Hashing Using Linear Probing

#include <stdio.h>

#include<stdlib.h>

#define TABLE\_SIZE 10

int h[TABLE\_SIZE]={NULL};

void insert()

{

int key,index,i,flag=0,hkey;

printf("\nenter a value to insert into hash table\n");

scanf("%d",&key);

hkey=key%TABLE\_SIZE;

for(i=0;i<TABLE\_SIZE;i++)

{

index=(hkey+i)%TABLE\_SIZE;

if(h[index] == NULL)

{

h[index]=key;

break;

}

}

if(i == TABLE\_SIZE)

printf("\nelement cannot be inserted\n");

}

void search()

{

int key,index,i,flag=0,hkey;

printf("\nenter search element\n");

scanf("%d",&key);

hkey=key%TABLE\_SIZE;

for(i=0;i<TABLE\_SIZE; i++)

{

index=(hkey+i)%TABLE\_SIZE;

if(h[index]==key)

{

printf("value is found at index %d",index);

break;

}

}

if(i == TABLE\_SIZE)

printf("\n value is not found\n");

}

void display()

{

int i;

printf("\nelements in the hash table are \n");

for(i=0;i< TABLE\_SIZE; i++)

printf("\nat index %d \t value = %d",i,h[i]);

}

main()

{

int opt,i;

while(1)

{

printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");

scanf("%d",&opt);

switch(opt)

{

case 1:

insert();

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:exit(0);

}

}

}

OUTPUT

Press 1. Insert 2. Display 3. Search 4.Exit

1

enter a value to insert into hash table

12

Press 1. Insert 2. Display 3. Search 4.Exit

1

enter a value to insert into hash table

2

Press 1. Insert 2. Display 3. Search 4.Exit

2

elements in the hash table are

at index 0 value = 0

at index 1 value = 0

at index 2 value = 12

at index 3 value = 2

at index 4 value = 0

at index 5 value = 0

at index 6 value = 0

at index 7 value = 0

at index 8 value = 0

at index 9 value = 0

Press 1. Insert 2. Display 3. Search 4.Exit