HW 4 : 트리

■ HW4_0

8장 트리의 Quiz/연습문제 중 일부

■ HW4_1(링크 표현법으로 구현한 이진 트리)

- void get_nonleaf_count(TreeNode *) : 이진 트리에서 비단말노드의 개수를 계산하는 함수를 작성해보자. 단말노드의 개수를 계산하는 교재의 <알고리즘 8.7>을 참조한다.
- void get_oneleaf_count(TreeNode *) : 이진 트리에서 자식이 하나인 노드의 개수를 반환
- void get_twoleaf_count(TreeNode *) : 이진 트리에서 자식이 둘인 노드의 개수를 반환
- int get_max (TreeNode *): 이진 트리에서 노드값들 중 최대값을 반환
- int get_min (TreeNode *): 이진 트리에서 노드값들 중 최소값을 반환
- int search(TreeNode *root, int key, (TreeNode *)[] result): 이진 트리에서 주어진 key 값을 갖는 노드들을 모두 찾아 그것들의 주소값을 result 배열에 저장하고 해당 노드의 갯수를 반환
- void node_increase(TreeNode *): 이진 트리의 노드들의 값을 1씩 증가
- int equal(TreeNode *, TreeNode *) : 두 개의 이진 트리가 같은 구조를 가지고 있고 대응되는 노드들이 같은 데이터를 가지고 있는지를 검사하여 참이면 1, 거짓이면 0을 반환
- TreeNode *copy(TreeNode *): 주어진 이진 트리를 복제한 새로운 트리를 반환

아래의 Skelecton(뼈대) Code에 위에 제시한 함수들을 작성하고 트리들을 이용하여 위의 함수들을 테스트한다.
교재의 preorder 함수를 이용하여 increase_node가 잘 실행되었나를 확인한다.
트리의 구조나 노드 값을 변경시키거나 주어진 main 함수를 변경시키며 다양하게 테스트해보라.

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#define TRUE 1
#define FALSE 0
#define MAX_TREE_SIZE 20

typedef struct TreeNode {
   int data;
   struct TreeNode *left, *right;
} TreeNode;

    //        root                                          root2
    //          15                                           15
    //     4          15                                 4          15
    //  15          16  25                            15          16  25
   //                                                                  28
TreeNode n1={15,  NULL, NULL};
TreeNode n2={4,  &n1,  NULL};
TreeNode n3={16, NULL, NULL};
TreeNode n4={25, NULL, NULL};
TreeNode n5={15, &n3,  &n4};
TreeNode n6={15, &n2,  &n5};
TreeNode *root= &n6;

TreeNode m1={15,  NULL, NULL};
TreeNode m2={4,  &n1,  NULL};
TreeNode m3={16, NULL, NULL};
TreeNode m7 = {28, NULL, NULL}; // 추가
TreeNode m4={25, NULL, &m7}; // 변경
TreeNode m5={15, &m3,  &m4};
TreeNode m6={15, &m2,  &m5};
TreeNode *root2= &m6;

//p281 Quiz 01
int get_nonleaf_count(TreeNode *t) {...}
//p281 Quiz 02
int equal(TreeNode *t1, TreeNode *t2) {...}
//p308 #25
int get_oneleaf_count(TreeNode *node) {...}
//p308 #26
int get_twoleaf_count(TreeNode *node) {...}
//p308 #27
int get_max(TreeNode *node) {...}
int get_min(TreeNode *node) {...}
//p308 #30
void node_increase(TreeNode *node) {...}
void preorder(TreeNode *root) // p271 코드 복사
{...}

int main(void)
{
     TreeNode *result[MAX_TREE_SIZE];
     TreeNode *clone;
     int i, num;

     printf("가)Wn");
     printf("트리 root 중 비단말노드의 개수는 %d.Wn", get_nonleaf_count(root));
     printf("트리 root2 중 비단말노드의 개수는 %d.Wn", get_nonleaf_count(root2));

     printf("트리 root 중 자식이 하나만 있는 노드의 개수는 %d.Wn", get_oneleaf_count(root));
     printf("트리 root2 중 자식이 하나만 있는 노드의 개수는 %d.Wn", get_oneleaf_count(root2));

     printf("트리 root 중 자식이 둘이 있는 노드의 개수는 %d.Wn", get_twoleaf_count(root));
     printf("트리 root2 중 자식이 둘이 있는 노드의 개수는 %d.Wn", get_twoleaf_count(root2));

     printf("트리 root 에서 가장 큰 수는 %d.Wn", get_max(root));
     printf("트리 root2 에서 가장 큰 수는 %d.Wn", get_max(root2));
     printf("트리 root 에서 가장 작은 수는 %d.Wn", get_min(root));
     printf("트리 root2 에서 가장 작은 수는 %d.Wn", get_min(root2));

     printf("Wn 나)Wn");
     num = search(root, 15, result);
     for (i = 0; i < num; i++)
          printf("(0x%p, %d), ", result[i], result[i]->data);
     printf("Wn");

     printf("Wn 다)Wn");
     preorder(root);
     node_increase(root);
     printf("Wn");
     preorder(root);
     printf("Wn");
     printf("%sWn", equal(root, root) ? "같다": "다르다");
     printf("%sWn", equal(root2, root2) ? "같다": "다르다");
     printf("%sWn", equal(root, root2) ? "같다": "다르다");

     printf("Wn 라)Wn");
     clone = copy(root) ;
     preorder(root) ;
     printf("Wn");
     preorder(clone) ;
     printf("Wn");
}
```

```
가)
트리 root중 비단말노드의 개수는 3.
트리 root2중 비단말노드의 개수는 4.
트리 root중 자식이 하나만 있는 노드의 개수는 1.
트리 root2중 자식이 하나만 있는 노드의 개수는 2.
트리 root중 자식이 둘이 있는 노드의 개수는 2.
트리 root2중 자식이 둘이 있는 노드의 개수는 2.
트리 root에서 가장 큰 수는 25.
트리 root2에서 가장 큰 수는 28.
트리 root에서 가장 작은 수는 4.
트리 root2에서 가장 작은 수는 4.

나)
(0x00BD803C, 15), (0x00BD8000, 15), (0x00BD8030, 15),

다)
15 4 15 15 16 25
16 5 16 16 17 26
같다
같다
다르다

라)
16 5 16 16 17 26
16 5 16 16 17 26
계속하려면 아무 키나 누르십시오 . . .
```

■ HW4_2(이진 탐색 트리 연습)

사용자로부터 입력을 받아 이진 탐색 트리 안에 저장하는 것을 포함한 다음의 기능을 수행하는 프로그램을 작성하라. (교재에 있는 알고리즘이나 프로그램을 활용하라.)

- i(nsert): 입력
- d(elete): 삭제
- s(earch): 탐색하여 "있음" 또는 "없음"을 출력
- p(rint): preorder를 이용하여 노드의 값을 순서대로 출력
- h(eight): 트리의 높이를 반환
- c(ount): 노드의 개수를 반환

- m(ax): 가장 큰 값을 출력하는 int get_maximum(TreeNode *)를 작성하라. 이진 탐색 트리에서 가장 큰 수는 가장 오른 쪽에 있다. 따라서 오른쪽 자식 링크를 따라서 링크가 NULL이 될때까지 가장 오른쪽으로 가면 된다.

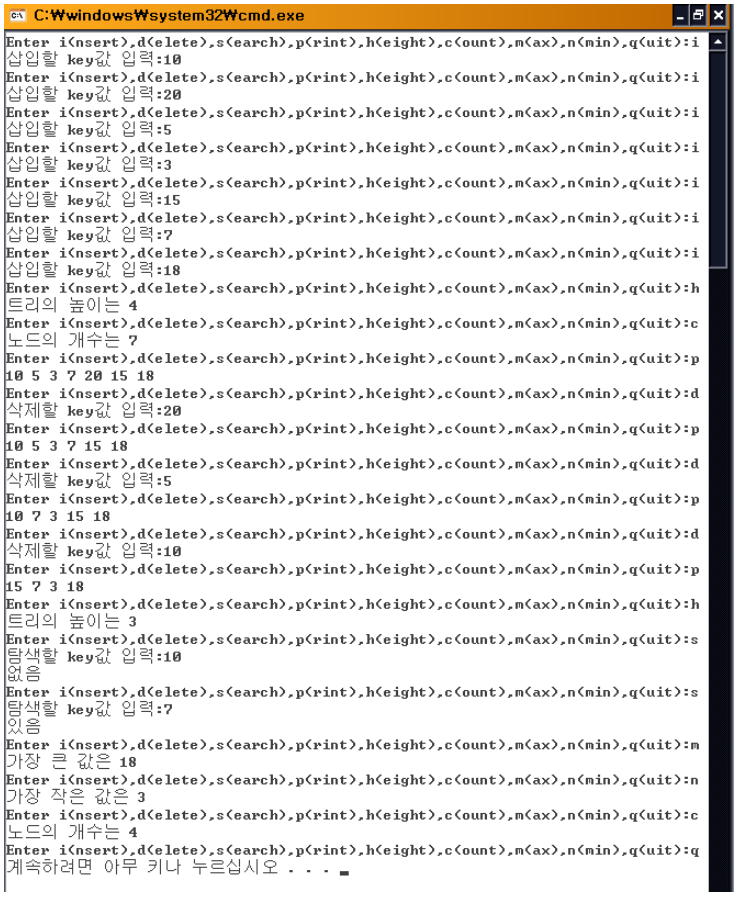
```
알고리즘: 이진 탐색 트리에서 최대값 탐색 알고리즘
get_maximum(x)
while RIGHT(x) ≠NULL
    do x <- RIGHT(x);
return DATA(x);
```

- m(min): 가장 작은 값을 출력하는 int get_miminum(TreeNode *)를 작성하라. 이진 탐색 트리에서 가장 작은 수는 가장 왼 쪽에 있다.
- q(uit): 종료

□ main함수에서 반복문을 이용하여 위의 함수들을 테스트하라 (p.304 프로그램 8.13의 main함수 참조).

□ 위의 i(nsert)를 이용하여 아래의 순서로 정수를 입력하고 이진탐색트리에 어떻게 저장되었는지 그림으로 그려보라. p(rint)를 이용하여 출력해보고 자신의 예상과 같은지 살펴보라.
입력 순서: 10->20->5->3->15->7->18

□ 그 외 다른 함수들도 모두 테스트해보라. **테스트를 통해 이진탐색트리의 삽입과 삭제를 충분히 익힌다.**



■ HW4_3

□ HW4_3_0(이진 탐색 트리 활용)

이진 탐색 트리를 이용하여 학생들과 관련된 자료를 저장하고 탐색하는 프로그램을 개발하여 보자. 하나의 학생은 학번(정수), 이름(문자열), 전화번호(문자열), 소속학과(문자열)의 정보를 가지고 있다.

이들 정보를 **학번을 키로 하여** 이진 탐색 트리에 저장하고 다음과 같은 메뉴가 가능하도록 프로그램을 작성하라. 학번 순으로 출력하는 것은 이진 탐색 트리의 중위(inorder) 순회 시 정렬된 숫자가 얻어지는 것을 이용하여 구현하라. 출력결과는 아래와 같다.

학생 정보 검색 프로그램
- i(nsert) : 학생 정보 입력.
- d(elete): 학번으로 학생 정보를 삭제
- s(earch): 학번으로 학생 정보를 탐색하여 모든 내용을 출력.
- p(rint): 학생 정보를 학번 순으로 출력
- c(ount student): 현재 저장된 학생들의 총 숫자를 출력
- q(uit): 종료

TreeNode 를 아래와 같이 정의하고 교재에 주어진 함수들을 그대로 혹은 조금 변형하여 사용한다.

```
#define MAX_STRING 100

typedef struct {
  int id;
  char name[MAX_STRING];
  char tel[MAX_STRING];
  char dept[MAX_STRING];
} element;

typedef struct TreeNode {
  element data;
  struct TreeNode *left, *right;
} TreeNode;
```

<HW4_3_0 의 실행예>

```
Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):i
학번 입력:3333
이름 입력:박수희
전화번호 입력:010-3333
학과 입력:컴퓨터학과
Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):i
학번 입력:1111
이름 입력:이지아
전화번호 입력:010-1111
학과 입력:문창과
Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):i
학번 입력:2222
이름 입력:김태희
전화번호 입력:010-2222
학과 입력:의상학과
Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):c
현재 저장된 학생의 총 수는 3
Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):p
학생 정보 학번 순 출력
--------------------------------------
학번: 1111
이름: 이지아
전화번호: 010-1111
학과: 문창과
--------------------------------------
학번: 2222
이름: 김태희
전화번호: 010-2222
학과: 의상학과
--------------------------------------
학번: 3333
이름: 박수희
전화번호: 010-3333
학과: 컴퓨터학과

Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):s
탐색할 학번 입력:3333
--------------------------------------
학번: 3333
이름: 박수희
전화번호: 010-3333
학과: 컴퓨터학과
Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):s
탐색할 학번 입력:4444
id가 4444인 학생은 없읍니다.
Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):d
삭제할 학번 입력:3333
Enter i(nsert), d(elete), s(earch), p(rint), c(ount), q(uit):p
학생 정보 학번 순 출력
--------------------------------------
학번: 1111
이름: 이지아
전화번호: 010-1111
학과: 문창과
--------------------------------------
학번: 2222
이름: 김태희
전화번호: 010-2222
학과: 의상학과
```

□ *HW4_3_1(이진 탐색 트리 활용 2)*

*위의 프로그램을 **이름을 키로하여** 다시 작성하라.*
어느 부분을 고쳐야하는가?
(교재 7.8 절(이진탐색트리의 응용: 영어 사전)을 참조할 수 있다.)

즉,
학생 정보 검색 프로그램은 다음과 같이 수정된다.
- *i(nsert) : 학생 정보 입력.*
- *d(elete): **이름**으로 학생 정보를 삭제*
- *s(earch): **이름**으로 학생 정보를 탐색하여 모든 내용을 출력.*
- *p(rint): 학생 정보를 **이름** 순으로 출력*
- *c(ount student): 현재 저장된 학생들의 총 숫자를 출력*
- *q(uit): 종료*