Xilinx Zynq FPGA,TI DSP,
MCU 기반의
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

< 2 진 트리 _(재귀함수 없이 구현하기)>

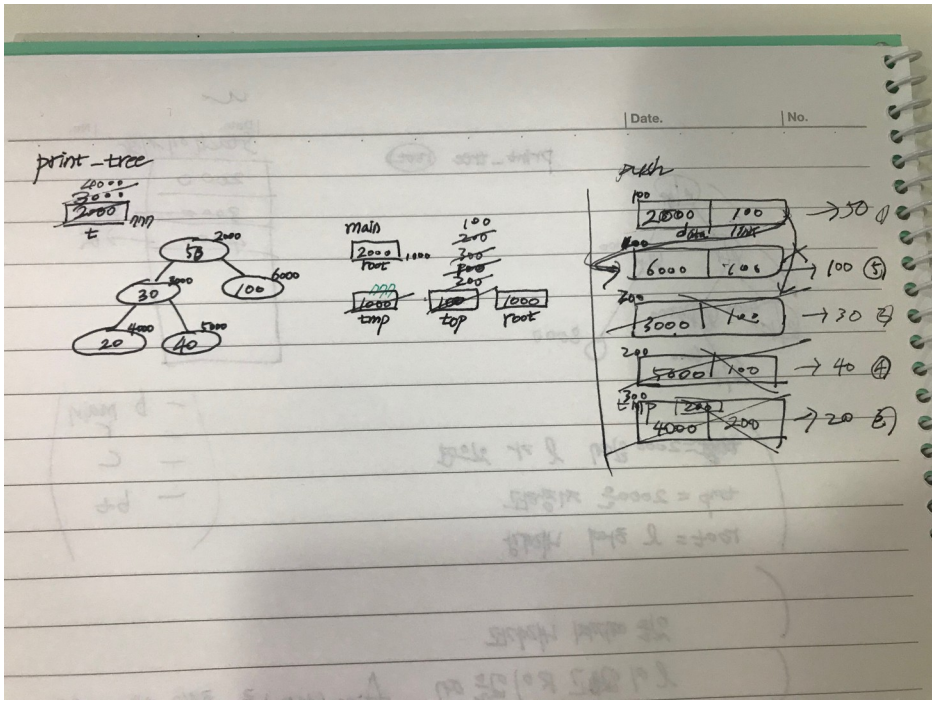
```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

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;
}
```



--재귀를 안쓰는 방법은 stack 을 사용하는 것이다.(Push, pop)
을 이용해야한다. (print 의 경우)

--재귀를 안쓰는 방법 두번째는 while 문을 사용하는 것이다.

<중요>

stack *get_stack_node(void)

라는 포인터 함수의 경우 인자로 보이드를 취하고 있다.

→ void *pop(stack **top)

데이터 타입에 보이드가 온 이유: 포인터 함수의 타입을 맞추기 위함이라고 보아도 된다.