###

### Client

###

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include "item.h"

#include "st.h"

int main () {

Item item;

Key k;

int i, cont, maxN;

printf("Input size of hash table to guarantee load factor < 1/2:");

scanf("%d", &maxN);

ST st = STinit(maxN);

cont = 1;

while(cont) {

printf("\nOperations on hash tables\n");

printf("===============\n");

printf("1.Insert\n") ; printf("2.Search\n");

printf("3.Delete\n"); printf("4.Display\n");

printf("5.Exit\n"); printf("Enter your choice : ");

if(scanf("%d",&i)<=0) {

printf("Integers only!\n");

exit(0);

}

else {

switch(i) {

case 1: printf("Enter item: \n");

item = ITEMscan();

if (ITEMcheckvoid(STsearch(st, KEYget(item))))

STinsert(st, item);

break;

case 2: printf("Enter key: \n");

k = KEYscan();

if (ITEMcheckvoid(STsearch(st, k)))

printf("data with given key not found!\n");

else

printf("data with given key found!\n");

break;

case 3: printf("Enter key: \n");

k = KEYscan();

if (ITEMcheckvoid(STsearch(st, k)))

printf("data with given key not in symbol table!\n");

else

STdelete(st, k);

break;

case 4: STdisplay(st);

break;

case 5: cont = 0;

break;

default: printf("Invalid option\n");

}

}

}

return 0;

}

###

### st.h

###

#ifndef ST\_H\_DEFINED

#define ST\_H\_DEFINED

typedef struct symboltable \*ST;

ST STinit(int) ;

void STinsert(ST, Item) ;

Item STsearch(ST, Key) ;

void STdelete(ST, Key) ;

void STdisplay(ST) ;

#endif

###

### st.c

###

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "item.h"

#include "st.h"

typedef struct STnode\* link;

struct STnode { Item item; link next; } ;

struct symboltable { link \*heads; int M; link z; };

link NEW( Item item, link next) {

link x = malloc(sizeof \*x);

x->item = item; x->next = next;

return x;

}

ST STinit(int maxN) {

int i;

ST st = malloc(sizeof \*st) ;

st->M = maxN/5;

st->heads = malloc(st->M\*sizeof(link));

st->z = NEW(ITEMsetvoid(), NULL);

for (i=0; i < st->M; i++)

st->heads[i] = st->z;

return st;

}

int hash(Key v, int M) {

int h = 0, base = 127;

for ( ; \*v != '\0'; v++)

h = (base \* h + \*v) % M;

return h;

}

int hashU(Key v, int M) {

int h, a = 31415, b = 27183;

for ( h = 0; \*v != '\0'; v++, a = a\*b % (M-1))

h = (a\*h + \*v) % M;

return h;

}

void STinsert (ST st, Item item) {

int i = hash(KEYget(item), st->M);

printf("hash =%d\n", i);

st->heads[i] = NEW(item, st->heads[i]);

}

Item searchR(link t, Key k, link z) {

if (t == z)

return ITEMsetvoid();

if ((KEYcompare(KEYget(t->item), k))==0)

return t->item;

return

searchR(t->next, k, z);

}

Item STsearch(ST st, Key k) {

return searchR(st->heads[hash(k, st->M)], k, st->z);

}

link deleteR(link x, Key k) {

if ( x == NULL )

return NULL;

if ((KEYcompare(KEYget(x->item), k))==0) {

link t = x->next;

free(x);

return t;

}

x->next = deleteR(x->next, k);

return x;

}

void STdelete(ST st, Key k) {

int i = hash(k, st->M);

st->heads[i] = deleteR(st->heads[i], k);

}

void visitR(link h, link z) {

if (h == z)

return;

ITEMshow(h->item);

visitR(h->next, z);

}

void STdisplay(ST st) {

int i;

for (i=0; i < st->M; i++) {

printf("st->heads[%d] = ", i);

visitR(st->heads[i], st->z);

printf("\n");

}

}

###

### item.h

###

#ifndef \_DATO\_INCLUDED

#define \_DATO\_INCLUDED

#define MAXC 10

typedef struct item\* Item;

typedef char \*Key;

Item ITEMscan();

void ITEMshow(Item data);

int ITEMcheckvoid(Item data);

Item ITEMsetvoid();

Key KEYscan();

int KEYcompare(Key k1, Key k2);

Key KEYget(Item data);

#endif

###

### item.c

###

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include "item.h"

struct item { char \*name; int value; };

Item ITEMscan() {

char name[MAXC];

int value;

printf("name = ");

scanf("%s", name);

printf("value = ");

scanf("%d", &value);

Item tmp = (Item) malloc(sizeof(struct item));

if (tmp == NULL)

return ITEMsetvoid();

else {

tmp->name = strdup(name);

tmp->value = value;

}

return tmp;

}

void ITEMshow(Item data) {

printf("name = %s value = %d ", data->name, data->value);

}

int ITEMcheckvoid(Item data) {

Key k1, k2="";

k1 = KEYget(data);

if (KEYcompare(k1,k2)==0)

return 1;

else

return 0;

}

Item ITEMsetvoid() {

char name[MAXC]="";

Item tmp = (Item) malloc(sizeof(struct item));

if (tmp != NULL) {

tmp->name = strdup(name);

tmp->value = -1;

}

return tmp;

}

Key KEYscan() {

char name[MAXC];

Key k;

scanf("%s", name);

k = strdup(name);

return k;

}

int KEYcompare(Key k1, Key k2) {

return strcmp(k1,k2);

}

Key KEYget(Item data) {

return data->name;

}