TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018.03.09
12 일차
강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 신민철
akrn33@naver.com

이진트리

```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
#define EMPTY 0
struct node{
	int data;
	struct node* left;
	struct node* right;
};

typedef struct node tree;


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

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

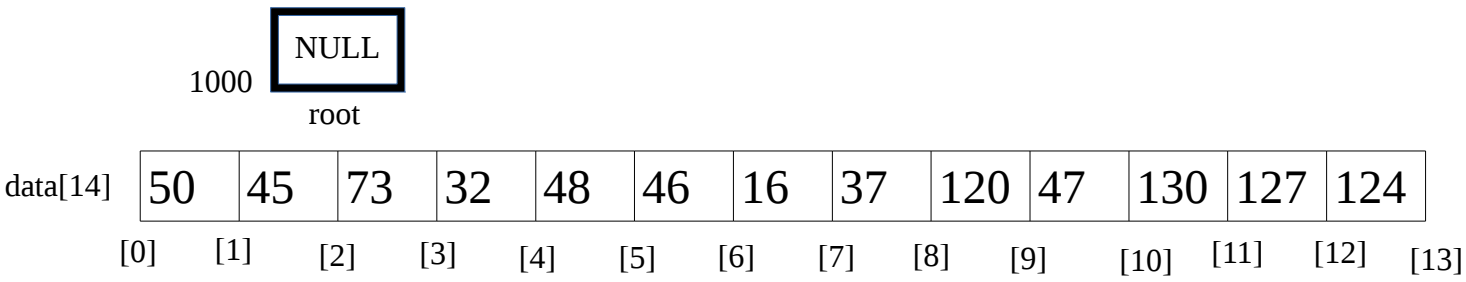
```
            root = root->right;

        free(tmp);

        return root;
}

tree* get_node()
{
        tree* tmp;
        tmp = (tree*)malloc(sizeof(tree));
        return tmp;
}

tree* find_max(tree* root, int* data)
{
        if(root->right)
                root->right = find_max(root->right, data);
        else
        {
                *data = root->data;
                root = chg_node(root);
        }
        return root;
}

tree* delete_tree(tree* root, int data)
{
        int num;
        tree* tmp;
        if(root == NULL)
        {
                printf("Not Found\n");
                return NULL;
        }
```

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

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);
	}
}
```

```
void tree_ins(tree** root, int data)
{
	if(*root == NULL)
	{
		*root = get_node();
		(*root)->data = data;
		return ;
	}
	else if((*root)->data >data)
		tree_ins(&(*root)->left, data);
	else if((*root)->data <data)
		tree_ins(&(*root)->right, data);

}

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

	tree* root = NULL;
	printf("root do = %p",&root);
	printf("data do =  %p\n",data);
	for(i = 0; data[i]; i++)
	{
		tree_ins(&root, data[i]);
	}
	print_tree(root);

	delete_tree(root, 50);
	printf("After Delete\n");
```

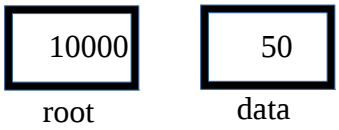
```
    print_tree(root);

    return 0;
}
```

Main

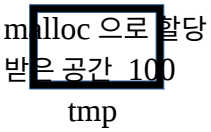


For

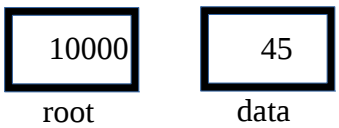
Tree_ins



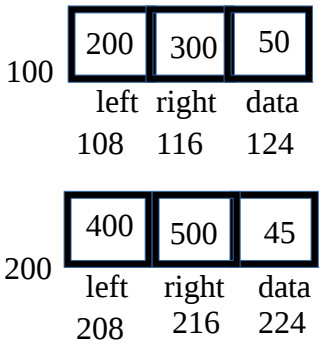
Get_node



Tree_ins



Heap



Tree_ins

108 (root)　45 (data)

Get_node

malloc 으로 할당 받은 공간 200

tmp

Tree_ins

10000 (root)　73 (data)

Tree_ins

116 (root)　73 (data)

Get_node

malloc 으로 할당 받은 공간 300

tmp

Tree_ins

10000 (root)　32 (data)

Tree_ins

108 (root)　32 (data)

Tree_ins

208 (root)　32 (data)

Get_node

malloc 으로 할당 받은 공간 400

tmp

Tree_ins

10000 (root)　48 (data)

Tree_ins

108 (root)　48 (data)

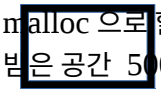
Tree_ins

216 (root)　48 (data)

	left	right	data
300		900	73
	308	316	324
400	700	800	32
	408	416	424
500	600		48
	508	516	524
600		1000	46
	608	616	624
700			16
	708	716	724
800			37
	808	816	824
900		1100	120
	908	916	924
1000			47
	1008	1016	1024
1100	1200		130
	1108	1116	1124
1200	1300		127
	1208	1216	1224
1300			124
	1208	1216	1224

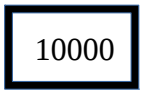
Get_node

malloc 으로 할당
받은 공간 500

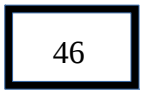


tmp

Tree_ins

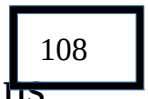


root

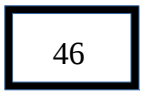


data

Tree_ins

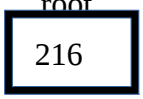


root

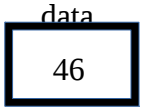


data

Tree_ins



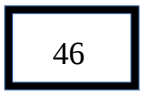
root



data

Tree_ins

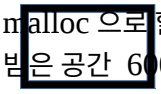
root



data

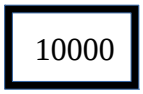
Get_node

malloc 으로 할당
받은 공간 600

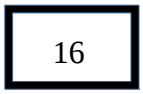


tmp

Tree_ins

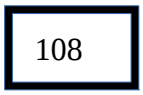


root

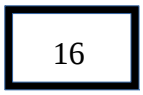


data

Tree_ins



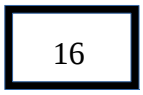
root



data

Tree_ins

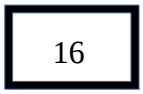
root



data

Tree_ins

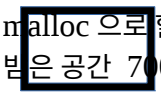
root



data

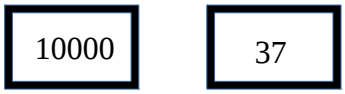
Get_node

malloc 으로 할당
받은 공간 700



tmp

Tree_ins



root data

Tree_ins

root data

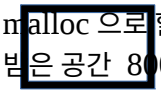
Tree_ins

root data

Tree_ins

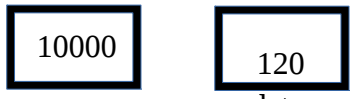
root data

Get_node



tmp

Tree_ins



root data

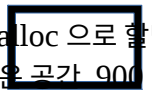
Tree_ins

root data

Tree_ins

root data

Get_node



tmp

Tree_ins

root data

Tree_ins

root data

Tree_ins

root data

Tree_ins

root data

Tree_ins

root data

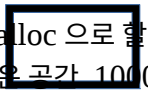
Tree_ins

root data

Tree_ins

root data

Get_node



malloc 으로 할당
받은 공간 1000

tmp

Tree_ins

root data

Tree_ins

root data

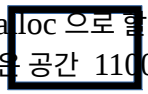
Tree_ins

root data

Tree_ins

root data

Get_node



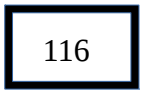
malloc 으로 할당
받은 공간 1100

tmp

Tree_ins

root data

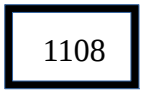
Tree_ins



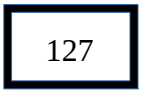
root

data

Tree_ins

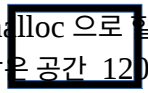


root



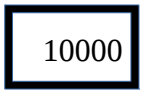
data

Get_node

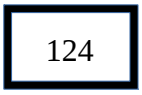


tmp

Tree_ins

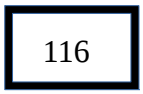


root

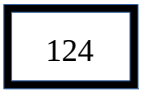


data

Tree_ins

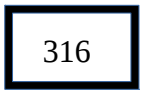


root



data

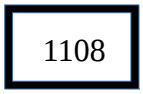
Tree_ins



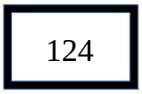
root

data

Tree_ins



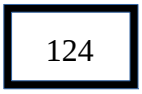
root



data

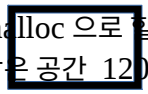
Tree_ins

root



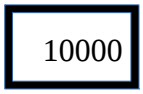
data

Get_node



tmp

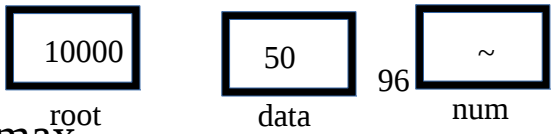
Print_tree



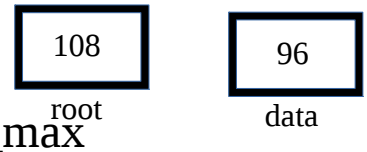
root

출력결과-→
data = 50, left = 45, right = 73
data = 45, left = 32, right = 48
data = 32, left = 16, right = 37
data = 16, left = NULL,right = NULL
data = 37, left = NULL,right = NULL
data = 48, left = 46, right = NULL
data = 46, left = NULL,right = 47v
data = 47, left = NULL,right = NULL
data = 73, left = NULL,right = 120
data = 120, left = NULL,right = 130
data = 130, left = 127, right = NULL
data = 127, left = 124, right = NULL
data = 124, left = NULL,right = NULL

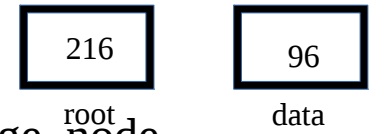
Deletetree 는
왼쪽방향의 가장 큰 자식노드를 찾아서
루트노드에 값을 넣어주고 그 자식노드를 삭제하고
그 자식노드의 자식을 올려주는 방식이다
Delete_tree



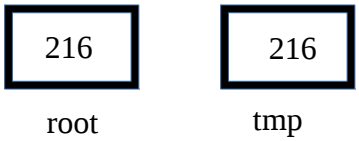
Find_max



Find_max



Change_node



After Delete
data = 48, left = 45, right = 73

data = 45, left = 32, right = 46
data = 32, left = 16, right = 37
data = 16, left = NULL,right = NULL
data = 37, left = NULL,right = NULL
data = 46, left = NULL,right = 47
data = 47, left = NULL,right = NULL
data = 73, left = NULL,right = 120
data = 120, left = NULL,right = 130
data = 130, left = 127, right = NULL
data = 127, left = 124, right = NULL
data = 124, left = NULL,right = NULL