TI DSP,Xilinx zynq FPGA,MCU 및 Xilinx zynq FPGA 프로그래밍 전문가 과정

강사-INNOVA LEE(이상훈)

Gccompil3r@gmail.com

학생-윤지완

Yoonjw7894@naver.com

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


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

typedef struct stack
{
int data;
struct stack *link;
}stack;
typedef enum
{
RR,
RL,
LR,
LL
}rot;
typedef enum __rot
{
  RR,
  RL,
  LL,
  LR
} rot;
tree get_node()
{
tree *tmp;
tmp=(tree*)malloc(sizeof(tree));
tmp->level=1;
tmp->left=NULL;
tmp->right=NULL;
return tmp;
}
void insert(tree **root,int data)
{
int cnt=0;
tree **tmp=root;
stack *top = NULL;
while(*tmp)
{
printf("save stack:%d , data = %d₩n",cnt++,data);
push(&top,*tmp);
if((*tmp)->data>data)

tmp=&(*tmp)->left;
else if((*tmp)->data<data)
tmp=&(*tmp)->right;
}

*tmp=get_node();
(*tmp)->data=data;
while(stack_is_not_empty(top))
{
```

```
printf("exsist stack = %d,data = %d₩n",--cnt,data);
tree **t = (tree*)pop(&top);
(*t)->level = update_level (*t);
if(abs(rotation_check(*t))>1)
{
printf("insert rotation₩n");
int main(void)
{
tree *root = NULL;
int arr[14]= {50, 45, 73, 32, 48, 46, 16,
37, 120, 47, 130, 127, 124};
int i;
for(i=0;arr[i];i++)
insert(&root,data);
print_tree(&root,data);
return 0;
```

오늘은 avl tree를 재귀없애는 프로그래밍이다.
기본 트리는 avl기존에 것과 같지만 insert부분,스택을 이용해 push,pop을 통해서 데이터를 넣고 뺀다.일단 처음에 main에서 50에 값을 for문으로 배열에 값을 넣고 insert함수로 들어간다.
이때 맨 처음에 tree *root=null이 기때문에 while(*tmp)은 건너뛰게 된다. 그리고 데이터 50에 동적할당을 하고 data에 50을 넣는다. 그 다음 while(stack_is_not_empty(top))에서 지금 stack은 비어있기 때문에 건너뛰고 다시 main으로 돌아와서 4data= 45번을 가지고 insert 함수를 호풀하고 위와 동일하게 하고 이번에는 root의 값이 존재하고 그것을 **tmp가 가리키고 있기때문에 tmp는 값을 가지므로 while문이 실행이 된다. 그럼 printf문을 실행하고 push를 호출한다. 그럼

```
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;

}
```

push가 실행이 되고 data가 NULL이기 때문에 IF문은 건너뛰고 동적 메모리를 할당하고 TOP이 가리키는 data에 data를 넣고 포인터는 4바이트 의 크기를 가지고 있으므로 2000번지이므로
(*tmp)->link=tmp를 통해서 2004번지의 값을 tmp에 넣는다. 그 다음 insert로 다시와서 45는
if((*tmp)->data>data)

tmp=&(*tmp)→left; 이므로 값을 넣고 while문을 빠져나와서

```
while(stack_is_not_empty(top))
{
printf("exsist stack = %d,data = %d₩n",--cnt,data);
tree **t = (tree*)pop(&top);
(*t)->level = update_level (*t);
if(abs(rotation_check(*t))>1)
{
printf("insert rotation₩n");
```

다음 while문에 오고 이번에는 stack 공간이 값이 있으니 실행이 되고 *t에 &top에 주소 1000을 넣는다.t가 가리키고 있는 번지의 level을 update시키고 회전 되는지 확인하고 안되면 다시 빠져 나온다. 이순으로 프로그램이 진행되며 처음에 이중 포인터를 쓰는 이유를 아에 몰랐는데 설명듣고 완전히는 아니지만 어느정도 이해는 했습니다.