TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 윤연성
whatmatters@naver.com

구조체

```
struct point
{
	int x;
	int y;
}


int main(void)
{
	struct point p1, p2;			/구조체 point 의 변수 정의를 p1, p2 로 두개로 정의해도됨
}
```

구조체 배열

배열선언하는 것과 비슷함 앞에 struct 가 붙는것

- > 는 포인터를 이용하여 구조체의 멤버를 가르키는 것이다 p- > =number; = (*p).number 과 같다
포인터 p 가 가리키는 구조체의 멤버 number 이라는 뜻

(*p).number = p→number
p 가 가르키는 구조체의 멤버 number 을 의미

*p.number = 구조체 p 의 멤버 numbe 가 가리키는 것

*p→number = p 가 가르키는 구조체의 멤버 number 가 가리키는 내용

```
s.number	//
(*p).number	//  이 3 개는 같음
p→number	//
```

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

```
queue *dequeue *tmp = head;

if (tmp == NULL)
   printf("There are no data that you 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;
```

```
#include <stdio.h>
#include <stdlib.h>
int Tree_search(Tree *root, int data)
{
        if (root==NULL)
        {
        printf("데이터가 존재하지 않습니다.");
return 0;
}
        if((root)->data> data)
        Tree_search((root)->left_link , data);
        else if((root)->data== data)
return data;
        else
        Tree_search((root)->right_link , data);
}

struct node{
        Int data;
```

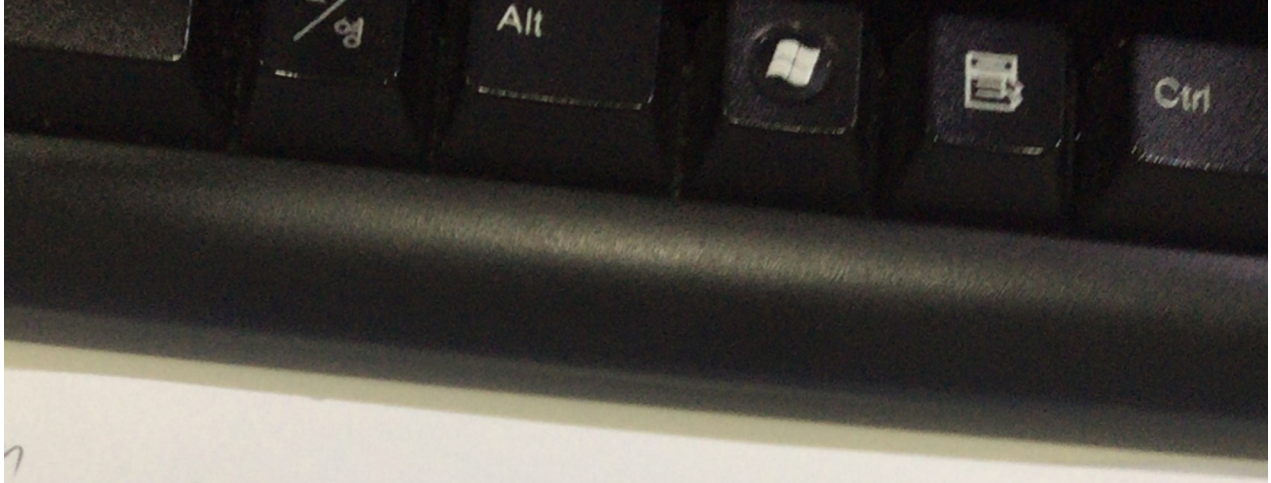
```
        struct node *left_link;
        struct node *right_link;
};
        typedef struct node Tree;
        Tree* get_node()
{
        Tree *tmp;
        tmp = (Tree *)malloc(sizeof(Tree));
        tmp->left_link =NULL;
        tmp->right_link=NULL;
return tmp;
}
        void Tree_ins(Tree** root, int data)
{
        if (*root==NULL)
{
        *root=get_node();
        (*root)->data=data;
return;
}
        if((*root)->data> data)
        Tree_ins(&(*root)->left_link, data);
        else
        Tree_ins(&(*root)->right_link, data);
}
void preorder(Tree *root)
{
        if(root)
        {
        printf("%d ", root->data);
        preorder(root->left_link);
        preorder(root->right_link);
        }
}
void inorder(Tree *root)
{
        if(root)
        {
        inorder(root->left_link);
        printf("%d ", root->data);
        inorder(root->right_link);
        }
}
void postorder(Tree *root)
{
        if(root)
{
        postorder(root->left_link);
        postorder(root->right_link);
        printf("%d ", root->data);
}
}
```

```
	Tree* node_change(Tree *root)
{
	Tree *tmp;
	tmp=root;

	if(!root->right_link)
	root=root->left_link;

	else if(!root->left_link)
	root=root->right_link;
	free(tmp);
	return root;
	}
	Tree * min_node(Tree *root, int *data)
	{
	if(root->left_link)
	root->left_link=min_node(root->left_link, data);

	else
	{
	*data = root->data;
	root=node_change(root);
	}
	return root;
	}
	Tree * delete_Tree(Tree *root, int data)
	{
	int num;
	Tree *tmp;
	}
	if(root == NULL){
	printf("not found");
return NULL;
}

	else if(root->data>data)
	root->left_link=delete_Tree(root->left_link, data);

	else if(root->data<data)
	root->right_link=delete_Tree(root->right_link, data);

	else if(root->left_link&&root->right_link)
	{
	root->right_link = min_node(root->right_link, &num);
	root->data = num;
	}
	else
	{
	root = node_change(root);
	}
	return root;int main(void)
{
```

```
        Int i;
        int test[20] = {45, 27, 17, 62, 57, 73, 52, 65, 76, 69, 63, 64, 71};
        Tree *root = NULL;
        for(i = 0; test[i] > 0; i++)
        Tree_ins(&root, test[i]);
        printf("기본 데이터 : ");

        for(i = 0; test[i]; i++)
        printf("%d ", test[i]);
        printf("\npreorder 데이터 : ");
        preorder(root);
        printf("\ninorder 데이터 : ");
        inorder(root);
        printf("\npostorder 데이터 : ");
        postorder(root);
        printf("\n");

        Tree_search(root, 65);
        delete_Tree(root, 27);
        delete_Tree(root, 45);
        delete_Tree(root, 17);
        delete_Tree(root, 71);
}
        printf(“\npreorder 데이터 : ");
        preorder(root);
        printf(“\ninorder 데이터 : ");
        inorder(root);
        printf("\npostorder 데이터 : ");
        postorder(root);
        printf("\n");

return 0;
}
```

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

struct node                // 구조체 노드
{
   int data;
   struct node *link;
};

typedef struct node Stack;              //기존의 struct node 를 Stack 로 정의

Stack *get_node()                       //  *get_node() 변수
{
   Stack tmp;                           //*tmp 정의
   tmp=(Stack *)malloc(sizeof(Stack));
   tmp→link=EMPTY;                              // tmp - > link   포인터 tmp 가 link 를 가르킴
   return tmp;                          //반환 tmp
}

void push(Stack **top, int data)
{
   Stack *tmp;
   tmp = *top;
   *top = get_node();
   (*top)->data = data;
   (*top)->link = tmp;
}

int pop(Stack **top)
{
   Stack *tmp;
   int num;
   tmp = *top;
   if(*top == 0)
   {
      printf("Stack is empty!\n");
      return 0;
   }

   num = tmp->data;
   *top = (*top)->link;
   free(tmp);

      return num;
}
int main(void)
{
   Stack *top = EMPTY;
   push(&top, 10);
```

```
    push(&top, 20);
    push(&top, 30);
    printf("%d\n", pop(&top));
    printf("%d\n", pop(&top));
    printf("%d\n", pop(&top));
    printf("%d\n", pop(&top));
    return 0;
}
```



, 45, 73, 32

1000

2000

50 | 3000 | 4000

data l r

2000 2002

3000

45

d l r

73

d r

_ ins

50

data

node tree_ins 5000

1000

r

73

d

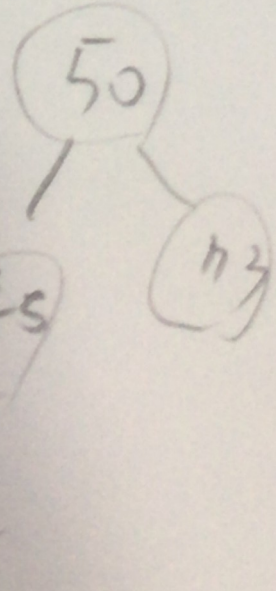
32

d l r

ns

45

d



ns