EDDI
Electronic Design
Development Institute

에디로봇아카데미
임베디드 마스터 Lv2 과정

제 1기

2021. 10. 22

김태훈

CONTENTS

EDDI
Electronic Design
Development Institute

** QUEUE는 지난번 숙제 때 해서 생략하겠습니다.

CONTENTS

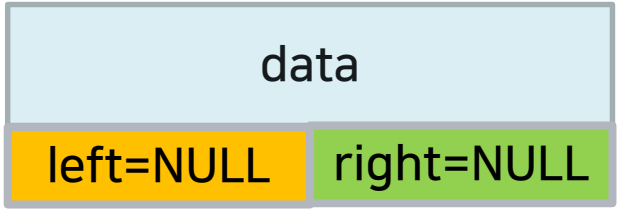
EDDI
Electronic Design Development Institute

Tree 구조체

```
typedef struct _tree tree;
struct _tree
{
    int data;
    struct _tree *left;
    struct _tree *right;
};

tree* create_tree_node(void)
{
    tree *tmp;

    tmp = (tree *)malloc(sizeof(tree));
    tmp->left = NULL;
    tmp->right = NULL;
    return tmp;
}
```



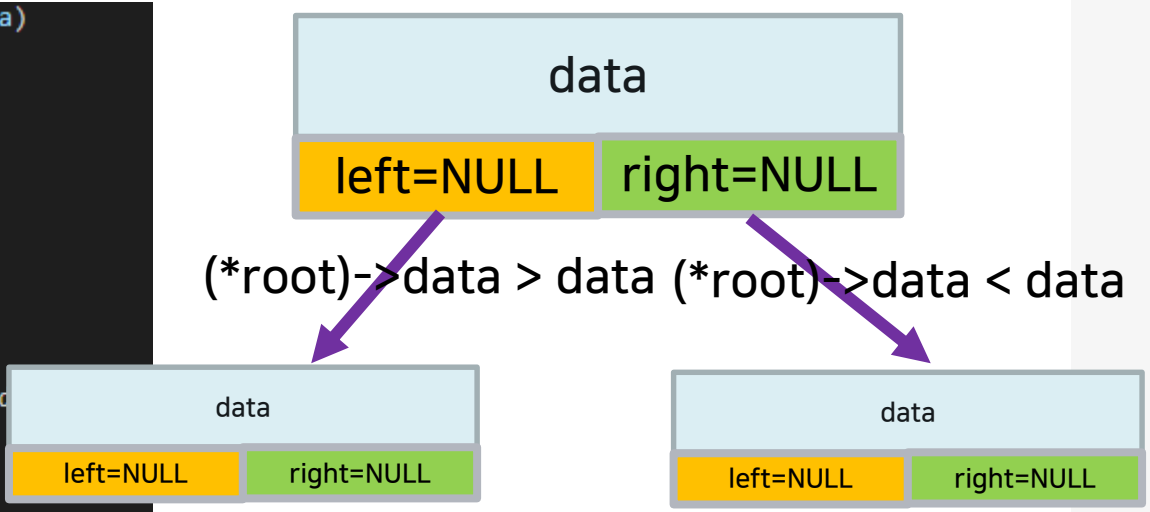
CONTENTS

EDDI
Electronic Design Development Institute

Insert(using Recursion)

```
void insert_tree_data(tree **root, int data)
{
    if(!(*root))
    {
        *root = create_tree_node();
        (*root)->data = data;
        return;
    }
    if((*root)->data < data)
    {
        insert_tree_data(&(*root)->right, d
    }
    else // (*root)->data > data
    {
        insert_tree_data(&(*root)->left, data);
    }

}
```



CONTENTS

EDDI
Electronic Design Development Institute

Insert(using loop)

```
void insert_tree_data(tree **root, int data)
{
    if(!(*root))
    {
        *root = create_tree_node();
        (*root)->data = data;
        return;
    }
    tree* tmp = *root;
```

```
    while(tmp)
    {
        if(tmp->data < data)
        {
            if(tmp->right)
                tmp = tmp->right;
            else
            {
                tmp->right = create_tree_node();
                tmp->right->data = data;
                return;
            }
        }
        else
        {
            if(tmp->left)
                tmp = tmp->left;
            else
            {
                tmp->left = create_tree_node();
                tmp->left->data = data;
                return;
            }

        }
    }
```

CONTENTS

EDDI
Electronic Design Development Institute

find(using recursion)

```
tree* find_tree_data(tree **root, int data)
{
    if(!(*root))
    {
        printf("there is no %d in tree\n",data);
        return *root;
    }
    if((*root)->data < data)
    {
        return find_tree_data(&(*root)->right,data);
    }
    else if((*root)->data > data)
    {
        return find_tree_data(&(*root)->left,data);
    }
    else
        return *root;
}
```

CONTENTS

EDDI
Electronic Design
Development Institute

delete(using recursion)

```
void delete_tree_data(tree **root, int data)
{
    if(!(*root))
    {
        printf("there is no %d in tree\n", data);
        return;
    }

    if((*root)->data < data)
        delete_tree_data(&(*root)->right,data);
    else if((*root)->data > data)
        delete_tree_data(&(*root)->left, data);
    else
```

```
else
{
    if((*root)->right && (*root)->left)
    {
            tree** cursor = &(*root)->right;
            while((*cursor)->left)
            {
                cursor = &(*cursor)->left;
            }
            tree* tmp2 = (*cursor);
            (*root)->data = (*cursor)->data;
            *cursor = (*cursor)->right;
            free(tmp2);
        }
        else if((*root)->right)
        {
            tree* tmp2 = *root;
            (*root) = (*root)->right;
            free(tmp2);
        }
        else if((*root)->left)
        {
            tree* tmp2 = *root;
            (*root) = (*root)->left;
            free(tmp2);
        }
        else
        {
            free(*root);
            *root = NULL;
        }

}
```

CONTENTS

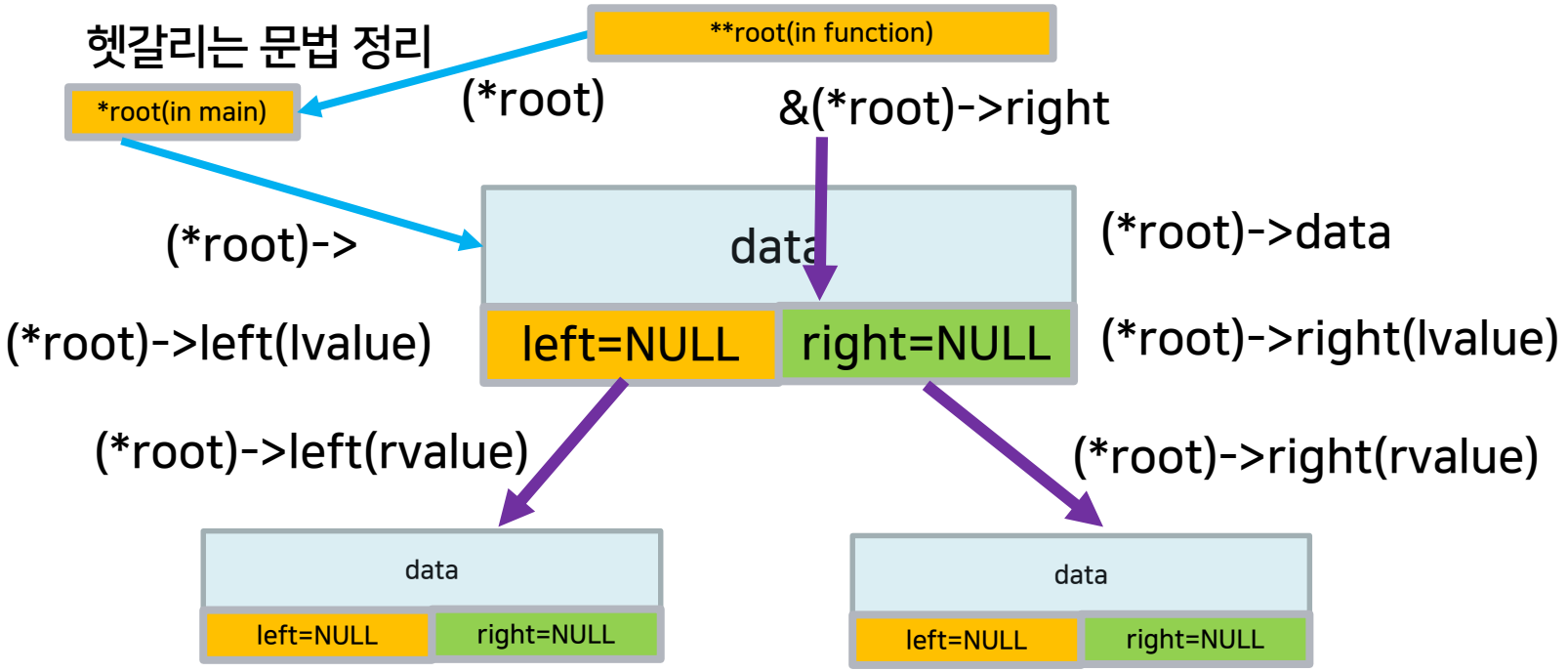
EDDI
Electronic Design
Development Institute

delete(using loop)

```
void delete_tree_data(tree **root, int data)
{
    tree** tmp = root;
    if(!(*tmp))
    {
        printf("there is no %d in tree\n",data);
        return;
    }
    while((*tmp)->data != data)
    {
        if((*tmp)->data < data)
        {
            tmp = &(*tmp)->right;
        }
        else
        {
            tmp = &(*tmp)->left;
        }
        if(!(*tmp))
        {
            printf("there is no %d in tree\n", data);
            return;
        }
    }
```

```
    if((*tmp)->right && (*tmp)->left)
    {
        tree** cursor = &(*tmp)->right;
        while((*cursor)->left)
        {
            cursor = &(*cursor)->left;
        }
        tree* tmp2 = (*cursor);
        (*tmp)->data = (*cursor)->data;
        *cursor = (*cursor)->right;
        free(tmp2);
    }
    else if((*tmp)->right)
    {
        tree* tmp2 = *tmp;
        (*tmp) = (*tmp)->right;
        free(tmp2);
    }
    else if((*tmp)->left)
    {
        tree* tmp2 = *tmp;
        (*tmp) = (*tmp)->left;
        free(tmp2);
    }
    else
    {
        free(*tmp);
        *tmp = NULL;
    }
```

CONTENTS



Lvalue(등호의 왼쪽) = 자리(메모리 공간)를 의미

Rvalue(등호의 오른쪽) = data 값을 의미

CONTENTS

EDDI
Electronic Design
Development Institute

헷갈리는 개념 정리

Function 호출시 : assembly 분석했던 것을 생각해보면
Main stack에서 parameter 값을 레지스터로 ‘복사 ' 해가서
Function의 stack frame을 만들고 지역변수 자리 마련하고 레지스터에서 값을 다시 ‘복사 ' 해온다.
이게 call by value라고 부르는 이유인데, 이 때문에 function에서는 값 자체를 바꾸든 대입을 하든 function stack frame에 있는 메모리 공간이기 때문에 Main엔 아무 영향이 없다.
Main에 영향을 주기 위해선 pointe를 복사해와서 main stack frame에 ‘직접’ 접근 해야 한다.