데이터 구조

프로젝트 2

Contents

- **Binary Search Tree를 2가지 타입으로 구현하여 관련된 함수들을 제작**
- **제출 파일 : tree.h, tree.c (main함수는 교수가 따로 사용할 예정)**
  - 제출 기한 : 12/24 ~ 23:59 (~12/25 10:00)
  - 조건
    - BST를 배열과 연결리스트 형태로 모두 구현한다.
    - Insert, Delete 등의 함수를 각각의 형태에서 구현하고, 서로 변환이 가능하도록 함수를 제작한다.
    - 검색을 실패하는 경우에 대한 출력은 해야하지만, 입력에 관한 예외처리는 없다고 가정한다.
    - 전역 변수 써도 상관 없습니다.
    - Leftnode의 개념(작다) = 사전적으로 앞에 온다. (a<->z)

자료 구조의 형태

❖ tree.h

```
#define STRINGSIZE 30
#define MAXDEGREE 6


typedef struct tree_Array                    //배열 형태로 저장하는 트리
{
        char** data;
        int maxindex;
}tree_A;


typedef struct treenode                      //리스트 트리를 위한 노드
{
        char data[STRINGSIZE];
        struct treenode* left;
        struct treenode* right;
}treenode;

typedef struct tree_LinkedList               //리스트 형태로 저장하는 트리
{
        treenode* root;
}tree_LL;
```

프로그램 구현 내용

❖ tree.h

```
tree_A* CreateArrayTree();
tree_LL* CreateLListTree();

void InsertNodeA(tree_A *t,char *str);
void InsertNodeLL(tree_LL *t,char* str);

void DeleteNodeA(tree_A* t, char* str);
void DeleteNodeLL(tree_LL* t, char* str);

tree_A* List2Array(tree_LL* t);
tree_LL* Array2List(tree_A* t);

void PrintNodeA(tree_A* t, int type);
void PrintNodeLL(tree_LL* t, int type);

void CheckBSTA(tree_A* t);
void CheckBSTLL(tree_LL* t);

char* FindnthA(tree_A* t, int num);
char* FindnthLL(tree_LL* t, int num);

void FindPathA(tree_A* t, char* str);
void FindPathLL(tree_LL* t, char* str);
```

프로그램 구현 내용

❖ tree.c

tree_A* CreateArrayTree();
>> 트리를 배열 형태로 생성 (MAXDGREE를 기반으로 배열을 동적할당)
tree_LL* CreateLListTree();
>> 트리를 연결리스트 형태로 생성 (빈 공간 생성)

void InsertNodeA(tree_A *t,char *str);
>> 배열 형태의 트리에서 str 을 추가
Void InsertNodeLL(tree_LL *t,char* str);
>> 연결리스트 형태의 트리에서 str을 추가

Void DeleteNodeA(tree_A* t, char* str);
>> 배열 형태의 트리에서 str을 삭제
void DeleteNodeLL(tree_LL* t, char* str);
>> 연결리스트 형태의 트리에서 str을 삭제

tree_A* List2Array(tree_LL* t);
>> 연결 리스트 형태의 트리를 배열 형태의 트리로 변경 하여 리턴
tree_LL* Array2List(tree_A* t);
>> 배열 형태의 트리를 연결리스트 형태로 리턴

프로그램 구현 내용

❖ tree.c

void PrintNodeA(tree_A* t, int type);
void PrintNodeLL(tree_LL* t, int type);
>> 트리를 출력 하는데 type 에 따라서 출력
0: pre, 1: in, 2: post order 로 출력하고 3: level order

Void CheckBSTA(tree_A* t);
Void CheckBSTLL(tree_LL* t);
>> t의 형태를 출력(다음 슬라이드 참고)

Char* FindnthA(tree_A* t, int num);
Char* FindnthLL(tree_LL* t, int num);
>> num 번째로 나오는 문자열을 리턴

Void FindPathA(tree_A* t, char* str);
Void FindPathLL(tree_LL* t, char* str);
>> str을 찾아가는 과정을 출력(다음 슬라이드 참고)

프로그램 구현 내용

❖ tree.c

- **Void CheckBSTA(tree_A* t);**
- **Void CheckBSTLL(tree_LL* t);**

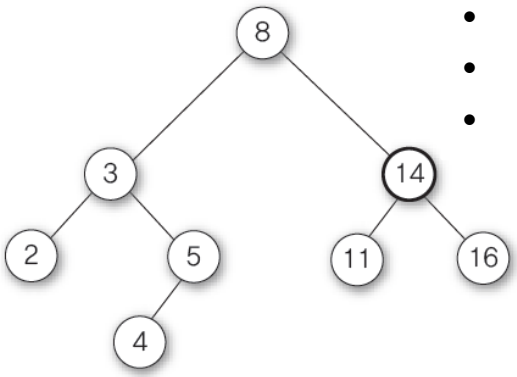
  - 현재 트리의 형태가 무엇인지를 출력

  1) "포화 이진트리"
  2) "완전 이진트리"
  3) "편향 이진트리"
  4) "일반 이진트리" 입니다. 라고 출력

- **Void FindPathA(tree_A* t, char* str);**
- **Void FindPathLL(tree_LL* t, char* str);**

  >> root를 기준으로 str을 찾아가는 과정을 출력

  왼쪽의 경우 L, 오른쪽의 경우 R을 출력하고, 만약 str이 없는 경우에는 데이터가 없습니다. 라고 출력



- 4를 검색했다면. : "LRL" 출력
- 6을 검색했다면. : "데이터가 없습니다." 출력
- 16을 검색했다면: "RR" 출력