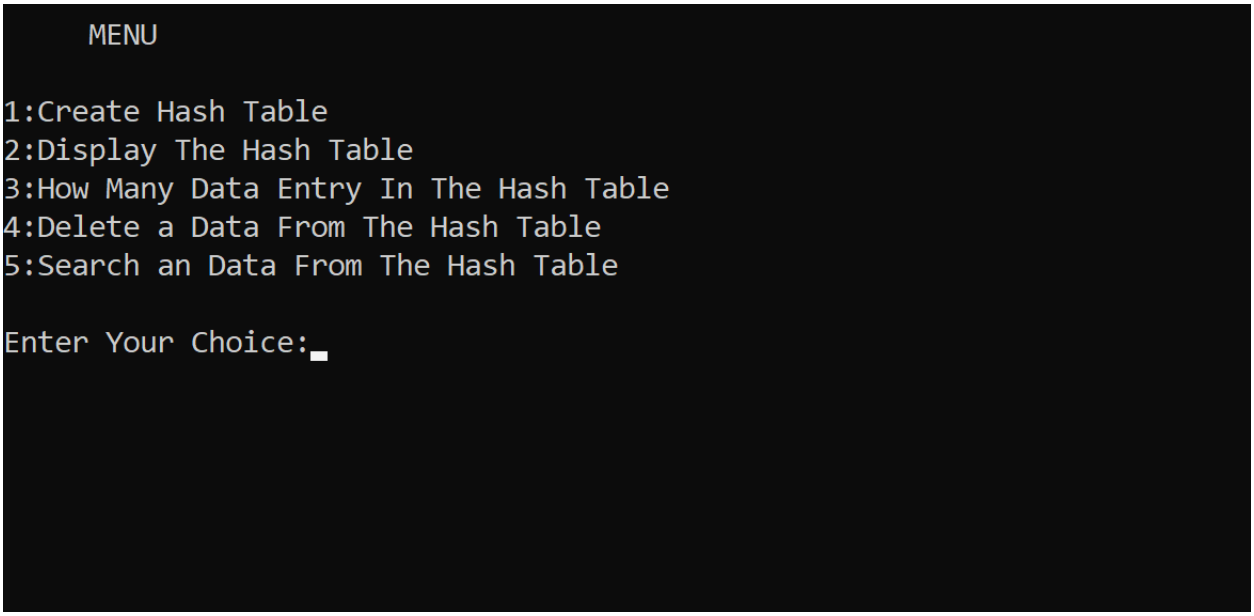
Main Menu declaring

Step 8:

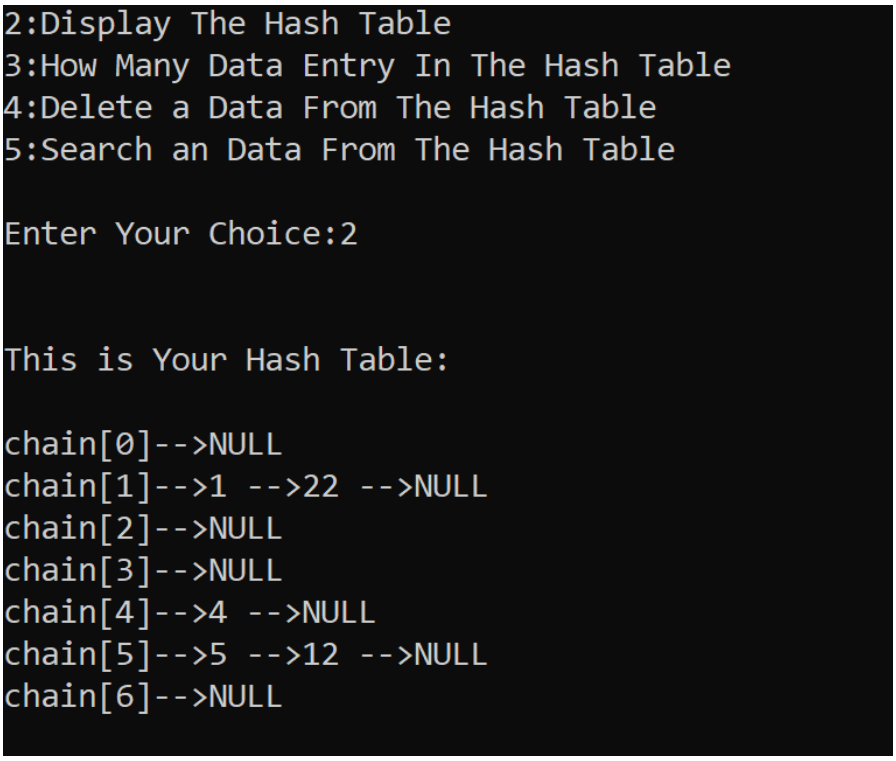
```
190
191      case 1:
192
193
194          printf("\nEnter Your Hash table Value:\n");
195
196          scanf("%d",&m);
197          insert(m);
198          count++;
199
200      break;
201
202
203      case 2:
204
205      printf("\n\nThis is Your Hash Table:\n\n" );
206      print();
207
208      break;
209
210      case 3:
211
212      printf("\n There are %d Value in Your Hash table\n",count);
213
214      break;
215
216      case 4:
217
218         printf("\nWhat Value You Want to Delete From Hash Table: ");
219         scanf("%d",&x);
220         Delete(x);
221
222         break;
223
224      case 5:
225
226         printf("\nEnter The Search Value In the Hash Table: ");
227         scanf("%d",&y);
228         search(y);
229
230
231      }
232
233
234  }
```

User Input process By (switch case)

Step 9:



Run The programme and the Display Manual



The Hash Table after performing some operation

Conclusion:

The hash table is a very uniqe design and process to store and retrieving data and very efficient. In this project we try to implement the chained hash table to perform some data operation. Though all the work of human being is not fully perfect hence the project will give the proper idea and implementation of chained Hash Table and efficiently run the programme and perform the features operation perfectly.