#include <stdio.h>

#include <cs50.h>

#include <string.h>

#include <ctype.h>

#define HASHSIZE 5

typedef struct sllist

{

char\* val;

struct sllist\* next;

}

sllnode;

// прототипы функций

void create(sllnode\* HashTable[]);

unsigned int hash(char\* string);

void insert(sllnode\* HashTable[], char\* key);

void showtable (sllnode\* HashTable[]);

bool find(sllnode\* HashTable[], char\* key);

void del(sllnode\* HashTable[], char\* key);

void destroy(sllnode\* HashTable[]);

int main(void)

{

sllnode\* HashTable[HASHSIZE];

create(HashTable);

// слова для тестов

char\*string1="Ana";char\*string2="Auto";char\*string3="Banan";char\*string4="Biblt";char\*string5="Bolk";char\*string6="Bern";char\*string7="Dru";

insert(HashTable, string1);insert(HashTable, string2);insert(HashTable, string3);insert(HashTable, string4);insert(HashTable, string5);

insert(HashTable, string6);insert(HashTable, string7);

showtable (HashTable);

del(HashTable, "Bolk");

showtable (HashTable);

destroy(HashTable);

}

// удаляем таблицу и очищаем память

void destroy(sllnode\* HashTable[])

{

// проходимся по каждому связанному списку и удаляем его элементы

for (int i = 0; i < HASHSIZE; i++)

{

sllnode\* ptr = HashTable[i];

while (ptr->next != NULL)

{

del(HashTable, ptr->next->val);

}

}

// удаляем основу хэш-таблицы

for (int i = 0; i < HASHSIZE; i++)

{

free(HashTable[i]);

}

}

// удаляет из таблицы значение по ключу key

void del(sllnode\* HashTable[], char\* key)

{

// проверка, существует ли значение в таблице

if (find(HashTable, key))

{

int index = hash(key);

sllnode\* ptrprev = HashTable[index];

sllnode\* ptrcurr = HashTable[index]->next;

while (ptrcurr->val != key)

{

ptrcurr = ptrcurr->next;

ptrprev = ptrprev->next;

}

ptrprev->next = ptrcurr->next;

free(ptrcurr);

}

}

// ищет значение в хэш таблице

bool find(sllnode\* HashTable[], char\* key)

{

int index = hash(key);

sllnode\* ptr = HashTable[index];

// проверка первого элемента списка

if (ptr->val == key)

{

return true;

}

// цикл по элементам

while (ptr->next != NULL)

{

if (ptr->val == key)

{

return true;

}

else

{

ptr = ptr->next;

}

}

// проверка последнего элемента списка

if (ptr->val == key)

{

return true;

}

return false;

}

// выводит всю хеш-таблицу

void showtable (sllnode\* HashTable[])

{

for (int i = 0; i < HASHSIZE; i++)

{

sllnode\* ptr = HashTable[i];

while (ptr->next != NULL)

{

ptr = ptr->next;

if (ptr->val != NULL)

{

printf("%s ", ptr->val);

}

}

printf("\n");

}

}

// вставляет значение key в таблицу по индексу

void insert(sllnode\* HashTable[], char\* key)

{

int index = hash(key);

sllnode\* ptr = malloc(sizeof(sllnode));

// проверка на ошибку

if (ptr == NULL)

{

printf("Error1\n");

exit(1);

}

ptr->val = key;

ptr->next = HashTable[index]->next;

HashTable[index]->next = ptr;

}

// hash function по первой букве слова

unsigned int hash(char\* string)

{

int hash = toupper (string[0]) - 'A';

return hash % HASHSIZE;

}

// выделяем память под таблицу

void create(sllnode\* HashTable[])

{

for (int i = 0; i < HASHSIZE; i++)

{

HashTable[i] = malloc(sizeof(sllnode));

// проверяем на ошибку

if (HashTable[i] == NULL)

{

printf("Error1\n");

exit(1);

}

HashTable[i]->val = NULL;

HashTable[i]->next = NULL;

}

}