**Subject Name: Advance Data Structure**

**Group: D**

**Assignment No.: 7**

**BATCH B1**

**Year: 2017-18**

**College: VIIT**

**Date of Completion: 30-3-2018**

**Aim:**

Implement all the functions of a dictionary (ADT) using hashing.

Data: Set of (key, value) pairs, Keys are mapped to values, Keys must be comparable,

Keys must be unique

Standard Operations: Insert (key, value), Find (key), Delete (key)

**Objective:** To implement Dictionary (ADT) and perform operations like insert, search, delete on it. Software and Hardware Requirement:

1. 64-bit Open source Linux or its derivative.
2. Open Source C++ Programming tool like G++/GCC.

**Theory:**

**DICTIONARY ADT:**

Dictionary (map, association list) is a data structure, which is generally an association of unique keys with some values. One may bind a value to a key, delete a key (and naturally an associated value) and lookup for a value by the key. Values are not required to be unique. Simple usage example is an explanatory dictionary. In the example, words are keys and explanations are values.

**Example:** consider a data structure that stores bank accounts; it can be viewed as a dictionary, where account numbers serve as keys for identification of account objects.

1. The dictionary ADT models a searchable collection of key-element entries
2. The main operations of a dictionary are searching, inserting, and deleting items
3. Multiple items with the same key are allowed
4. Applications:
   1. word-definition pairs
   2. credit card authorizations
   3. DNS mapping of host names (e.g., datastructures.net) to internet IP addresses (e.g., 128.148.34.101)
5. Dictionary ADT methods:
   1. find(k): if the dictionary has an entry with key k, returns it, else, returns null
   2. find All(k): returns an iterator of all entries with key k
   3. insert(k, o): inserts and returns the entry (k, o)
   4. remove(e): remove the entry e from the dictionary
   5. entries(): returns an iterator of the entries in the dictionary (f) size(), is Empty()
6. How to Implement a Dictionary?
   1. Sequences / Arrays
      1. ordered
      2. unordered
   2. Binary Search Trees
   3. Skip lists
   4. Hash tables
7. Implementing a Dictionary with a Sequence
   1. unordered sequence
      1. searching and removing takes O(n) time
      2. inserting takes O(1) time
      3. applications to log files (frequent insertions, rare searches and removals) 34 14 12 22 18
   2. array-based ordered sequence (assumes keys can be ordered)
      1. - searching takes O(log n) time (binary search)
      2. - inserting and removing takes O(n) time
      3. - application to look-up tables (frequent searches, rare insertions and removals)

**Algorithm:**

Dictionary Hashing

1. Enter a choice ’ch’ to choose from the choice menu.
2. If ch=1, insert ’key’ and ’meaning’ in the dictionary. To insert, first find the appropriate position in the hash table by passing ’key’ as argument to the hash function. If a collision occurs then use separate chaining to resolve.
3. If ch=2, display all the words in the dictionary.
4. If ch=3, Search a specific word by the ’key’

**Code:**  
/\*

* Hashing.cpp
* Created on: Mar 16, 2017
* Author: student

\*/

#include <iostream> #include<string>

using namespace std;

typedef struct node

{

int key,value;

struct node \*next;

} node;

class Hash

{

private:

node \*array[10];

node \*New,\*temp;

public: Hash();

node \*CreateNode();

void insert();

int hashfun(int key);

void CreateLL(int keyvalue,node \*Newnode);

void display(); void search(); void Delete(); } ;

Hash::Hash()

{

Int i;

for(i=0;i<10;i++)

{

array[i]=NULL;

}

node \*Hash::CreateNode()

{

New=new node;

cout<<"\nEnter Key : "; cin>>New->key;

cout<<"\nEnter Value : "; cin>>New->value;

New->next=NULL; return (New); }

int Hash::hashfun(int key)

{ return( key%10); }

void Hash::CreateLL(int keyvalue,node \*Newnode)

{

if(array[keyvalue]==NULL) array[keyvalue]=Newnode;

}

else { temp=array[keyvalue]; while(temp->next!=NULL) temp=temp->next;

temp->next=Newnode; }

}

void Hash::display()

{

for(int i=0;i<10;i++)

{

temp=array[i]; cout<<i<<" ==> ";; while(temp!=NULL)

{

cout<<"( "<<temp->key<<" , "<<temp->value<<" ) ,"; temp=temp->next;

}

cout<<endl;

}

}

void Hash::insert()

{

int keyvalue,flag=0;

node \*Newnode;

char ch;

do {

Newnode=CreateNode(); keyvalue=hashfun(Newnode->key);

cout<<"\n keyvalue : "<<keyvalue; temp=array[keyvalue];

while(temp!=NULL) {

if(temp->key==Newnode->key) { cout<<"\n\tKey Must Be Unique....!!!"; flag=1; break;

}

temp=temp->next;

}

if(flag==0)

CreateLL(keyvalue,Newnode);

cout<<"\n\n\tDo u want to add(y/Y) : ";

cin>>ch;

}while(ch==’Y’ || ch==’y’);

}

void Hash::search()

{

int flag=0,Newkey;

cout<<"Enter Key to Search : "; cin>>Newkey; int index=hashfun(Newkey);

temp=array[index];

while(temp!=NULL) {

if(temp->key==Newkey) {

cout<<"( "<<temp->key<<" , "<<temp->value<<" ) ,";

//cout<<"\nAvailable ";

flag=1;

}

temp=temp->next; } if(flag==0) cout<<"\Key Not Found.....!!!!";

}

void Hash::Delete() {

int delKey,flag=0; node \*temp1,\*temp2; cout<<"\nEnter Key To Delete : "; cin>>delKey;

int index=hashfun(delKey);

temp1=array[index]; temp2=temp1->next; if(temp1>key==delKey){ array[index]=temp1->next; delete(temp1); flag=1; cout<<"\n\t Deleted...!!!";

}

else{

while(temp1!=NULL && temp2!=NULL)

{

if(temp2->key==delKey) {

temp1->next=temp2->next;

delete(temp2);

flag=1;

cout<<"\n\t Deleted...!!!";

break;

}

temp1=temp1->next; temp2=temp2->next;

}

}

if(flag==0)

cout<<"\n\tKey Not Available....!!!";

}

int main() {

Hash obj;

int ch;

do {

cout<<"\n\n\n\t\*\*\*\*\* MENU \*\*\*\*\*";

cout<<"\n1.Insert (Key,Value)\n2.Search Key\n3.Display All (Key,Value)\n4.Delete Key\n0.

cout<<"\t Enter your choice : "; cin>>ch;

switch (ch) { case 0: return 0 ; case 1:

obj.insert(); break;

case 2:

obj.search(); break;

case 3:

obj.display(); break;

case 4:

obj.Delete(); break;

default:

cout<<"\nInvalid choice.....!!!";

}

} while(1); return 0 ;

}

**Output :**

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 1

Enter Key : 10

Enter Value : 100 keyvalue : 0

Do u want to add(y/Y) : y

Enter Key : 50 Enter Value : 200 keyvalue : 0

Do u want to add(y/Y) : y

Enter Key : 30 Enter Value : 300 keyvalue : 0

Do u want to add(y/Y) : y

Enter Key : 40 Enter Value : 400 keyvalue : 0

Do u want to add(y/Y) : y

Enter Key : 50 Enter Value : 500

keyvalue : 0 Key Must Be Unique....!!!

Do u want to add(y/Y) : y

Enter Key : 50

Enter Value : 5556 keyvalue : 0 Key Must Be Unique....!!! Do u want to add(y/Y) : y

Enter Key : 789 Enter Value : 5456 keyvalue : 9

Do u want to add(y/Y) : n

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 3

1. ==> ( 10 , 100 ),( 50 , 200 ),( 30 , 300 ),( 40 , 400 ) ,
2. ==>
3. ==>
4. ==>
5. ==>
6. ==>
7. ==>
8. ==>
9. ==>
10. ==>

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 1

Enter Key : 45

Enter Value : 120

keyvalue : 5

Do u want to add(y/Y) : 1

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 78

Invalid choice.....!!!

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 1

Enter Key : 78 Enter Value : 1000 keyvalue : 8

Do u want to add(y/Y) : n

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 3

1. ==> ( 10 , 100 ),( 50 , 200 ),( 30 , 300 ),( 40 , 400 ) ,
2. ==>
3. ==>
4. ==>
5. ==>
6. ==> ( 45 , 120 ) ,
7. ==>
8. ==>
9. ==> ( 78 , 1000 ) ,
10. ==>

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 2

Enter Key to Search : 78 ( 78 , 1000 ) ,

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 4 Enter Key To Delete : 45

Deleted...!!!

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice : 3

1. ==> ( 10 , 100 ),( 50 , 200 ),( 30 , 300 ),( 40 , 400 ) ,
2. ==>
3. ==>
4. ==>
5. ==>
6. ==>
7. ==>
8. ==>
9. ==> ( 78 , 1000 ) ,
10. ==>

\*\*\*\*\* MENU \*\*\*\*\*

1.Insert ( Key,Value )

2.Search Key

3.Display All ( Key,Value )

4.Delete Key

0 .Exit

Enter your choice :

**Conclusion:**

We understood implementing of dictionary using hash tables and performing operations like insert, search and delete.