| Program No | 8 |
| --- | --- |
| Roll No | 1308 |
| Unit | 2 |
| Title | Hashing |
| Program | Modulo Division with Linear Probe |

**Source Code:**

/\*

Name: Manish Chilwal

Roll No.: 1308

Unit 2 : Hashing

Program: Modulo Division with Linear Probe

\*/

#include<iostream>

#include<conio.h>

#define MAX 10

#define EMPTY -1

using namespace std;

/\* 1. Node Template \*/

/\* 2. Template of Hash Table \*/

class Hash

{

int h[MAX];

public:

Hash()

{

int i;

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

{

h[i] = EMPTY;

}

}

void Insert(int key);

void Display();

void Search(int skey);

};

/\* 3. Funcion \*/

void Hash::Insert(int key)

{

int addr = key % MAX; //Modulo Division calculation-HASH FUNCTION

int oaddr = addr;

if(h[addr] == EMPTY) // addr is available

{

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

else // Collision!

{

do

{

addr=(addr+1)%MAX;

if(addr == oaddr)

{

cout << "Hash Table Full";

return;

}

}

while(h[addr] != EMPTY);

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

}

void Hash::Display()

{

int i;

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

{

cout << "h[" << i << "] = " << h[i] << endl;

}

}

void Hash::Search(int skey)

{

int saddr = skey % MAX; // HASH FUNCTION

int oaddr = saddr; // ORIGINAL ADDRESS

if(h[saddr] == skey)

{

cout << "Value " << skey << " is found at " << saddr;

}

else

{

do

{

saddr = (saddr + 1)%MAX;

if(saddr == oaddr)

{

break;

}

}

while(h[saddr] != skey);

if(h[saddr] == skey)

{

cout << "Value " << skey << " is found at " << saddr;

}

else

{

cout << "Value " << skey << " is not found";

}

}

}

/\* 4. Menu \*/

int main()

{

int ch, num;

Hash h;

while(1)

{

system("cls");

cout << "\*\*\* Hashing using Modulo Division & Linear Probe \*\*\*" << endl << endl;

cout << "1. Insert a value in the Hash Table" << endl;

cout << "2. Display the Hash Table" << endl;

cout << "3. Search the Hash Table" << endl;

cout << "4. Exit " << endl;

cout << "Enter a choice: ";

cin >> ch;

switch(ch)

{

case 1:

cout << "Enter a value: ";

cin >> num;

h.Insert(num);

getch();

break;

case 2:

h.Display();

getch();

break;

case 3:

cout << "Enter a value: ";

cin >> num;

h.Search(num);

getch();

break;

case 4:

exit(1);

default:

cout << "Please Enter a valid choice ;)";

getch();

break;

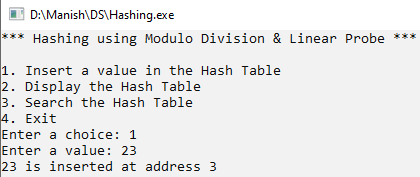
}//end of switch

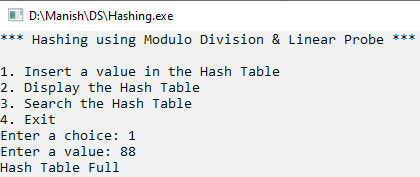
}//end of while

}//end of main

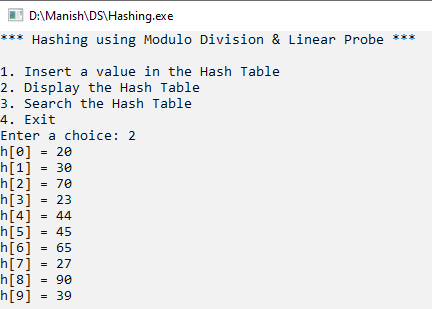
**Output:**

**Insert value:**

****

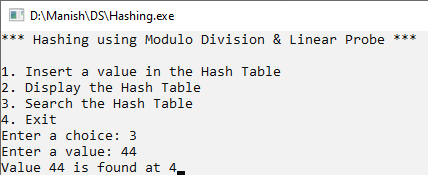
****

**Display HashTable:**

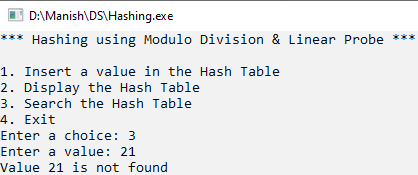
****

**Search:**

**Successful:**

****

**Unsuccessful:**

****

| Program No | 8 |
| --- | --- |
| Roll No | 1308 |
| Unit | 2 |
| Title | Hashing |
| Program | Digit Extraction with Linear Probe |

**Source Code:**

/\*

Name: Manish Chilwal

Roll No.: 1308

Unit 2 : Hashing

Program: Digit Extraction with Linear Probe

\*/

#include<iostream>

#include<conio.h>

#define MAX 10

#define EMPTY -1

using namespace std;

/\* 1. Node Template \*/

/\* 2. Template of Hash Table \*/

class Hash

{

int h[MAX];

public:

Hash()

{

int i;

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

{

h[i] = EMPTY;

}

}

void Insert(int key);

void Display();

void Search(int skey);

};

/\* 3. Funcion \*/

void Hash::Insert(int key)

{

int a[6];

int i,r,max=key,d=0;

while(max>0)

{

d++;

max /=10;

}

if(d > 6)

{

cout << "Enter a key less then 6 digit in size: ";

return;

}

int nkey = key;

for(i = 5;i>=0;i--)

{

r = nkey % 10;

nkey = nkey/10;

a[i] = r;

}

int addr = a[4]; //Digit Extraction calculation-HASH FUNCTION

int oaddr = addr;

if(h[addr] == EMPTY) // addr is available

{

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

else // Collision!

{

do

{

addr=(addr+1)%MAX;

if(addr == oaddr)

{

cout << "Hash Table Full";

return;

}

}

while(h[addr] != EMPTY);

h[addr] = key;

cout << key << " is inserted at address " << addr << endl;

}

}

void Hash::Display()

{

int i;

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

{

cout << "h[" << i << "] = " << h[i] << endl;

}

}

void Hash::Search(int skey)

{

int a[6];

int i,r;

int nkey = skey;

for(i = 5;i>=0;i--)

{

r = nkey % 10;

nkey = nkey/10;

a[i] = r;

}

int saddr = a[4]; // HASH FUNCTION

int oaddr = saddr; // ORIGINAL ADDRESS

if(h[saddr] == skey)

{

cout << "Value " << skey << " is found at " << saddr;

}

else

{

do

{

saddr = (saddr + 1)%MAX;

if(saddr == oaddr)

{

break;

}

}

while(h[saddr] != skey);

if(h[saddr] == skey)

{

cout << "Value " << skey << " is found at " << saddr;

}

else

{

cout << "Value " << skey << " is not found";

}

}

}

/\* 4. Menu \*/

int main()

{

int ch, num;

Hash h;

while(1)

{

system("cls");

cout << "\*\*\* Hashing using Digiit extraction & Linear Probe \*\*\*" << endl << endl;

cout << "1. Insert a value in the Hash Table" << endl;

cout << "2. Display the Hash Table" << endl;

cout << "3. Search the Hash Table" << endl;

cout << "4. Exit " << endl;

cout << "Enter a choice: ";

cin >> ch;

switch(ch)

{

case 1:

cout << "Enter a value: ";

cin >> num;

h.Insert(num);

getch();

break;

case 2:

h.Display();

getch();

break;

case 3:

cout << "Enter a value: ";

cin >> num;

h.Search(num);

getch();

break;

case 4:

exit(1);

default:

cout << "Please Enter a valid choice ;)";

getch();

break;

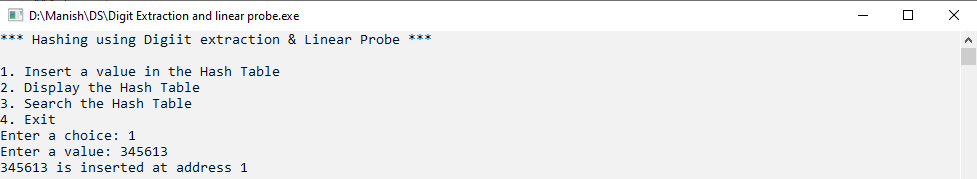
}//end of switch

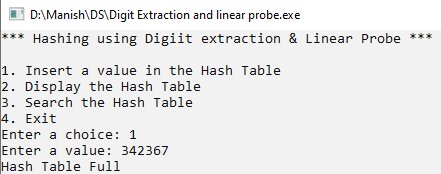
}//end of while

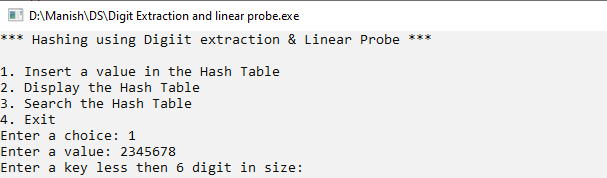
}//end of main

**Output:**

**Insert value:**

****



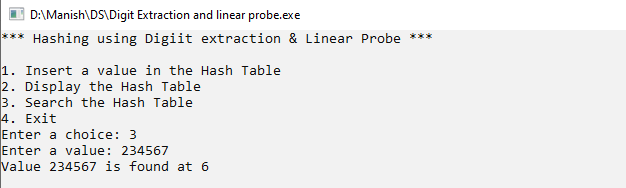
****

**Display HashTable:**



**Search:**

**Successful:**



**Unsuccessful:**

