**Linear probing**

#include <iostream>

#include <cstdio>

#include <cstdlib>

using namespace std;

const int T\_S = 5;

class HashTable {

public:

int k;

int v;

HashTable(int k, int v) {

this->k = k;

this->v = v;

}

};

class DelNode:public HashTable {

private:

static DelNode \*en;

DelNode():HashTable(-1, -1) {}

public:

static DelNode \*getNode() {

if (en == NULL)

en = new DelNode();

return en;

}

};

DelNode \*DelNode::en = NULL;

class HashMapTable {

private:

HashTable \*\*ht;

public:

HashMapTable() {

ht = new HashTable\* [T\_S];

for (int i = 0; i < T\_S; i++) {

ht[i] = NULL;

}

}

int HashFunc(int k) {

return k % T\_S;

}

void Insert(int k, int v) {

int hash\_val = HashFunc(k);

int init = -1;

int delindex = -1;

while (hash\_val != init && (ht[hash\_val] == DelNode::getNode() || ht[hash\_val] != NULL && ht[hash\_val]->k != k)) {

if (init == -1)

init = hash\_val;

if (ht[hash\_val] == DelNode::getNode())

delindex = hash\_val;

hash\_val = HashFunc(hash\_val + 1);

}

if (ht[hash\_val] == NULL || hash\_val == init) {

if(delindex != -1)

ht[delindex] = new HashTable(k, v);

else

ht[hash\_val] = new HashTable(k, v);

}

if(init != hash\_val) {

if (ht[hash\_val] != DelNode::getNode()) {

if (ht[hash\_val] != NULL) {

if (ht[hash\_val]->k== k)

ht[hash\_val]->v = v;

}

} else

ht[hash\_val] = new HashTable(k, v);

}

}

int SearchKey(int k) {

int hash\_val = HashFunc(k);

int init = -1;

while (hash\_val != init && (ht[hash\_val] == DelNode::getNode() || ht[hash\_val] != NULL && ht[hash\_val]->k!= k)) {

if (init == -1)

init = hash\_val;

hash\_val = HashFunc(hash\_val + 1);

}

if (ht[hash\_val] == NULL || hash\_val == init)

return -1;

else

return ht[hash\_val]->v;

}

void Remove(int k) {

int hash\_val = HashFunc(k);

int init = -1;

while (hash\_val != init && (ht[hash\_val] == DelNode::getNode() || ht[hash\_val] != NULL && ht[hash\_val]->k!= k)) {

if (init == -1)

init = hash\_val;

hash\_val = HashFunc(hash\_val + 1);

}

if (hash\_val != init && ht[hash\_val] != NULL) {

delete ht[hash\_val];

ht[hash\_val] = DelNode::getNode();

}

}

~HashMapTable() {

delete[] ht;

}

};

int main() {

HashMapTable hash;

int k, v;

int c;

while(1) {

cout<<"1.Insert element into the table"<<endl;

cout<<"2.Search element from the key"<<endl;

cout<<"3.Delete element at a key"<<endl;

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

cout<<"Enter your choice: ";

cin>>c;

switch(c) {

case 1:

cout<<"Enter element to be inserted: ";

cin>>v;

cout<<"Enter key at which element to be inserted: ";

cin>>k;

hash.Insert(k, v);

break;

case 2:

cout<<"Enter key of the element to be searched: ";

cin>>k;

if(hash.SearchKey(k) == -1) {

cout<<"No element found at key "<<k<<endl;

continue;

} else {

cout<<"Element at key "<<k<<" : ";

cout<<hash.SearchKey(k)<<endl;

}

break;

case 3:

cout<<"Enter key of the element to be deleted: ";

cin>>k;

hash.Remove(k);

break;

case 4:

exit(1);

default:

cout<<"\nEnter correct option\n";

}

}

return 0;

}

**Output**:

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 1

Enter element to be inserted: 10

Enter key at which element to be inserted: 2

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 1

Enter element to be inserted: 7

Enter key at which element to be inserted: 6

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 1

Enter element to be inserted: 4

Enter key at which element to be inserted: 5

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 1

Enter element to be inserted: 12

Enter key at which element to be inserted: 3

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 15

Enter correct option

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 1

Enter element to be inserted: 15

Enter key at which element to be inserted: 8

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 2

Enter key of the element to be searched: 6

Element at key 6 : 7

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 3

Enter key of the element to be deleted: 2

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 2

Enter key of the element to be searched: 2

No element found at key 2

1.Insert element into the table

2.Search element from the key

3.Delete element at a key

4.Exit

Enter your choice: 4