자료구조론

Binary Tree Traversal

u@hataeseong-ui-MacBook-Pro:~/Desktop/2017_CSE2010_2016025041/HW4$./a.out input.txt

Preorder: 54 27 13 1 44 37 89 71 64 92

Inorder: 1 13 27 44 37 54 64 71 89 92

Postorder: 1 13 37 44 27 64 71 92 89 54

u@hataeseong-ui-MacBook-Pro:~/Desktop/2017_CSE2010_2016025041/HW4$

- 실행시 출력결과(일치)

```
while(1){
	command = fgetc(input);
	if(feof(input)) break;
	switch(command) {
		case 'i':
			printf("Inorder:   ");
			printInorder(root);
			printf("\n");
			break;
		case 'r':
			printf("Preorder:  ");
			printPreorder(root);
			printf("\n");
			break;
		case 'o':
			printf("Postorder: ");
			printPostorder(root);
			printf("\n");
```

```
                        break;
                default:
                        break;
        }
}

fclose(input);
free(root);
return 0;
```
- input 파일을 읽어와서 각 입력에 맞는 함수 호출

```
TreeNode * createTreeNode(void)
{
        TreeNode *tmp;
        tmp = (TreeNode *)malloc(sizeof(TreeNode));
        tmp->left = NULL;
        tmp->right = NULL;
        return tmp;
}
```
- 동적할당 후 노드 리턴

```
void printInorder(TreeNode *root){
        if(root != NULL){
                printInorder(root->left);
                printf("%d ", root->data);
                printInorder(root->right);
        }
}
```
- inorder : 왼쪽 트리 -> 루트 -> 오른쪽 트리 순
- 간편하게 재귀함수로 구현

```
void printPreorder(TreeNode *root){
	if(root != NULL){
		printf("%d ", root->data);
		printPreorder(root->left);
		printPreorder(root->right);
	}
}
```

- preorder : 루트 -> 왼쪽 트리 -> 오른쪽 트리 순
- 마찬가지로 재귀함수로 구현

```
void printPostorder(TreeNode *root){
	if(root != NULL){
		printPostorder(root->left);
		printPostorder(root->right);
		printf("%d ", root->data);
	}
}
```

- postorder : 왼쪽 트리 -> 오른쪽 트리 -> 루트 순
- 재귀함수로 구현