TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 문한나
mhn97@naver.com

Tree – 재귀함수 없이 구현하기(insert)

```
int main(void)
{
    int i;
    int data[14] = {50, 45, 73, 32, 48, 46, 16,
                    37, 120, 47, 130, 127, 124};

    tree *root = NULL;

    for(i = 0; data[i]; i++)
        non_recur_tree_ins(&root, data[i]);

    print_tree(&root);

    non_recur_delete_tree(&root, 50);
    printf("After Delete₩n");

    print_tree(&root);

    return 0;
}
```

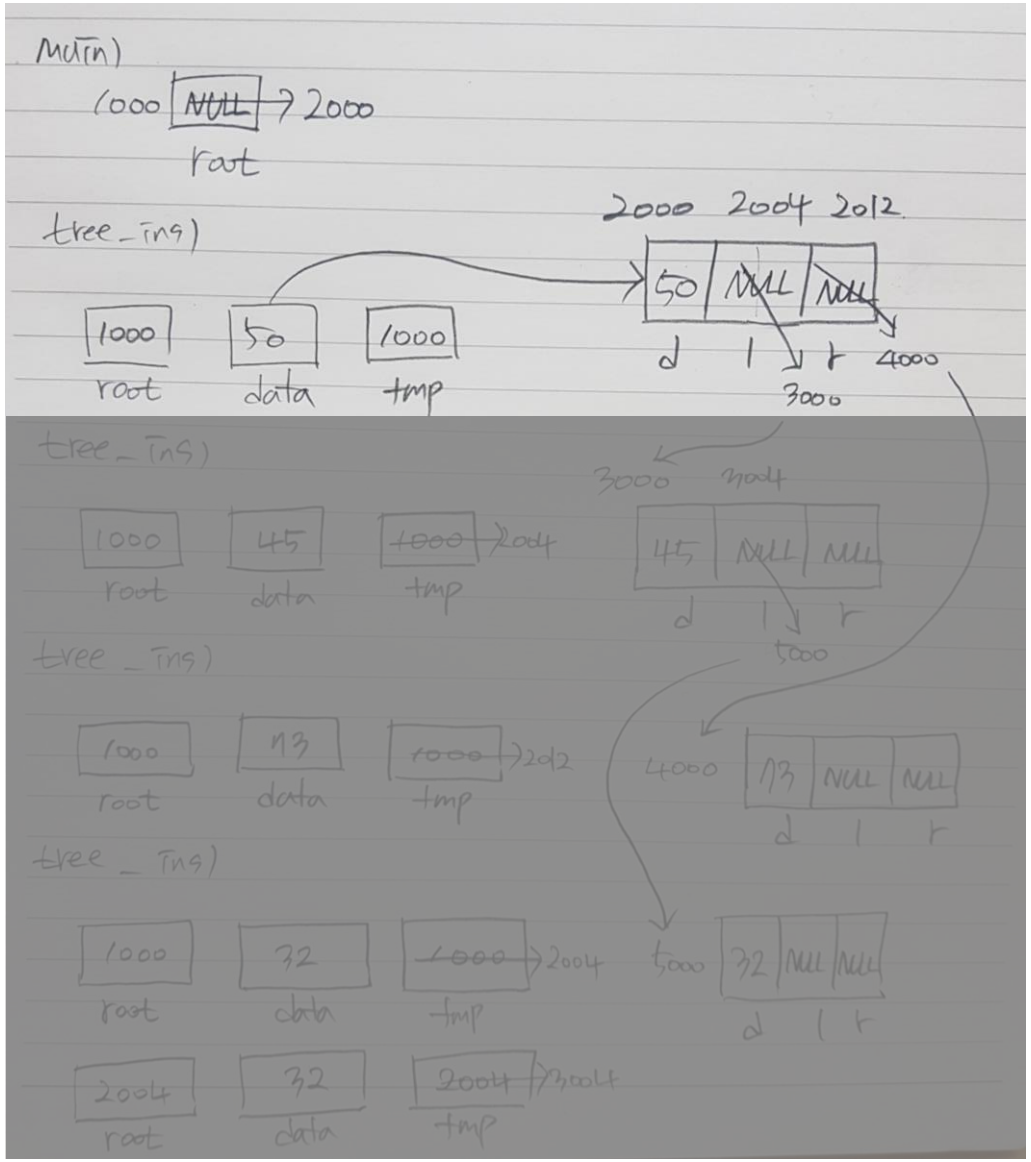
```
void non_recur_tree_ins(tree **root, int data)
{
    tree **tmp = root;

    while(*tmp)
    {
        if((*tmp)->data > data)
            tmp = &(*tmp)->left;
        else if((*tmp)->data < data)
            tmp = &(*tmp)->right;
    }

    *tmp = get_tree_node();
    (*tmp)->data = data;
}
```

조건문에 걸리지 않는다

```
stack *get_stack_node(void)
{
    stack *tmp;
    tmp = (stack *)malloc(sizeof(stack));
    tmp->link = NULL;
    return tmp;
}
```



```c
int main(void)
{
    int i;
    int data[14] = {50, 45, 73, 32, 48, 46, 16,
                    37, 120, 47, 130, 127, 124};

    tree *root = NULL;

    for(i = 0; data[i]; i++)
        non_recur_tree_ins(&root, data[i]);

    print_tree(&root);

    non_recur_delete_tree(&root, 50);
    printf("After Delete₩n");

    print_tree(&root);

    return 0;
}
```

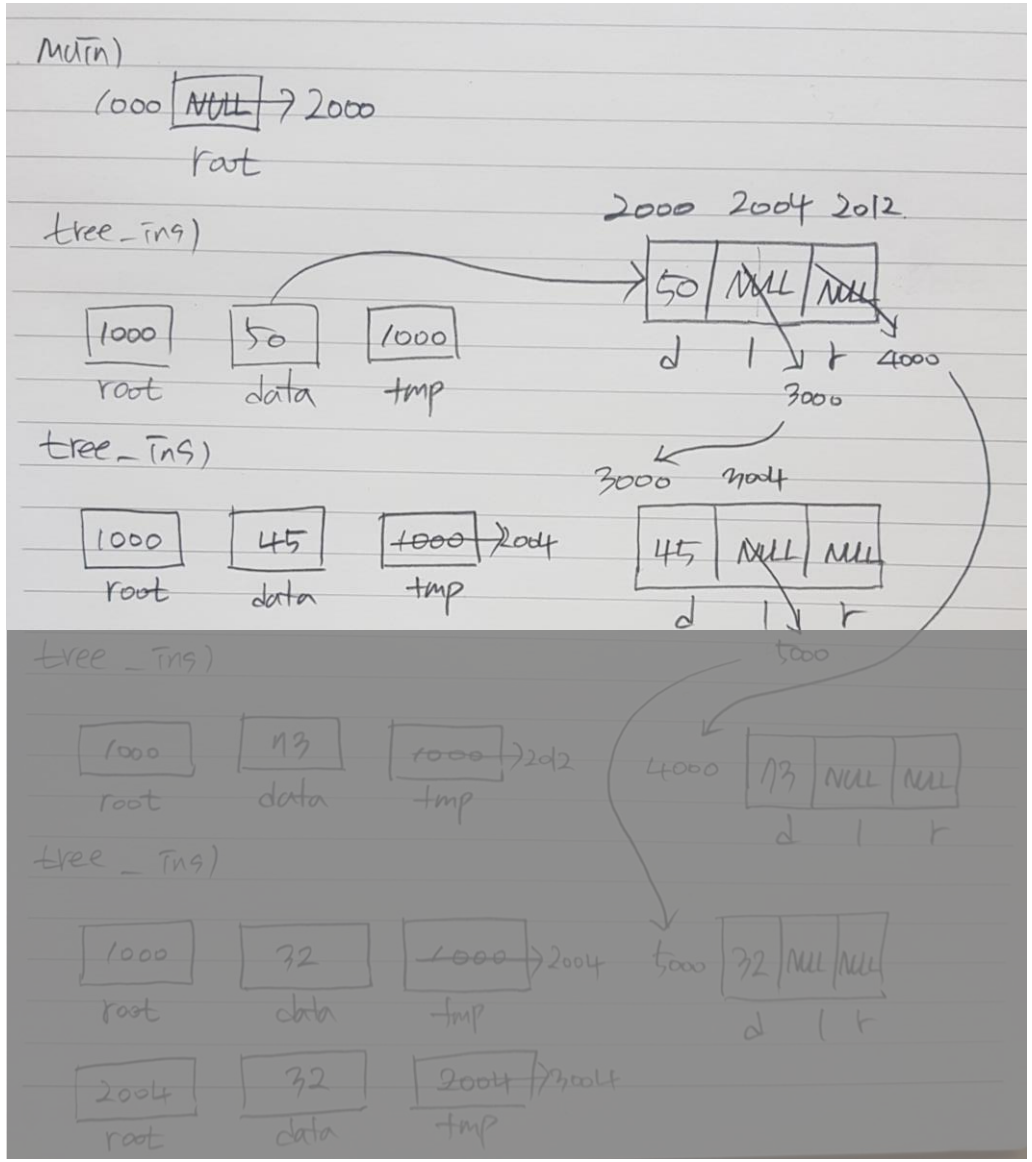
```c
void non_recur_tree_ins(tree **root, int data)
{
    tree **tmp = root;

    while(*tmp)
    {
        if((*tmp)->data > data)
            tmp = &(*tmp)->left;
        else if((*tmp)->data < data)
            tmp = &(*tmp)->right;
    }

    *tmp = get_tree_node();
    (*tmp)->data = data;
}
```

```c
stack *get_stack_node(void)
{
    stack *tmp;
    tmp = (stack *)malloc(sizeof(stack));
    tmp->link = NULL;
    return tmp;
}
```

tmp의 left 주소에 노드를 추가한다



```
int main(void)
{
    int i;
    int data[14] = {50, 45, 73, 32, 48, 46, 16,
                    37, 120, 47, 130, 127, 124};

    tree *root = NULL;

    for(i = 0; data[i]; i++)
        non_recur_tree_ins(&root, data[i]);

    print_tree(&root);

    non_recur_delete_tree(&root, 50);
    printf("After Delete₩n");

    print_tree(&root);

    return 0;
}
```

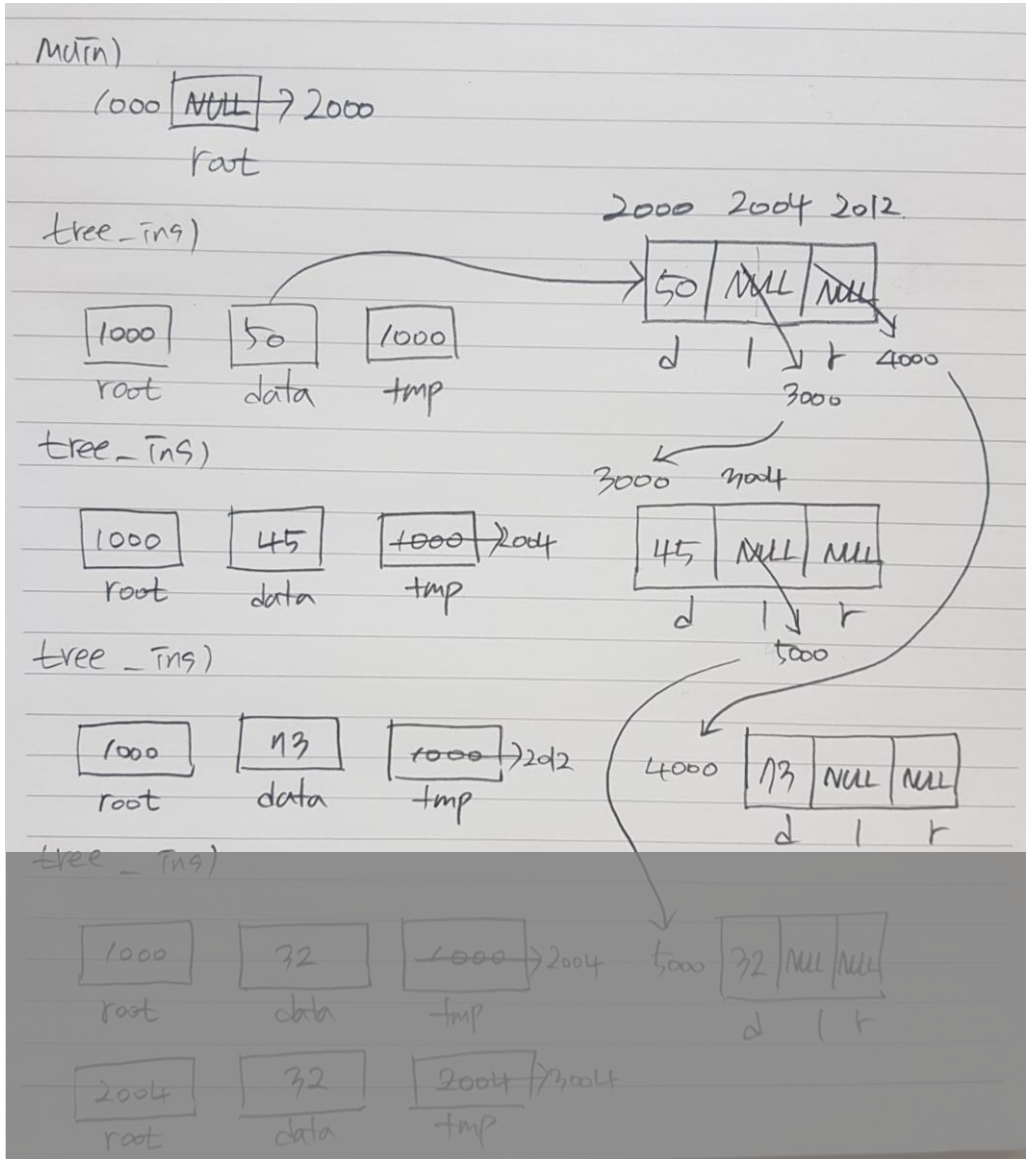
```
void non_recur_tree_ins(tree **root, int data)
{
    tree **tmp = root;

    while(*tmp)
    {
        if((*tmp)->data > data)
            tmp = &(*tmp)->left;
        else if((*tmp)->data < data)
            tmp = &(*tmp)->right;
    }

    *tmp = get_tree_node();
    (*tmp)->data = data;
}
```

tmp의 right 주소에 노드를 추가한다

```
stack *get_stack_node(void)
{
    stack *tmp;
    tmp = (stack *)malloc(sizeof(stack));
    tmp->link = NULL;
    return tmp;
}
```



```c
int main(void)
{
    int i;
    int data[14] = {50, 45, 73, 32, 48, 46, 16,
                    37, 120, 47, 130, 127, 124};

    tree *root = NULL;

    for(i = 0; data[i]; i++)
        non_recur_tree_ins(&root, data[i]);

    print_tree(&root);

    non_recur_delete_tree(&root, 50);
    printf("After Delete\n");

    print_tree(&root);

    return 0;
}
```

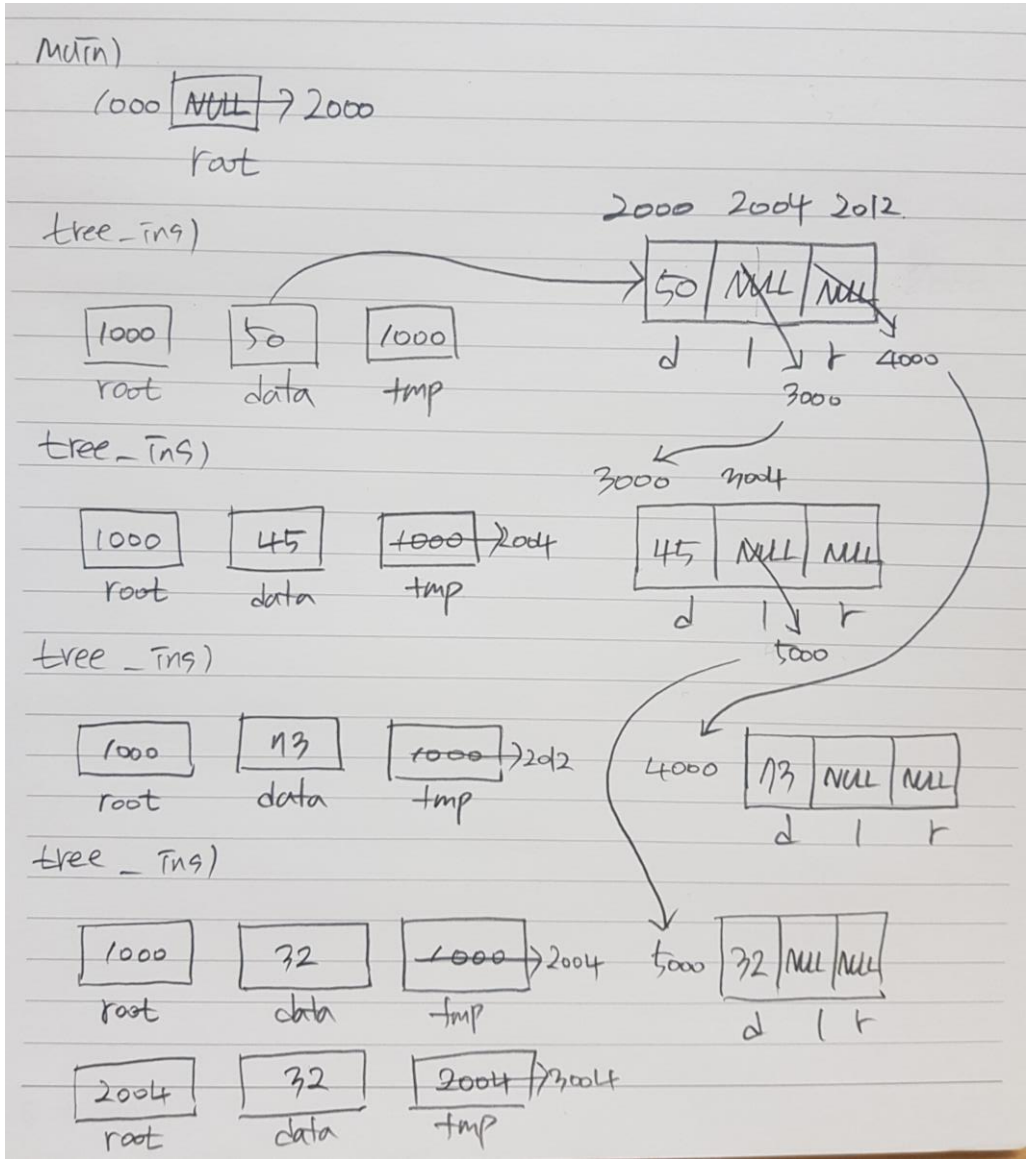
```c
void non_recur_tree_ins(tree **root, int data)
{
    tree **tmp = root;

    while(*tmp)
    {
        if((*tmp)->data > data)
            tmp = &(*tmp)->left;
        else if((*tmp)->data < data)
            tmp = &(*tmp)->right;
    }

    *tmp = get_tree_node();
    (*tmp)->data = data;
}
```

```c
stack *get_stack_node(void)
{
    stack *tmp;
    tmp = (stack *)malloc(sizeof(stack));
    tmp->link = NULL;
    return tmp;
}
```

data값을 비교하여 조건문에 걸리지 않을 때 노드를 추가한다



Tree – 재귀함수 없이 구현하기(delete)