

. , 가 2 . , 가

	가
	,
	( + 1)
	가
	가 0

tree\_traverse.cpp

```
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    Node *left;
    Node *right;
};
Node *Root;

void InitTree(int data)
{
    Root=(Node *)malloc(sizeof(Node));
    Root->data=data;
}

Node *AddChild(Node *Parent,int data,bool bLeft)
{
    Node *New;

    New=(Node *)malloc(sizeof(Node));
    New->data=data;
    New->left=NULL;
    New->right=NULL;
```

```
    if (bLeft) {
        Parent->left=New;
    } else {
        Parent->right=New;
    }
    return New;
}

void PreOrder(Node *R)
{
    printf("%d ",R->data);
    if (R->left) PreOrder(R->left);
    if (R->right) PreOrder(R->right);
}

void InOrder(Node *R)
{
    if (R->left) InOrder(R->left);
    printf("%d ",R->data);
    if (R->right) InOrder(R->right);
}

void PostOrder(Node *R)
{
    if (R->left) PostOrder(R->left);
    if (R->right) PostOrder(R->right);
    printf("%d ",R->data);
}

void FreeTree(Node *R)
{
    if (R->left) FreeTree(R->left);
    if (R->right) FreeTree(R->right);
    free(R);
}

void main()
{
    Node *Left,*Right;

    InitTree(1);
    Left=AddChild(Root,2,true);
    Right=AddChild(Root,3,false);
    AddChild(Left,4,true);
    AddChild(Left,5,false);
    AddChild(Right,6,true);

    PreOrder(Root);puts("");
    InOrder(Root);puts("");
    PostOrder(Root);puts("");
}
```

```
FreeTree (Root) ;  
}
```

- 
- 

From:

<http://obg.co.kr/doku/> - **OBG WiKi**

Permanent link:

<http://obg.co.kr/doku/doku.php?id=programming:tree>

Last update: **2020/11/29 14:09**

