TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

8회차 (2018-03-05)

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com

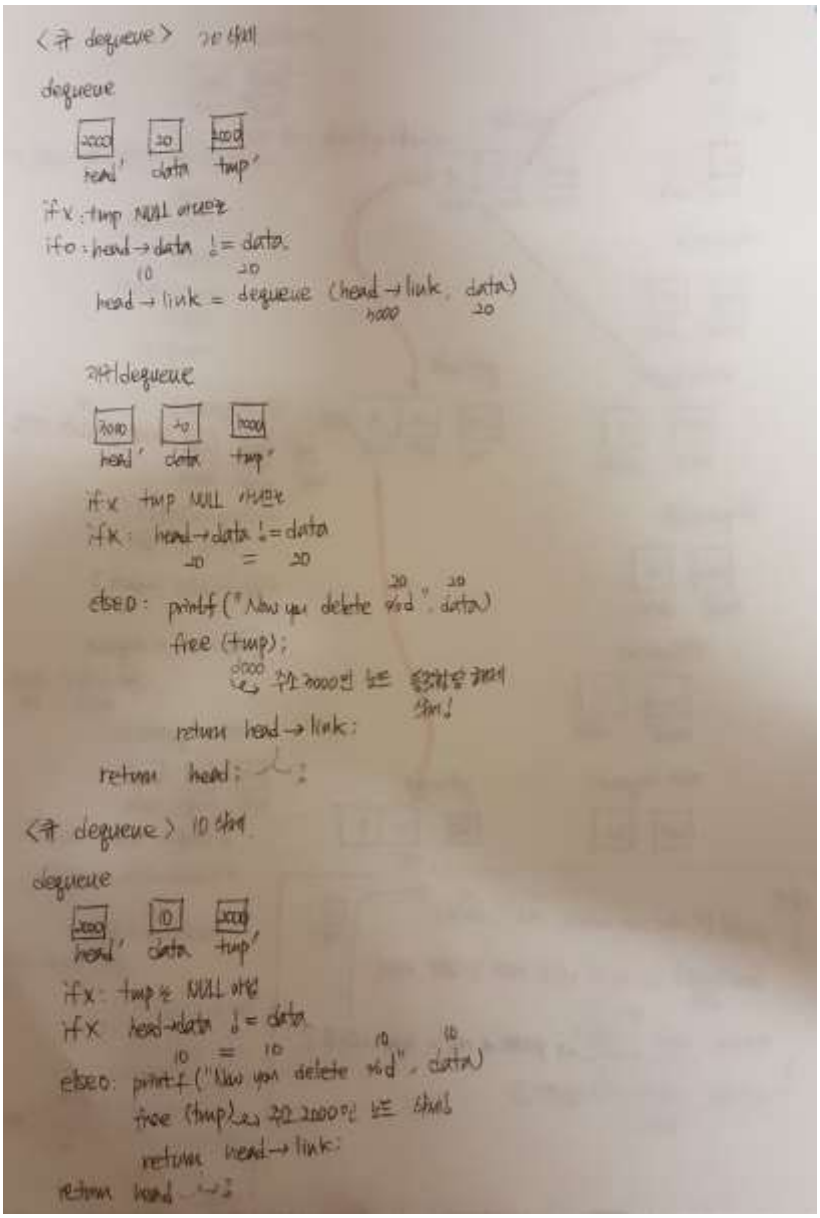
학생 - 정유경
ucong@naver.com

1. 큐 - dequeue 구현 및 그림해석

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <time.h>
#define EMPTY 0

struct node{
        int data;
        struct node * link;
};
typedef struct node queue;

queue* get_node()
{
        queue * tmp;
        tmp=(queue*) malloc(sizeof(queue));
        tmp->link = EMPTY;
        return tmp;
}

void print_queue(queue * head)
{
        queue * tmp = head;
        while(tmp)
        {
                printf("%d\n",tmp->data);
                tmp =tmp->link;
        }
        return ;
}
queue* dequeue(queue * head,int data)
```

```
{
        queue *tmp=head;
        if(tmp==NULL)
                printf("There is no data that you can delete\n");
        if(head->data != data)
                head->link = dequeue(head->link,data);
        else
        {
                // queue*res = head ->link;
                printf("Now you delete %d\n",data);
                free(tmp);
                return head -> link;
        }
        return head;
}

void Enqueue(queue** head, int data)
{

        if(*head==NULL)
        {
                *head=get_node();//head포인터가 새 노드를 가리킨다
                (*head)->data= data;
                return;
        }
        Enqueue(&(*head)->link,data);
        printf("재귀함수 호출!\n");
}
int main(void)
{
        int i;
        queue * head = NULL;
```

```
        srand(time(NULL));
        for(i=0;i<3;i++)
        Enqueue(&head, (i+1)*10);
        print_queue(head);
        head = dequeue(head,20);
        print_queue(head);
        return 0;
}
```



2. 트리를 사용하는 이유

스택형태로 데이터가 정렬되어 있다면 10억번째에 데이터가 있을 경우 10억번을 검색해야 한다.

하지만 이진트리를 사용하면 10억은 약 2의 31승 즉 31레벨까지만 검색하면 되므로

31번의 검색을 수행하면 된다.

즉, 경로설정을 빨리 해주기 위해 트리개념이 도입되었다

기본 트리의 경우 완벽하게 트리구성이 안되는 경우가 많다.

그래서 AVL트리가 만들어졌다.

4. 트리 자료구조 작성하기

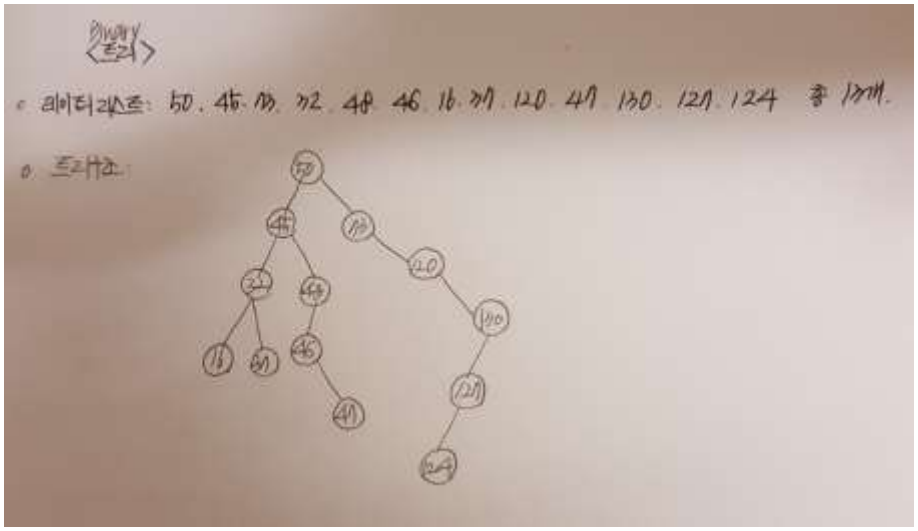
트리 자료구조에 사용할 데이터 리스트

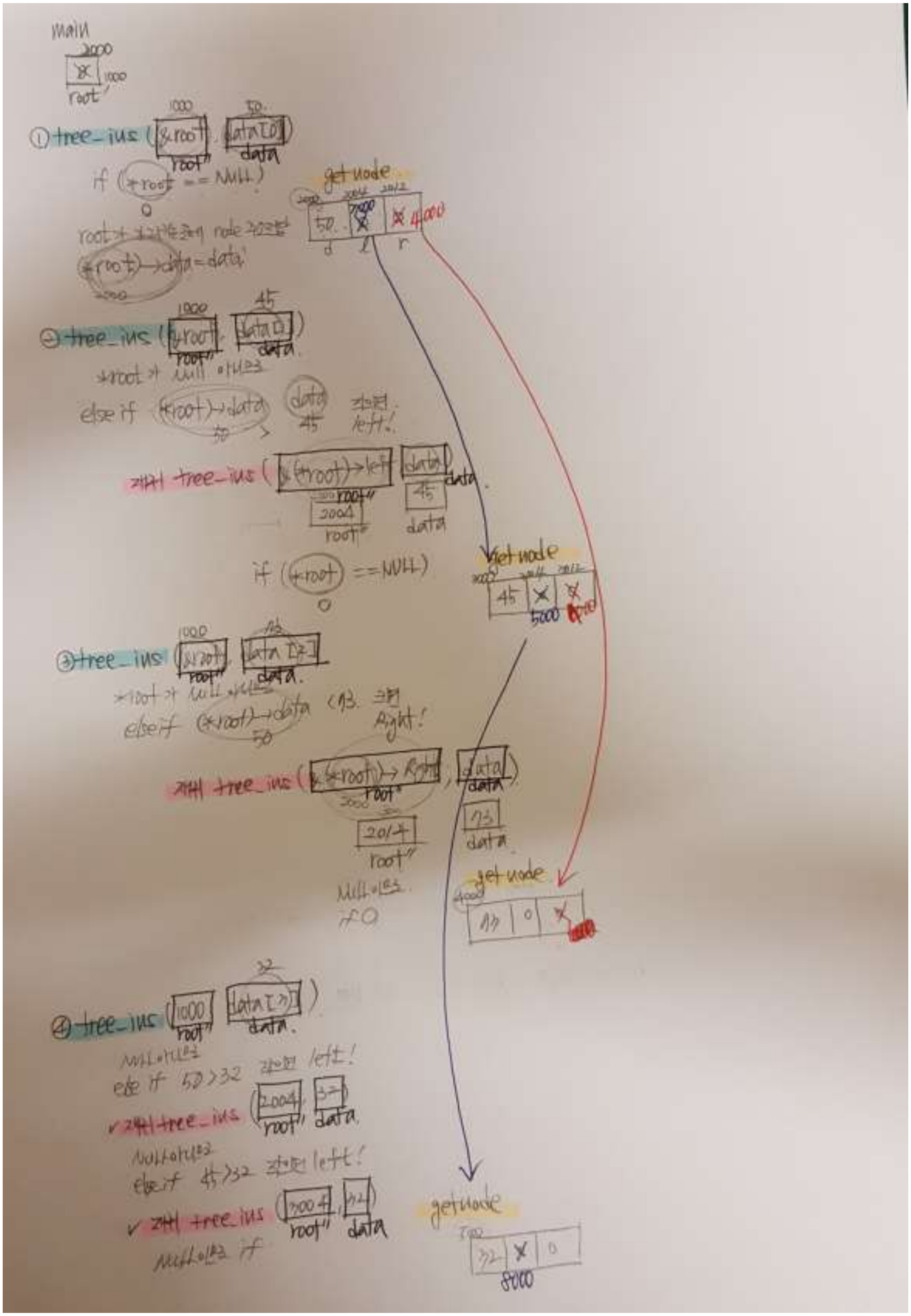
50 45 73 32 48 46 16 37 120 47 130 127 124

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

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

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

```
}

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

		//다음은 생략해도 된다
		if(root->left)
			printf("left=%d, ", root->left->data);
		else
			printf("left=NULL, ");

		if(root->right)
			printf("right=%d, ", root->right->data);
		else
			printf("right=NULL, ");
		//여기까지 생략해도 된다
		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)
{
        int i;
        int data[14] = {50,45,73,32,48,46,16,37,128,47,130,127,124};
        tree *root = NULL;
        for(i=0;data[i];i++)
                tree_ins(&root,data[i]);

        print_tree(root->left);
        print_tree(root->right);
        print_tree(root);
        delete_tree(root,50);
        printf("After Delete\n");
        print_tree(root);
        return 0;
}
```




main
root
① tree_ins (&root, data[0])
if (*root == NULL)
getnode
(*root)->data = data
② tree_ins
else if
left!
재귀 tree_ins
if (*root == NULL)
③ tree_ins
Right!
재귀 tree_ins
④ tree_ins
else if 50>32
left!
재귀 tree_ins
else if 45>32
재귀 tree_ins
root''
data

⑤ tree_ins (1000 root", data[4] data) 48

NULL이 아니므로
else if 50>48 left!

✓재귀tree_ins (2004 root", 48 data)

NULL이 아니므로
else if 45<48 right!

✓재귀tree_ins (3012 root", 48 data) getnode

NULL이므로
if

d	l	r
48	0	0

600

⑥ tree_ins (1000 root", data[5] data) 46

NULL이 아니므로
else if 50>46 left!

✓재귀tree_ins (2004 root", 46 data)

NULL이 아니므로
else if 45<46 Right!

✓재귀tree_ins (3012 root", 46 data) getnode

NULL이므로
if

d	l	r
46	0	0

1000

⑦ tree_ins (1000 root", data[6] data) 16

NULL이 아니므로
else if 50>16 left!

✓재귀tree_ins (2004 root", 16 data)

NULL이 아니므로
else if 45>16 left!

✓재귀tree_ins (3004 root", 16 data)

NULL이 아니므로
else if 32>16 left!

✓재귀tree_ins (5004 root", 16 data) getnode

NULL이므로 if

d	l	r
16	0	0

2000