// BST.h

typedef struct treeNode {

element key;

struct treeNode \*left;

struct treeNode \*right;

} treeNode;

treeNode\* searchBST(treeNode\* root, element key) ;

treeNode\* insertKey(treeNode \*p, element key) ;

void insert(treeNode\*\* root, element key);

void deleteNode(treeNode \*root, element key);

void menu();

void inorderDisplay(treeNode\* root);

//---------------------------------------------------------------------------------------------------

// BST.c

#include "std.h"

treeNode\* searchBST(treeNode\* root, element key) {

treeNode\* p; int compare;

p = root;

// printf("\n찾을 단어 : %s", key.word);

// printf("\n현재단어 : %s", p->key.word);

while (p != NULL)

{

compare = strcmp(key.word, p->key.word);

if(compare < 0) {

p = p->left;

// printf("\n검색 중 왼쪽단어 : %s", p->key.word);

}

else if(compare > 0) {

p = p->right;

// printf("\n검색 중 오른쪽 단어 : %s", p->key.word);

}

else {

printf("\n찾은 단어 : %s", p->key.word);

return p;

}

}

return p;

}

treeNode\* insertKey(treeNode \*p, element key) {

treeNode \*newNode;

int compare;

if (p == NULL) {

newNode = (treeNode\*)malloc(sizeof(treeNode));

newNode->key = key;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

else {

compare = strcmp(key.word, p->key.word);

if(compare < 0) {

p->left = insertKey(p->left, key);

return p;

}

else if(compare > 0){

p->right = insertKey(p->right, key);

return p;

}

else {

printf("\n 이미 같은 단어가 있습니다! \n");

return p;

}

}

}

void insert(treeNode\*\* root, element key)

{

\*root = insertKey(\*root, key);

}

void deleteNode(treeNode \*root, element key)

{

treeNode \*parent, \*p, \*succ, \*succ\_parent;

treeNode \*child;

parent=NULL;

p=root;

while((p != NULL) && (strcmp(p->key.word, key.word)!=0)){

parent=p;

if(strcmp(key.word, p->key.word)<0)

p=p->left;

else p=p->right;

}

if(p == NULL) {

printf("\n 찾는 키가 이진트리에 없습니다!!");

return;

}

// 단말노드의 삭제

if((p->left == NULL) && (p->right == NULL)){

if(parent != NULL){

if(parent->left == p) parent->left=NULL;

else parent->right=NULL;

}

else root=NULL;

}

// 자식노드가 한 개인 노드의 삭제

else if((p->left == NULL) || (p->right == NULL)){

if(p->left != NULL) child=p->left;

else child=p->right;

if(parent != NULL){

if(parent->left == p) parent->left=child;

else parent->right=child;

}

else root=child;

}

// 자식노드가 두 개인 노드의 삭제

else{

succ\_parent=p;

succ=p->right;

while(succ->left != NULL){

succ\_parent=succ;

succ=succ->left;

}

if(succ\_parent->left == succ)

succ\_parent->left=succ->right;

else succ\_parent->right=succ->right;

p->key=succ->key;

p=succ;

}

free(p);

}

void inorderDisplay(treeNode\* root)

{

if(root){

inorderDisplay(root->left);

printf("\n%s : %s", root->key.word, root->key.mean);

inorderDisplay(root->right);

}

}

//---------------------------------------------------------------------------------------------------

// data.h

#define MAX\_WORD\_LENGTH 20

#define MAX\_MEAN\_LENGTH 200

typedef struct

{

char word[MAX\_WORD\_LENGTH];

char mean[MAX\_MEAN\_LENGTH];

} element;

//----------------------------------------------------------------------------------------------------------------

#include "std.h"

void menu()

{

printf("\n\*---------------------------\*");

printf("\n\t1 : 입력");

printf("\n\t2 : 삭제");

printf("\n\t3 : 검색");

printf("\n\t4 : 출력");

printf("\n\t5 : 종료");

printf("\n\*---------------------------\*\n");

}

void main()

{

char choice;

element e;

treeNode \*root=NULL, \*temp;

do{

menu();

choice = getchar(); getchar();

switch(choice-'0')

{

case 1 :

printf("\n[단어 입력] 단어를 입력하세요 : "); gets(e.word);

printf("\n[단어 입력] 단어 뜻을 입력하세요 : "); gets(e.mean);

insert(&root, e);

break;

case 2 :

printf("\n[단어 삭제] 삭제할 단어 : "); gets(e.word);

deleteNode(root, e);

break;

case 3 :

printf("\n[단어 검색] 검색할 단어 : "); gets(e.word);

temp=searchBST(root, e);

if(temp != NULL)

printf("\n단어 뜻 : %s", temp->key.mean);

else printf("\n사전에 없는 단어입니다.");

break;

case 4 :

printf("\t[사전 출력]");

inorderDisplay(root); printf("\n\t[사전 끝]\n");

break;

}

}while((choice-'0') != 5);

}