TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018.03.13
14 일차
강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 신민철
akrn33@naver.com

이진트리
(재귀함수 비호출)

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

typedef struct __tree
{
	int data;
	struct __tree *left;
	struct __tree *right;
} tree;

typedef struct __stack
{
	void *data;
	struct __stack *link;
} stack;

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

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

void *pop(stack **top)
{
	stack *tmp = *top;
	void *data = NULL;

	if(*top == NULL)
	{
		printf("stack is empty!\n");
		return NULL;
	}

	data = (*top)->data;
	*top = (*top)->link;
	free(tmp);

	//return (*top)->data;
	return data;
}

void push(stack **top, void *data)
{
	if(data == NULL)
		return;

	stack *tmp = *top;
```

```
        *top = get_stack_node();
        (*top)->data = malloc(sizeof(void *));
        (*top)->data = data;
        (*top)->link = tmp;
}

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;
}

bool stack_is_not_empty(stack *top)
{
        if(top != NULL)
                return true;
        else
                return false;
}

void print_tree(tree **root)
{
        tree **tmp = root;
        stack *top = NULL;
```

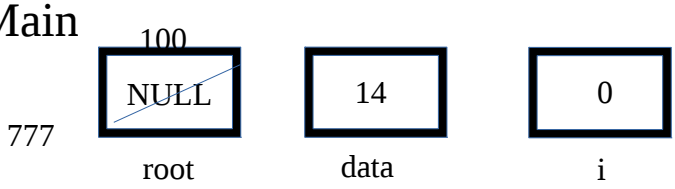
```
	push(&top, *tmp);

	while(stack_is_not_empty(top))
	{
		tree *t = (tree *)pop(&top);
		tmp = &t;

		printf("data = %d, ", (*tmp)->data);

		if((*tmp)->left)
			printf("left = %d, ", (*tmp)->left->data);
		else
			printf("left = NULL, ");

		if((*tmp)->right)
			printf("right = %d\n", (*tmp)->right->data);
		else
			printf("right = NULL\n");

		push(&top, (*tmp)->right);
		push(&top, (*tmp)->left);

		//tmp = &(*tmp)->left;

		//*tmp = (tree *)pop(&top);
	}
}

#if 0
void print_tree(tree *root)
{
	if(root)
	{
		printf("data = %d, ", root->data);
```

```
            if(root->left)
                printf("left = %d, ", root->left->data);
            else
                printf("left = NULL, ");

            if(root->right)
                printf("right = %d\n", root->right->data);
            else
                printf("right = NULL\n");

            print_tree(root->left);
            print_tree(root->right);
        }
}
#endif

tree *chg_node(tree *root)
{
        tree *tmp = root;

        if(!root->right)
                root = root->left;
        else if(!root->left)
                root = root->right;

        free(tmp);

        return root;
}

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

        while(*tmp)
```

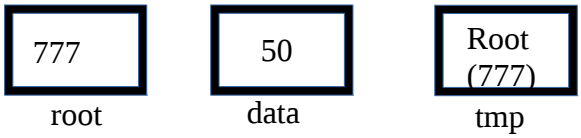
```
        {
                if((*tmp)->right)
                        tmp = &(*tmp)->right;
                else
                {
                        *data = (*tmp)->data;
                        *tmp = chg_node(*tmp);
                        break;
                }
        }
}

void non_recur_delete_tree(tree **root, int data)
{
        tree **tmp = root;
        int num;

        while(*tmp)
        {
                if((*tmp)->data > data)
                        tmp = &(*tmp)->left;
                else if((*tmp)->data < data)
                        tmp = &(*tmp)->right;
                else if((*tmp)->left && (*tmp)->right)
                {
                        find_max(&(*tmp)->left, &num);
                        (*tmp)->data = num;
                        return;
                }
                else
                {
                        (*tmp) = chg_node(*tmp);
                        return;
                }
        }
```

```
        printf("Not Found\n");
}

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;
}
```

Main

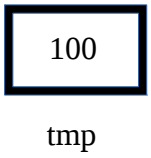


Non_recur_tree_ins

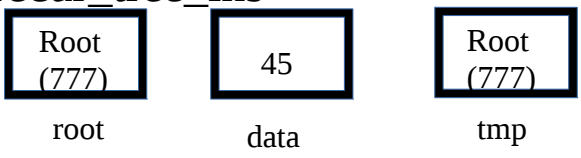


아직 트리에 데이터가 없어서

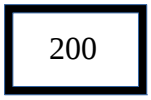
Get_tree_node



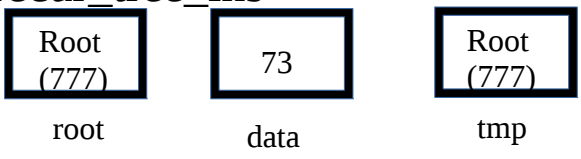
Non_recur_tree_ins



Get_tree_node

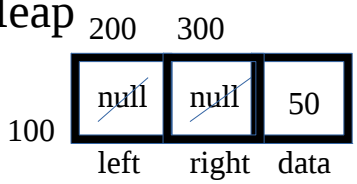


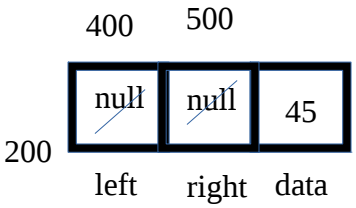
Non_recur_tree_ins

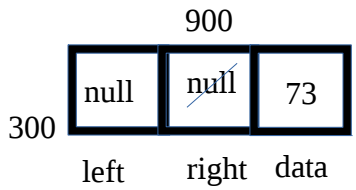


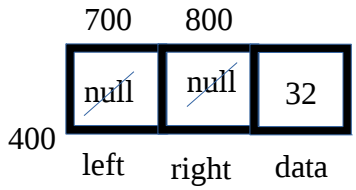
Get_tree_node

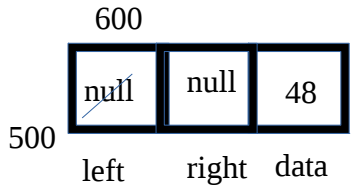
Heap

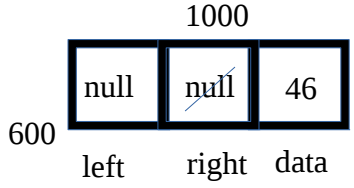




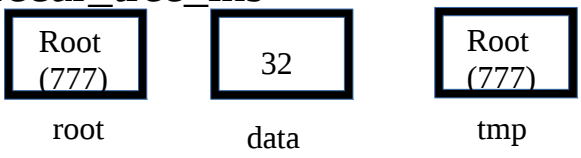


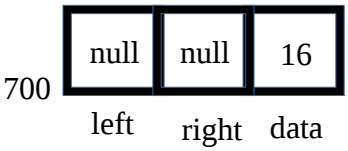






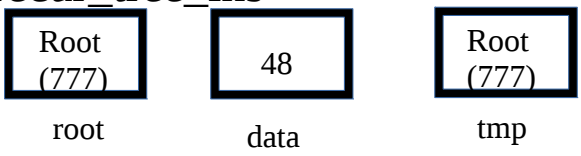
Non_recur_tree_ins

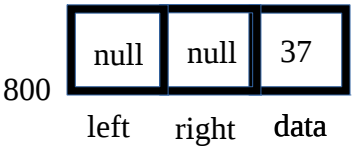




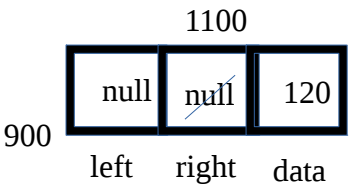
Get_tree_node

Non_recur_tree_ins

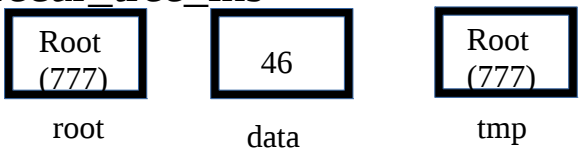




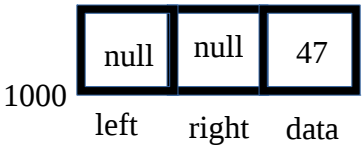
Get_tree_node



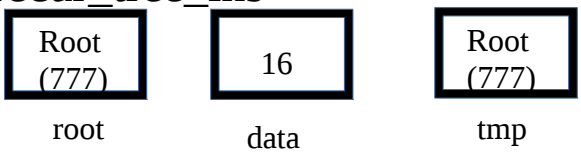
Non_recur_tree_ins



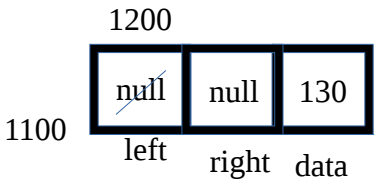
Get_tree_node



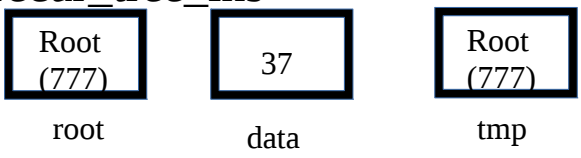
Non_recur_tree_ins



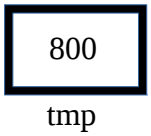
Get_tree_node

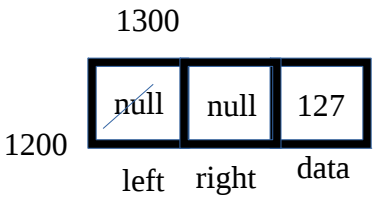


Non_recur_tree_ins

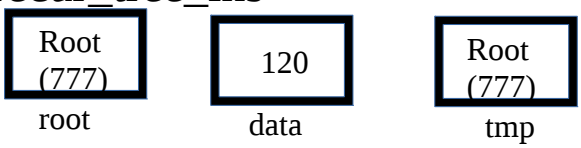


Get_tree_node

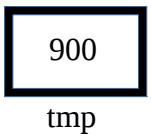


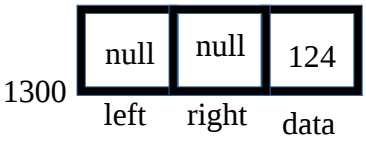


Non_recur_tree_ins

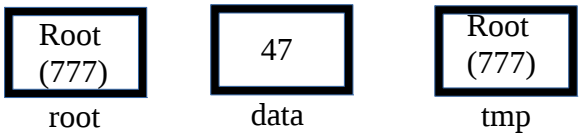


Get_tree_node

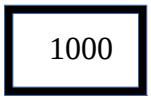




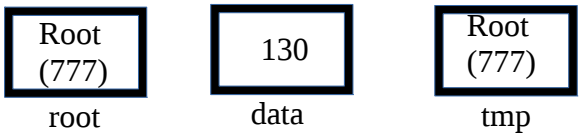
Non_recur_tree_ins



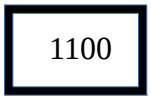
Get_tree_node



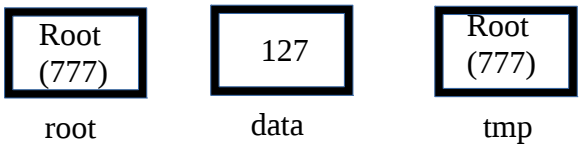
Non_recur_tree_ins



Get_tree_node



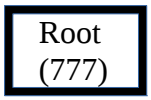
Non_recur_tree_ins



Get_tree_node

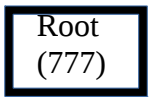
tmp

Non_recur_tree_ins



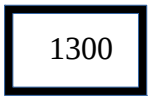
root

data



tmp

Get_tree_node



tmp

재귀함수를 없애니 함수호출 수가 재귀호출 때보다 현저히 줄어들었음을 알 수있다.

Print_tree

root

tmp

top

Push

top

data

tmp