240110 지난 시간 복습

1. 배열을 이용한 파일 데이터 선언 및 DB쿼리 프로그래밍 연습

2. 자기참조 구조체의 연결리스트를 이용한 파일 데이터 선언 및 DB쿼리 프로그래밍 연습

구조체 SAMPLE

struct self\_ex{

long id;

char grade;

int score;

struct\_self\_ex \*next;

}

자기참조 구조체의 SAMPLE

struct self\_ex{

long id;

char grade;

int score;

self\_ex\_pointer \*next;

}

/\*

240110 1번문제 연결리스트 구현(164~165p)

10-11주차 pdf 참조

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//////////////////////////////////////////////////////////////////////////////////////////////////

typedef struct fnode \*fpointer;

struct fnode {

char animal[10];

int item;

fpointer next;

};

void main()

{

fpointer head = NULL, ptr, another, before;

int k, how\_many = 0;

before = NULL;

head = (fpointer)malloc(sizeof(struct fnode));

strcpy(head->animal, "dog");

head->item = 10;

head->next = NULL;

for (k = 0; k < 4; k++) {

another = (fpointer)malloc(sizeof(struct fnode));

printf("Enter the sort of animal and its number : ");

scanf("%s %d", (another->animal), &(another->item));

ptr = head;

while (ptr != NULL) {

before = ptr;

ptr = ptr->next;

}

before->next = another;

another->next = NULL;

}

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

ptr = head;

while (ptr != NULL) {

printf("% s: % d\n", ptr->animal, ptr->item);

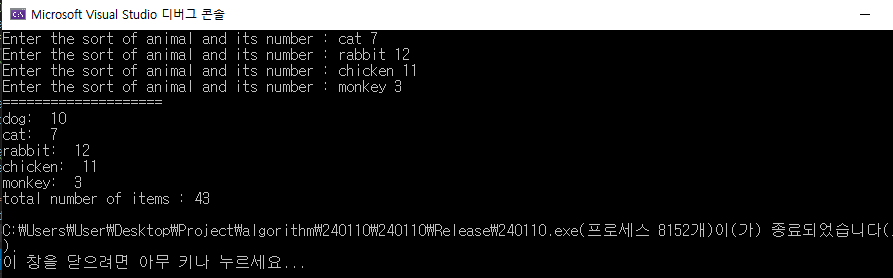
how\_many += ptr->item;

ptr = ptr->next;

}

printf("total number of items : %d\n", how\_many);

}



참고사항 : Before 함수 초기화 필요(before = NULL) or sdl검사 아니오 설정

/\*

240110 2번문제 연결리스트 구현(166p)

10-11주차 pdf 참조

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//////////////////////////////////////////////////////////////////////////////////////////////////

typedef struct list\_node \*list\_pointer;

struct list\_node {

int data;

list\_pointer link;

};

void print\_list(list\_pointer ptr);

list\_pointer make\_node();

int nodesum(list\_pointer ptr);

int nodenumber(list\_pointer ptr);

void main()

{

list\_pointer ptr;

ptr = make\_node();

print\_list(ptr);

printf("list안의 데이터 합: %d\n", nodesum(ptr));

printf("list안의 node 수 : %d\n", nodenumber(ptr));

}

list\_pointer make\_node()

{

list\_pointer first, second, third;

first = (list\_pointer) malloc(sizeof(struct list\_node));

second = (list\_pointer) malloc(sizeof(struct list\_node));

third = (list\_pointer) malloc(sizeof(struct list\_node));

third ->link = NULL;

second->link = third;

first->link = second;

first->data = 100;

second->data = 200;

third->data = 300;

return first;

}

void print\_list(list\_pointer ptr)

{

printf("The list contains: ");

for (; ptr; ptr = ptr->link)

printf("%4d", ptr->data);

printf("\n");

}

int nodenumber(list\_pointer ptr)

{

int count = 0;

for (; ptr; ptr = ptr->link)

count++;

return count;

}

int nodesum(list\_pointer ptr) {

int result = 0;

while (ptr != NULL) {

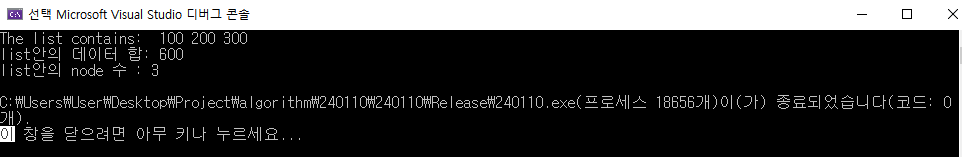
result += ptr->data;

ptr = ptr->link;

}

return result;

}



참고사항 : int nodesum은 직접 코딩하는 것이었음

240110 이진 검색 트리 구축과 응용

10-11주차 pdf 참조

이진 검색트리의 정의

1) 모든 원소는 키 값을 갖는다.

2) 왼쪽 서브 트리의 키들은 그의 루트의 키보다 작아야 한다.

3) 오른쪽 서브 트리의 키들은 그 루트의 키보다 커야 한다.

4) 왼쪽과 오른쪽 서브 트리도 또한 이진검색 트리이다(재귀적 성질)

/\*

240110 3번문제 이진 검색 트리 구축과 응용

10-11주차 pdf 참조

1) 검색은 루트로부터 시작한다.

1) key가 루트의 key값과 같다면 검색 성공

2) key가 루트의 key보다 작다면 왼쪽 서브 트리를 검색한다.

3) key가 루트의 key보다 크다면 오른쪽 서브 트리를 검색한다.

4) 서브 트리들은 재귀적으로 검색이 이루어진다.

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//////////////////////////////////////////////////////////////////////////////////////////////////

//2. 이진 트리의 노드 구조 정의를 위한 선언

typedef struct node {

struct node\* left;

char\* word;

struct node\* right;

}NODETYPE;

int insert\_node(NODETYPE\* root, char\* word);

NODETYPE\* search(NODETYPE\* root, char\* key);

void inorder(NODETYPE\* ptr);

void main()

{

char wbuf[30];

char key;

NODETYPE\* tree = NULL;

NODETYPE\* ptr;

printf("검색 트리에 저장할 단어를 입력하세요.\n");

printf("입력의 끝에는 quit를 입력하세요\n");

while (strcmp(gets(wbuf), "quit")) {

if (!tree)

{

tree = (NODETYPE\*)malloc(sizeof(NODETYPE));

tree->word = (char\*)malloc(strlen(wbuf + 1));

strcpy(tree->word, wbuf);

tree->left = tree->right = NULL;

}

else

insert\_node(tree, wbuf);

}

printf("검색할 데이터 : \n");

gets(wbuf);

ptr = search(tree, wbuf);

printf("-----트리 안의 단어들 (사전식 순서) -----\n");

inorder(tree);

}

int insert\_node(NODETYPE\* root, char\* word) {

NODETYPE\* tptr = root, \*before = NULL;

int cmp;

while (tptr) {

cmp = strcmp(word, tptr->word);

if (cmp < 0) {

before = tptr;

tptr = tptr->left;

}

else if (cmp > 0) {

before = tptr;

tptr = tptr->right;

}

else // found

return 0;

}

tptr = (NODETYPE\*)malloc(sizeof(NODETYPE));

tptr->word = (char\*)malloc(strlen(word) + 1);

strcpy(tptr->word, word);

tptr->left = tptr->right = NULL;

if (cmp < 0) before->left = tptr;

else before->right = tptr;

return 1;

}

NODETYPE\* search(NODETYPE\* root, char\* key)

{

NODETYPE\* tptr = root;

int cmp;

while (tptr) {

cmp = strcmp(key, tptr->word);

if (cmp < 0)

tptr = tptr->left;

else if (cmp > 0)

tptr = tptr->right;

else // found

return tptr;

}

return NULL; // not found

}

//이진검색트리의 운행 함수

void inorder(NODETYPE\* ptr) {

if (ptr) {

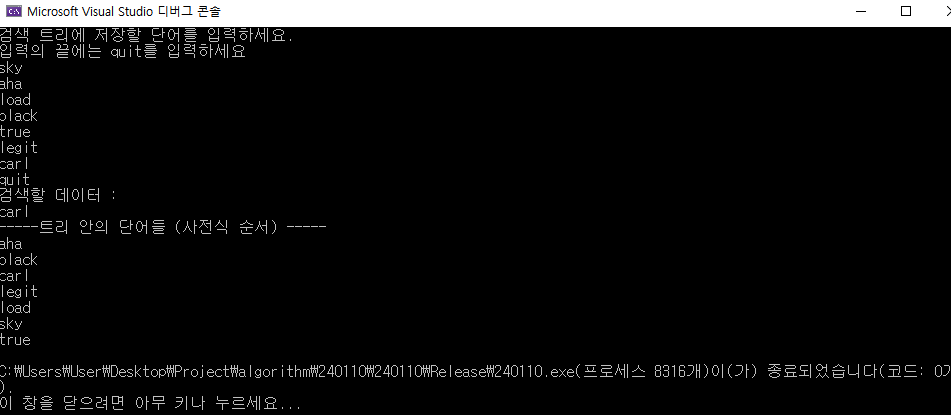
inorder(ptr->left);

printf("%s\n", ptr->word);

inorder(ptr->right);

}

}



/\*

240110 4번문제 이진 검색 트리 구축과 응용

10-11주차 pdf 참조, 262p 확인학습 6-2

1) 중위 운행 결과

2) insert\_node, search, inorder 함수 정의

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//////////////////////////////////////////////////////////////////////////////////////////////////

//2. 이진 트리의 노드 구조 정의를 위한 선언

typedef struct node {

struct node\* left;

int\* num;

struct node\* right;

}NODETYPE;

int insert\_node(NODETYPE\* root, int\* num);

NODETYPE\* search(NODETYPE\* root, int\* key);

void inorder(NODETYPE\* ptr);

void main()

{

int wbuf[30];

int key;

int count = 0;

NODETYPE\* tree = NULL;

NODETYPE\* ptr;

printf("검색 트리에 저장할 정수를 입력하세요.\n");

printf("입력의 끝에는 -1를 입력하세요\n");

while (strcmp(gets(wbuf), "-1")) {

if (!tree)

{

tree = (NODETYPE\*)malloc(sizeof(NODETYPE));

tree->num = (int\*)malloc(strlen(wbuf + 1));

strcpy(tree->num, wbuf);

tree->left = tree->right = NULL;

}

else

insert\_node(tree, wbuf);

count++;

if (count > 30) {

printf("too much numbers! \n");

exit(1);

}

}

printf("-----트리 안의 정수들 (오름차순) -----\n");

inorder(tree);

}

int insert\_node(NODETYPE\* root, int\* num) {

NODETYPE\* tptr = root, \*before = NULL;

int cmp;

while (tptr) {

cmp = strcmp(num, tptr->num);

if (cmp < 0) {

before = tptr;

tptr = tptr->left;

}

else if (cmp > 0) {

before = tptr;

tptr = tptr->right;

}

else // found

return 0;

}

tptr = (NODETYPE\*)malloc(sizeof(NODETYPE));

tptr->num = (char\*)malloc(strlen(num) + 1);

strcpy(tptr->num, num);

tptr->left = tptr->right = NULL;

if (cmp < 0) before->left = tptr;

else before->right = tptr;

return 1;

}

NODETYPE\* search(NODETYPE\* root, char\* key)

{

NODETYPE\* tptr = root;

int cmp;

while (tptr) {

cmp = strcmp(key, tptr->num);

if (cmp < 0)

tptr = tptr->left;

else if (cmp > 0)

tptr = tptr->right;

else // found

return tptr;

}

return NULL; // not found

}

//이진검색트리의 운행 함수

void inorder(NODETYPE\* ptr) {

if (ptr) {

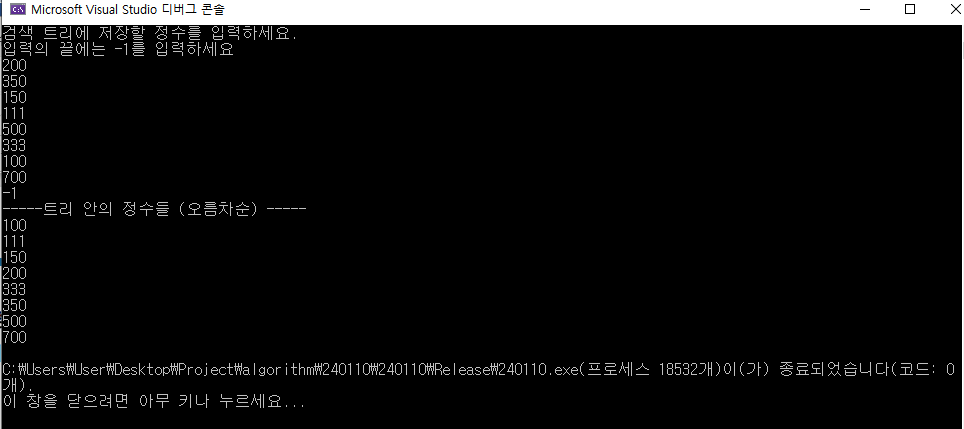
inorder(ptr->left);

printf("%s\n", ptr->num);

inorder(ptr->right);

}

}



/\*

240110 4번문제 이진 검색 트리 구축과 응용

10-11주차 pdf 참조, 262p 확인학습 6-2

1) 중위 운행 결과

2) insert\_node, search, inorder 함수 정의

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//////////////////////////////////////////////////////////////////////////////////////////////////

//2. 이진 트리의 노드 구조 정의를 위한 선언

typedef struct node {

struct node\* left;

char word;

int count;

struct node\* right;

}NODETYPE;

int insert\_node(NODETYPE\* root, char\* word);

NODETYPE\* search(NODETYPE\* root, char\* key);

void inorder(NODETYPE\* ptr);

void main()

{

char wbuf[30];

char key;

int cnum = 0;

NODETYPE\* tree = NULL;

NODETYPE\* ptr;

printf("검색 트리에 저장할 단어를 입력하세요.\n");

printf("입력의 끝에는 quit를 입력하세요\n");

while (strcmp(gets(wbuf), "quit")) {

if (!tree)

{

tree = (NODETYPE\*)malloc(sizeof(NODETYPE));

tree->word = (char\*)malloc(strlen(wbuf + 1));

strcpy(tree->word, wbuf);

// 첫 단어 카운트 = 1

tree->count + 1;

tree->left = tree->right = NULL;

}

else

insert\_node(tree, wbuf);

cnum++;

if (cnum > 30) {

printf("too much words! \n");

exit(1);

}

}

printf("-----트리 안의 정수들 (오름차순) -----\n");

inorder(tree);

}

int insert\_node(NODETYPE\* root, char\* word) {

NODETYPE\* tptr = root, \*before = NULL;

int cmp;

while (tptr) {

cmp = strcmp(word, tptr->word);

if (cmp < 0) {

before = tptr;

tptr = tptr->left;

}

else if (cmp > 0) {

before = tptr;

tptr = tptr->right;

}

else // found

(tptr->count)++;

return 0;

}

tptr = (NODETYPE\*)malloc(sizeof(NODETYPE));

tptr->word = (char\*)malloc(strlen(word) + 1);

strcpy(tptr->word, word);

tptr->count = 1;

tptr->left = tptr->right = NULL;

if (cmp < 0) before->left = tptr;

else before->right = tptr;

return 1;

}

NODETYPE\* search(NODETYPE\* root, char\* key)

{

NODETYPE\* tptr = root;

int cmp;

while (tptr) {

cmp = strcmp(key, tptr->word);

if (cmp < 0)

tptr = tptr->left;

else if (cmp > 0)

tptr = tptr->right;

else // found

return tptr;

}

return NULL; // not found

}

//이진검색트리의 운행 함수

void inorder(NODETYPE\* ptr) {

if (ptr) {

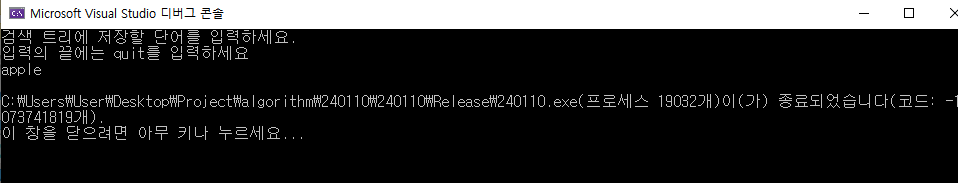
inorder(ptr->left);

printf("%s\n", ptr->word, ptr->count);

inorder(ptr->right);

}

}



컴파일 성공, 엑세스 위반 오류 발생

/\*

240110 6번문제 이진 검색 트리 구축과 응용

10-11주차 pdf 참조, 378p 9-15

학생정보 트리 탐색 예제 프로그램

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//////////////////////////////////////////////////////////////////////////////////////////////////

//2. 이진 트리의 노드 구조 정의를 위한 선언

typedef struct node {

struct node \*left;

long id;

float grade;

struct node \*right;

}SNODETYPE;

int insert\_node(SNODETYPE \*root, long sid, float grade);

SNODETYPE\* search(SNODETYPE \*root, long keyid);

void inorder(SNODETYPE \*root);

void over4(SNODETYPE \*root);

void main(int argc, char \*argv[])

{

SNODETYPE \*tree = NULL;

SNODETYPE \*ptr;

long sid;

float grade;

FILE \*input;

if ((input = fopen(argv[1], "r")) == NULL) {

printf("Invaild Open input file...\n");

exit(1);

}

while (fscanf(input, "%ld %f", &sid, &grade) != EOF) {

if (!tree)

{

tree = (SNODETYPE\*)malloc(sizeof(SNODETYPE));

tree->id = sid;

tree->left = tree->right = NULL;

tree->grade = grade;

}

else

insert\_node(tree, sid, grade);

}

printf("\n-----저장된 트리 순회 리스트-----\n");

inorder(tree);

printf("Enter student id number to search: ");

scanf("%ld", &sid);

ptr = search(tree, sid);

if (ptr)

printf("%ld is in this tree(%.2f).\n", ptr->id, ptr->grade);

else

printf("%ld is not exit. \n", sid);

printf("\n4.0이상의 학생 검색결과: \n");

over4(tree);

}

SNODETYPE \*search(SNODETYPE\* root, long keyid)

{

SNODETYPE\* tptr = root;

while (tptr) {

if (keyid < tptr -> id)

tptr = tptr->left;

else if (keyid > tptr->id)

tptr = tptr->right;

else // found

return tptr;

}

return NULL; // not found

}

int insert\_node(SNODETYPE \*root, long sid, float grade) {

SNODETYPE\* tptr = root, \*before = NULL;

int left;

while (tptr) {

if (sid < tptr->id) {

before = tptr;

tptr = tptr->left;

left = 1;

}

else if (sid > tptr->id) {

before = tptr;

tptr = tptr->right;

left = 0;

}

else {

return 0;

}

}

tptr = (SNODETYPE\*)malloc(sizeof(SNODETYPE));

tptr->id = sid;

tptr->left = tptr->right = NULL;

if (left) before->left = tptr;

else before->right = tptr;

return 1;

}

//이진검색트리의 운행 함수

void inorder(SNODETYPE \*ptr) {

if (ptr) {

inorder(ptr->left);

printf("%s\n", ptr->id);

inorder(ptr->right);

}

}

void over4(SNODETYPE \*ptr) {

if (ptr) {

over4(ptr->left);

if (ptr->grade > 4.0)

printf("%ld\t%.2f\n", ptr->id, ptr->grade);

over4(ptr->right);

}

}

