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

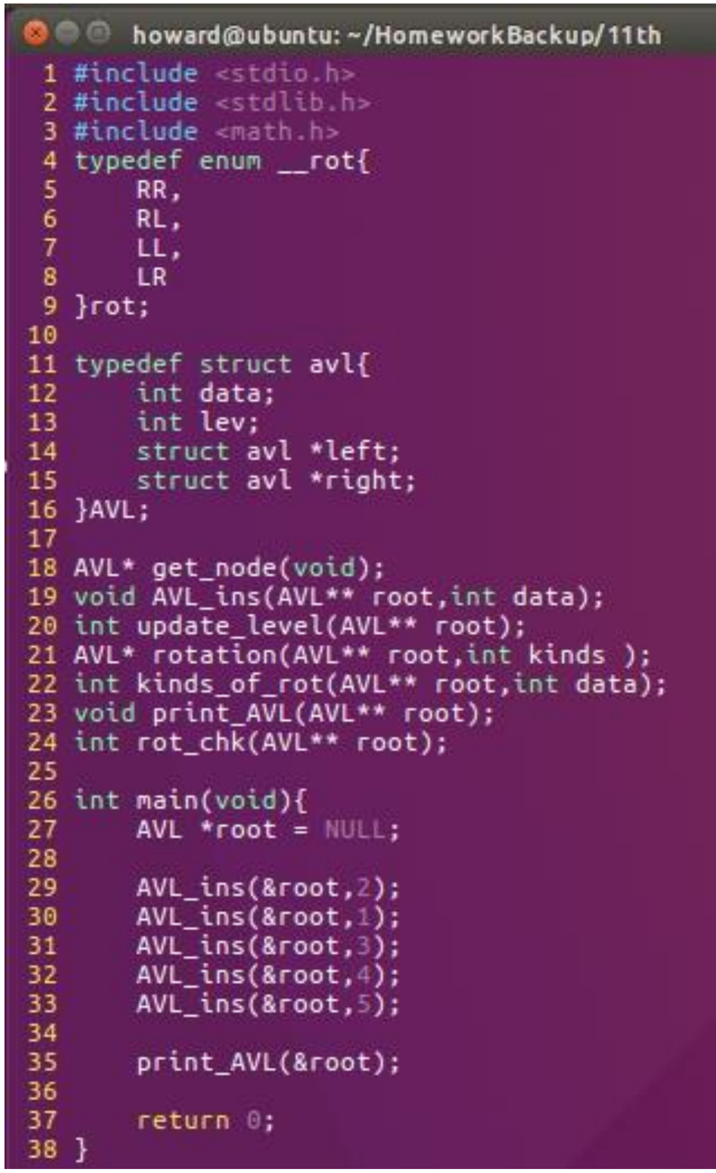
강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 김형주
mihaelkel@naver.com

What I leanred (18.03.08)

- Data Structure : AVL Tree

Data Structure : AVL Tree

- Consist of main(), get_node(), AVL_ins(), updata_level(), rotation(), kinds_of_rot(), print_AVL() rot_chk().

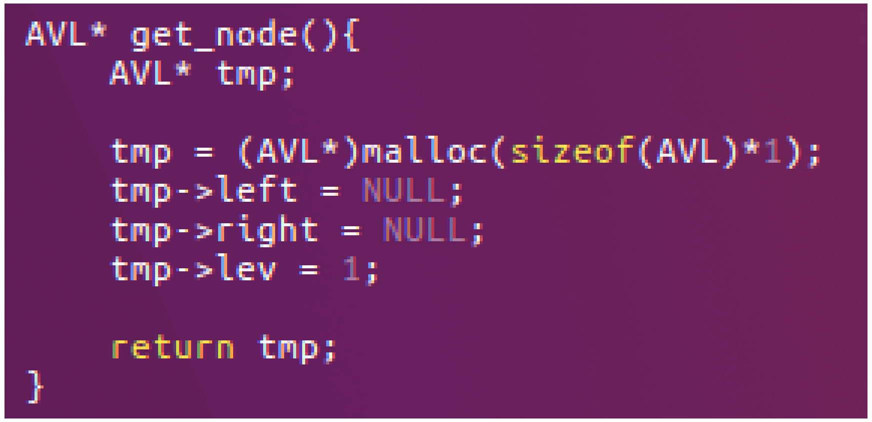


howard@ubuntu: ~/HomeworkBackup/11th

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef enum __rot{
        RR,
        RL,
        LL,
        LR
}rot;

typedef struct avl{
        int data;
        int lev;
        struct avl *left;
        struct avl *right;
}AVL;

AVL* get_node(void);
void AVL_ins(AVL** root,int data);
int update_level(AVL** root);
AVL* rotation(AVL** root,int kinds );
int kinds_of_rot(AVL** root,int data);
void print_AVL(AVL** root);
int rot_chk(AVL** root);

int main(void){
        AVL *root = NULL;

        AVL_ins(&root,2);
        AVL_ins(&root,1);
        AVL_ins(&root,3);
        AVL_ins(&root,4);
        AVL_ins(&root,5);

        print_AVL(&root);

        return 0;
}
```

Data Structure : AVL_Tree

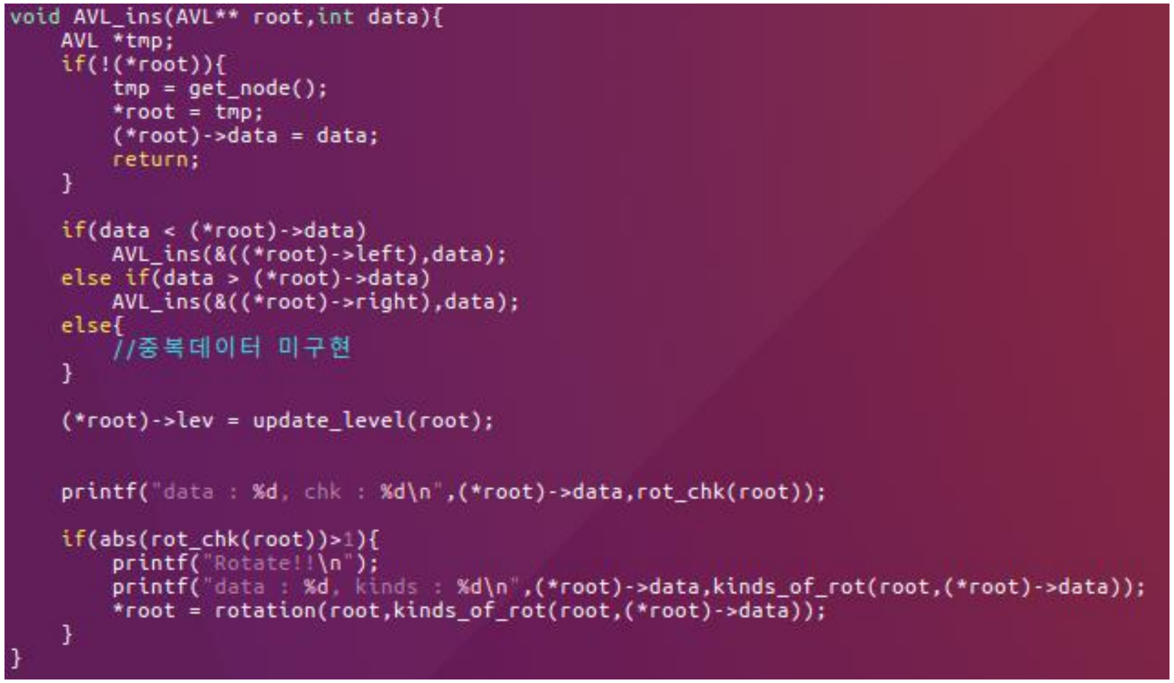
- AVL* get_node()
- This fuction returns Queue type pointer initialized left, right = NULL, lev = 1



```
AVL* get_node(){
    AVL* tmp;

    tmp = (AVL*)malloc(sizeof(AVL)*1);
    tmp->left = NULL;
    tmp->right = NULL;
    tmp->lev = 1;

    return tmp;
}
```

Data Structure : AVL_Tree

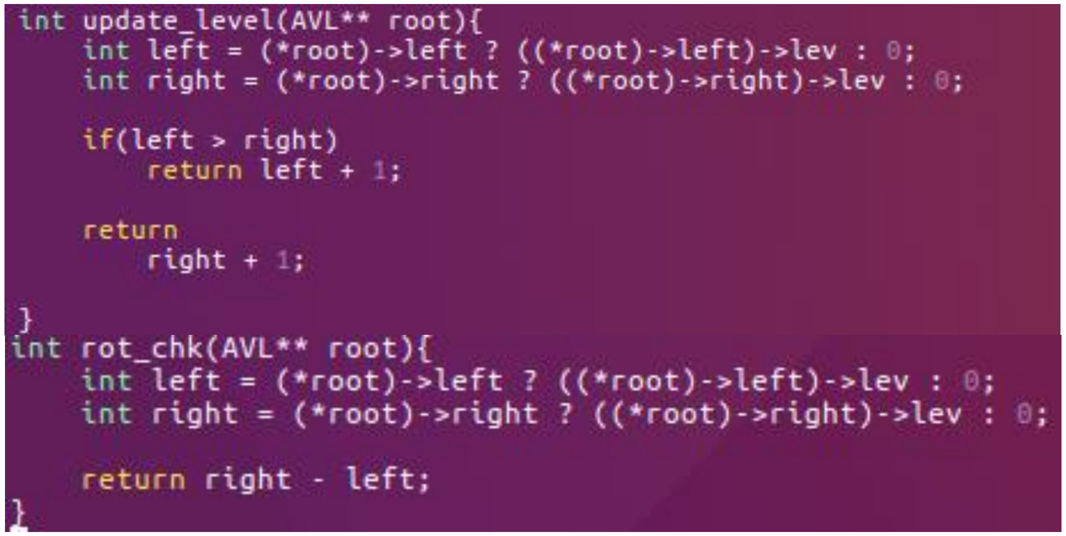
- Void AVL_ins(AVL** , int)
- This fuction basically works like a binary tree. But, when the balance is broken, it rotates.



```
void AVL_ins(AVL** root,int data){
    AVL *tmp;
    if(!(*root)){
        tmp = get_node();
        *root = tmp;
        (*root)->data = data;
        return;
    }

    if(data < (*root)->data)
        AVL_ins(&((*root)->left),data);
    else if(data > (*root)->data)
        AVL_ins(&((*root)->right),data);
    else{
        //중복데이터 미구현
    }

    (*root)->lev = update_level(root);


    printf("data : %d, chk : %d\n",(*root)->data,rot_chk(root));

    if(abs(rot_chk(root))>1){
        printf("Rotate!!\n");
        printf("data : %d, kinds : %d\n",(*root)->data,kinds_of_rot(root,(*root)->data));
        *root = rotation(root,kinds_of_rot(root,(*root)->data));
    }
}
```

Data Structure : AVL_Tree

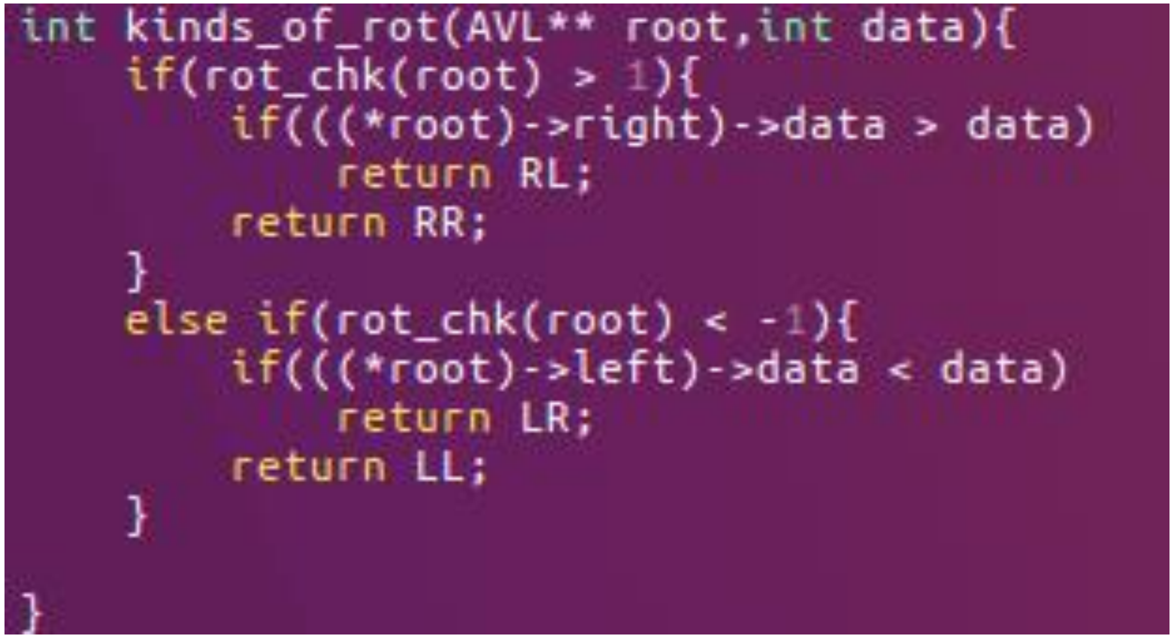
- Int update_level(AVL** root)
- This function returns node's level.
- Int rot_chk(AVL** root)
- This function returns children node's level difference.



```
int update_level(AVL** root){
    int left = (*root)->left ? ((*root)->left)->lev : 0;
    int right = (*root)->right ? ((*root)->right)->lev : 0;

    if(left > right)
        return left + 1;

    return
        right + 1;

}
int rot_chk(AVL** root){
    int left = (*root)->left ? ((*root)->left)->lev : 0;
    int right = (*root)->right ? ((*root)->right)->lev : 0;

    return right - left;
}
```

Data Structure : AVL_Tree

- Int kinds_of_rot(AVL** ,int)
- This fuction returns what kind of rotation should be operated.



```
int kinds_of_rot(AVL** root,int data){
    if(rot_chk(root) > 1){
        if(((*root)->right)->data > data)
            return RL;
        return RR;
    }
    else if(rot_chk(root) < -1){
        if(((*root)->left)->data < data)
            return LR;
        return LL;
    }

}
```