

**Xilinx Zynq FPGA, TI DSP, MCU 기반의  
프로그래밍 및 회로 설계 전문가 과정  
#6**

**강사 : Innova Lee(이 상훈)**

**학생 : 김 시윤**

## 1. 배운내용 복습

### 1. 구조체.

서로다른 타입의 변수를 한곳에서 사용하고싶을 때, 보통 int 형 변수는 int 형 함수에서만 사용이 가능하지만 구조체를 쓰면 여러 함수에서도 동작을 시킬 수 있어 유용하다. 형태에 제약을 덜 받는다고 생각하면 된다고 이해했다.

구조체는 보통 typedef 와 같이 쓴다 .

typedef 는 변수의 형태의 이름을 바꿔주는 방법으로

#define 은 변수의 이름을 바꾸지만 typedef 는 변수의 형태 int, char , double 등의 이름을 바꿔준다.

```
siyun@siyun-Z20NH-AS51BSU:~/my_proj/Homework/sanghoonlee/homework3$ cat structpoint.c
#include <stdio.h>

typedef enum __packet{
    ATTACK,
    DEFENCE,
    HOLD,
    STOP,
    SKILL,
    REBIRTH,
    DEATH = 44,
    KILL,
    ASSIST
}packet;

int main(void){
    packet packet;
    for(packet = ATTACK; packet <= REBIRTH; packet++)
        printf("enum num= %d\n", packet);
    for(packet = DEATH; packet <=ASSIST; packet++)
        printf("enum num = %d\n", packet);
    return 0;
}
```

```
siyun@siyun-Z20NH-AS51BSU:~/my_proj/Homework/sanghoonlee/homework3$ ./a.out
city = Seoul, name = Marth Kim, id = 800903-1012589 , age = 33
siyun@siyun-Z20NH-AS51BSU:~/my_proj/Homework/sanghoonlee/homework3$ cat structpoint.c
#include <stdio.h>
#include <stdlib.h>

typedef struct __id_card{
    char *name;
    char *id;
    unsigned int age;
}id_card;

typedef struct __city{
    id_card *card;
    char city[30];
}city;

int main(void)
{
    int i;
    city info = {NULL,"Seoul"};
    info.card = (id_card *)malloc(sizeof(id_card));

    info.card->name="Marth Kim";
    info.card->id = "800903-1012589";
    info.card->age = 33;

    printf("city = %s, name = %s, id = %s , age = %d\n",info.city,info.card->name,info.card->id,info.card->age);

    free(info.card);
    return 0;
}
```

struct를 이용한 실습예제.

## 2. 함수포인터.

```
#include <stdio.h>

// int (*) [2] aaa(void)
// return: int (*)[2]
// name: aaa
// parameter: void
// 배열 2개짜리 묶음의 주소를 반환하고 인자로 void를 취하는 함수 aaa
int (* aaa(void))[2]
{
    static int a[2][2] = {{10, 20}, {30, 40}}; //이 함수static함수가 이함수안에서만 전역변수가 됨
    printf("aaa called\n"); //int (*)p[2]
    return a;
}

// int (*)( bbb(void))(void)[2]
// case 1:
// 두개 있을때 함수 이름 부터 본다
// int (*)(*)(void)[2] bbb(void)
// int (*)[2] (*)(void) bbb(void)
// case2:
// int (*)( bbb(void))(void)[2]
// 맨끝이랑 맨끝을 붙인다
// int (*)[2](* bbb(void))(void)
// int (*)[2](*)(void) bbb(void)
// return: int (*)[2](*)(void)
// name: bbb
// parameter : void
// 배열 2개 짜리 묶음의 주소를 반환하고 인자로 void를 취하는 함수 포인터를 반환하며
// 인자로 void를 취하는 함수 bbb

// int (*) [2] aaa(void)
// int (*)[2] (*)(void) bbb(void)
// bbb 가 aaa를 리턴할수 있다.
int (* bbb(void))(void)[2]
{
    printf("bbb called\n");
    return aaa;
}

int (* ccc(void))(void)[2]
{
    printf("ccc called\n");
    return aaa;
}

int (* ddd(void))(void)[2]
{
    printf("ddd called\n");
    return aaa;
}

int (* eee(void))(void)[2]
{
    printf("eee called\n");
    return aaa;
}

int main(void)
{

```

```

    printf("bbb called\n");
    return aaa;
}

int (* ccc(void))(void)[2]
{
    printf("ccc called\n");
    return aaa;
}

int (* ddd(void))(void)[2]
{
    printf("ddd called\n");
    return aaa;
}

int (* eee(void))(void)[2]
{
    printf("eee called\n");
    return aaa;
}

int main(void)
{
    //rec 는 배열 2개 묶은걸 받을수 있는 포인터
    //int (*) [2] ret;
    int (*ret)[2];
    //int (*)[2](*)(void) bbb(void)
    //int (*)[2] (*)(void) (*)(void) p[][2]
    //bbb처럼 생긴 함수포인터 저장 비비비를 리턴하고 p를 인자로 취하는 함수 포인터
    //int (*)(p[][2])(void)(void)[2]
    //int (*)[2] (*)(void) bbb(void)
    //int (*)[2] (*)(void) (*)(void) p[][2]

    //배열 2개짜리 묶음의 주소를 반환하고
    //인자로 void를 취하는 함수 포인터를 반환하며
    //다시 인자로 void를 취하는 함수 포인터를 배열형태로 가짐

    int ((*(*p[][2])(void))(void))[2] = {{bbb, ccc}, {ddd, eee}};
    //int (*)[2] (*)(void) (*)(void) p[][2]
    //int (*)[2] (*)(void) (*)(void) [][2] p
    //int (*)[2] (*)(void) (*)(void) (*)[2] p1

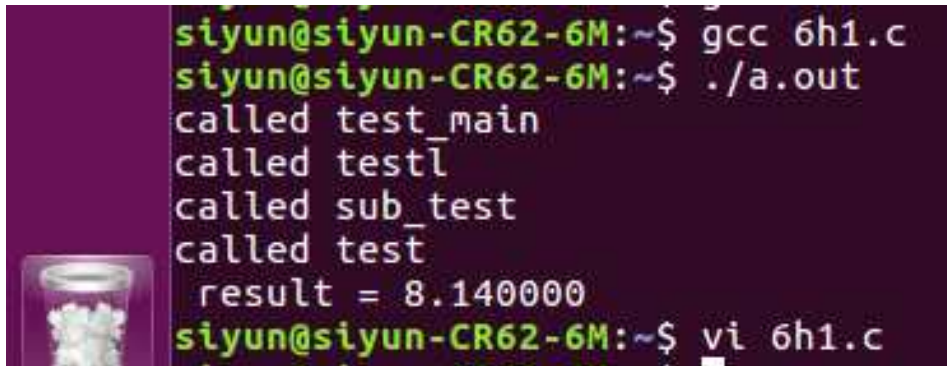
    int ((*(*p1[2])(void))(void))[2] = p;
    //int (*)[2] (*)(void) (*)(void) (*)[2] p1
    // ((*(p1[3]))());
    // (*(p1[3]))();
    // (*(p1[3]))();
    // (*) (*) (*) (*) p1[3] () ()
    // 각 별에 맞춰서 상쇄시켜라
    // = 3을 보라 eee와 같다
    // eee = aaa()

    ret = ((*(p1[3]))());
    //ret= address of array a-> a 배열의 주소
    printf("ret[0] = %d\n", *ret[0]); //10
    printf("ret[0][0] = %d\n", ret[0][0]); //10
    printf("ret[1] = %d\n", *ret[1]); //30
    printf("ret[1][1] = %d\n", ret[1][1]); //40
    return 0;
}
```

### 3. 과제 1.

`float (* (* test(void (*p)(void)))(float (*)(int, int)))(int, int)`

위와 같은 프로토타입의 함수가 구동되도록 프로그래밍 하시오.



```
siyun@siyun-CR62-6M:~$ gcc 6h1.c
siyun@siyun-CR62-6M:~$ ./a.out
called test_main
called testl
called sub_test
called test
result = 8.140000
siyun@siyun-CR62-6M:~$ vi 6h1.c
```

```
//float (*)(* test(void (*p)(void)))(float (*)(int,int))(int,int)
//float (*)(int,int)(* test(void (*p)(void)))(float (*)(int,int))
//float (*)(int,int)(*)(float (*)(int,int))test(void (*p)(void))

//리턴 float (*)(int,int)(*)(float (*)(int,int))
//name : test
//parameter : void (*p)(void)

//float (*)(int,int)(*)(float (*)(int,int))
//float (*)(float (*)(int,int))(int,int)
//float (*sub_test(float (*p)(int,int)))(int,int)
#include <stdio.h>

float (*sub_test(float (*test)(int,int)))(int,int)
{
    printf("called sub_test\n");
    return test;
}

//float (*sub_test(float (*p)(int,int)))(int,int)
//float (*p)(int,int)

float test(int n1,int n2)
{
    printf("called test\n");
    return (n1 + n2) + 3.14;
}

//void (*p)(void)
void testl(void)
{
    printf("called testl\n");
}

float (*(* test_main(void (*p)(void)))(float (*)(int,int)))(int,int)
{
    printf("called test_main\n");
    p();
    return sub_test;
}

int main(void)
{
    float result = test_main(testl)(test)(3,2);
    printf(" result = %f\n",result);
    return 0;
}
```

과제 2 Stack

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

#define EMPTY 0

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

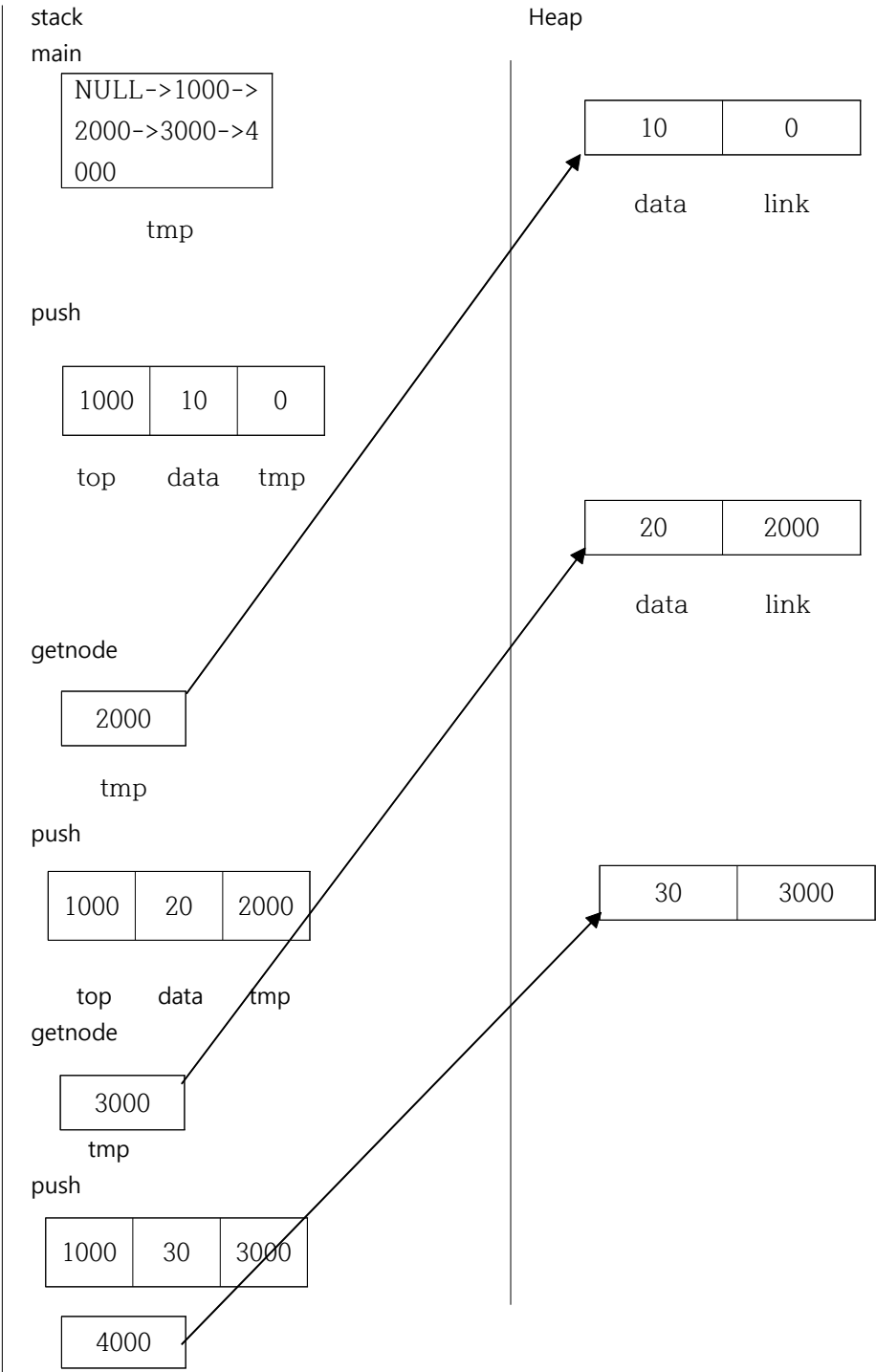
typedef struct node Stack;

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

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

void pop(Stack **top, int data)
{
    Stack *tmp;
    tmp = *top;
    *top = tmp->link;
    free(tmp);
    return data;
}

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;
}
```



여기까지는 직접 해보았는데 pop 가 그림으로 표현이 안됩니다.  
일단 알아본 동작과정을 설명하면

```
Stack = *tmp;
int num;
tmp=*top;
if(*top == EMPTY)

{
    printf("Stack is empty !!\n");
    return 0;
}
num = tmp->data;
*top= (*top)->link;
free(tmp);
return num;
}
```

top포인터를 tmp에 백업해두고 top가 가리키는 첫 노드의 값이 나중에 반납해야하기 때문에  
이전 top 데이터를 num 에다 복사해논다.  
그 복사해논 데이터로 확인하여 이프문을 통과하는거 같은데 자세히 모르겠습니다..