MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY

«KHARKIV POLYTECHNIC INSTITUTE»

Department of Software Engineering and Management Information Technologies

Report from lab № 2

discipline «Algorithm and Data Structures»

Kharkiv

2018

Theme :Basic Data Structures:

Objective: explore hash functions and hash tables and get programming skills

of algorithms that process them..

Task:

Develop a program that reads integers N, M (1 <N, M <256) from keyboard, N

<key, value> pairs (key is an integer, floating point number or a string, depending on

the variant, value is a string, all strings up to 255 characters), none of which is

repeated, and M keys. All strings separated by a space or a new line. The program

stores pairs of strings into a hash table, and displays the values corresponding to

listed keys.

Example of key-lines input.

3 2

abc x

gh yq

io qw

gh

io

Output.

yq

qw

10

Usage of ready data structures (e.g., STL) is prohibited, but string

implementations can be used (for example, std::string in C++).

Progress of the lab:

Theory : Many applications require a dynamic set that supports only the dictionary

opera- tions INSERT, SEARCH, and DELETE. A hash table is an effective data

structure for implementing dictionaries. Although searching for an element in a hash

table can take as long as searching for an element in a linked list— *O*(*n*) time in

the worst case—in practice, hashing performs extremely well. Under reasonable

assumptions, the average time to search for an element in a hash table is *O*(1) .

Construction of the hash function by multiplication (multiplication method) is

as follows. Let *m* be the number of hash values. Fix a constant *A* in interval

9

(0,1) and let *h*(*k* )=[*m*(*kA mod* 1)] , where *kAmod* 1 is fractional part of *kA* .

Advantage of multiplication method is that the quality of the hash function

only slightly dependents on the choice of *m* . Usually *m* is chosen as a power of

two, since for most computers multiplication is implemented as shift.

CODE:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace trivial\_hashing

{

class hashTable

{

public string name;

public int age;

public int key;

}

class Program

{

private double k;

public Program(double k)

{

this.k = k;

}

public int HashCode(int key, int tableSize)

{

return (int)(tableSize \* (k \* key % 1));

}

public void delete(hashTable[] HT, int size )

{

Console.WriteLine("\n enter the key of elment to delete");

int key = Convert.ToInt32(Console.ReadLine());

if (HT[HashCode(key, size)] == null)

{

Console.WriteLine("record does not exist ");

return;

}

HT[HashCode(key, size)] = null;

Console.WriteLine("Record has been deleted");

}

public void insert( hashTable[] HT, int size)

{

hashTable element = new hashTable();

Console.WriteLine("\n enter name of student ");

element.name =Console.ReadLine();

Console.WriteLine("\n enter age of student");

element.age = Convert.ToInt32(Console.ReadLine()); ;

Console.WriteLine("\n enter key ");

element.key = Convert.ToInt32(Console.ReadLine());

HT[HashCode(element.key, size)] = element;

}

public void search(hashTable[] HT, int size)

{

Console.WriteLine("\n Enter the key of element to search ");

int key = Convert.ToInt32(Console.ReadLine());

if (HT[HashCode(key, size)] == null)

{

Console.WriteLine("record does not exist ");

return;

}

Console.WriteLine("\n element searched for :");

Console.WriteLine("\n Name : " + HT[HashCode(key, size)].name);

Console.WriteLine("\n Age : " + HT[HashCode(key, size)].age);

}

static void Main(string[] args)

{

hashTable[] HT= new hashTable[256];

int table = 256;

Program p = new Program((Math.Sqrt(5) - 1) / 2);

while (true)

{

try

{

Console.WriteLine("\nMenu");

Console.WriteLine("1. Add a record to the Table");

Console.WriteLine("2. Delete a record from the Table");

Console.WriteLine("3. Search table ");

Console.WriteLine("4. Exit ");

string ch = Console.ReadLine();

switch (ch)

{

case "1":

{

p.insert(HT, table);

}

break;

case "2":

{

p.delete(HT, table);

}

break;

case "3":

{

p.search(HT, table);

}

break;

case "4":

return;

default:

{

Console.WriteLine("\nInvalid option");

}

break;

}

}

catch (Exception e)

{

Console.WriteLine("Check for the values entered.");

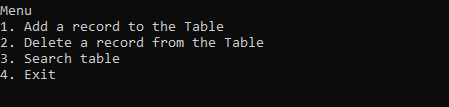
}

}

}

}

}



:

Figure.1- Menu of program

\

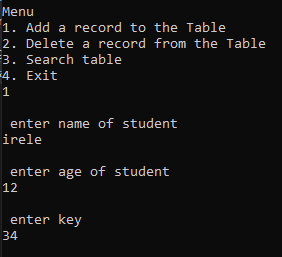


Figure.2- Add a record

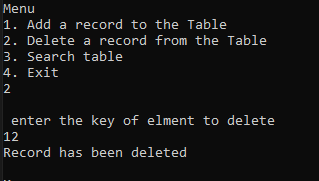


Figure.3- Delete a record

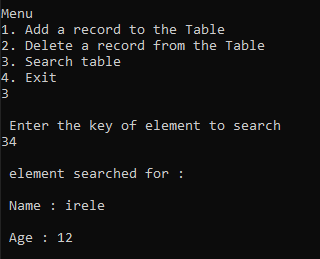


Figure.4- Minimum and maximum element

Conclusion:

In this laboratory the study of Hash tables was considered

* Develop a hash function.
* Searching of hash tables.
* Deleting from Hash table.