TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

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

```
struct student
{
I       nt number;
        char name[10];
        double grade;
};


int main(void)
{
        struct student s;                       //구조체 변수 s 선언
        printf("학번을 입력하시오: ");
        scanf("%d", &s.number);                         //s.number 주소값 전달
        printf("이름을 입력하시오: ");
        scanf("%s", s.name);                            //s.name 배열자체가 주소값
        printf("학점을 입력하시오(실수): ");
        scanf("%lf", &s.grade);
}
        printf("학번: %d\n", s.number);
        printf("이름: %s\n", s.name);
        printf("학점: %f\n", s.grade);

return 0;



-------------------------------

struct date                             //구조체선언
{
        int year;                               //멤버
        int month;                      //멤버
        int day;                //멤버
};
struct student
{
        int number;
        char name[10];
        struct date dob;                        //date 구조체 안에 dob 구조체 포함시킴
        double grade;
};
        struct
        student s1;                             // 구조체 변수 s1 선언
        s1.dob.year = 1983;             //s1 구조체 안에서 구조체변수 dob 안의 멤버 year
        s1.dob.month = 03;              //s1 구조체 안에서 구조체변수 dob 안의 멤버 month
        s1.dob.day = 29;                //s1 구조체 안에서 구조체변수 dob 안의 멤버 day
```

int (*pf)(int, int); int f(int, int) 의 함수를 가르키는 포인터

pf = &add 둘다 가능!
pf = add

int sub(int, int); // 함수 원형 정의
int (*pf)(int, int); // 함수 포인터 정의
...
pf = sub; // 함수의 이름을 함수 포인터에 대입
result = pf(10, 20); // 함수 포인터를 통하여 함수 호출

선언문은 선언 대상이 되는 변수 명에서 시작해서 오른쪽으로 가면서 해석한다.
선언문의 끝이나 오른쪽 괄호를 만나면 방향을 바꾸어 왼쪽으로 가면서 해석한다.
왼쪽으로 가면서 해석을 하다 왼쪽 괄호를 만나면 다시 오른쪽으로 가면서 해석한다.
int **a;
interpretation : a is a pointer to pointer to int
a 는 int 형을 가리키는 포인터의 포인터다.

int *a[3];
interpretation : a is an array of 3 pointer to int
a 는 int 형을 가리키는 포인터를 3 개 저장하는 배열이다.

int (*a)[3];
interpretation : a is a pointer to array of 3 int
a 는 int 형을 3 개 저장하고있는 배열을 가리키는 포인터다.

int (*p)(char);
interpretation : p is a pointer to function that takes a char as an argument and returns an int
p 는 char 를 인자로 갖고, int 형을 반환하는 함수에 대한 포인터다.

char** (*p)(float, float);
interpretation : p is a pointer to function that takes float, float as arguments and retruns a pointer to pointer to char
p 는 float, float 을 인자로 갖고, char 을 가리키는 포인터의 포인터를 반환하는 함수에 대한 포인터다.

void* (*a[5])(char* const, char* const);
interpretation : a is an array of 5 pointer to function that takes
char* const, char* const as arguments and returns pointer to void
a 는 char* const, char *const 를 인자로 갖고, void 를 가리키는 포인터를 리턴하는
함수에 대한 포인터를 5 개 저장하는 배열이다.

int* (*(fp1)(int))[10];
interpretation : fp1 is a function that takes an int as an arguments and returns a pointer to array of 10 pointers to int
fp1 은 int 를 인자로 갖고, int 를 가리키는 포인터를 10 개 저장하고 있는 배열의 포인터를 리턴하는 함수다.

int* (*(arr[5])())();
interpretation : arr is an array of 5 function that returns a pointer to function that returns a pointer to int
arr 는 int 를 가리키는 포인터를 리턴하는 함수에 대한 포인터를 리턴하는 함수 5 개를 저장하는 배열이다.

float (*(*b())[])();
interpretation : b is a pointer to function that returns a pointer to array of function that returns float
b 는 float 를 리턴하는 함수에 대한 배열을 저장하고있는 포인터를 리턴하는 함수에 대한 포인터다.void* (c)(char, int ()());
interpretation : c is a function that takes char, int ()() as arguments and returns pointer to void
c 는 char, int()()를 인자로 갖고, void 를 가리키는 포인터를 리턴하는 함수다.

void** (*d)(int &, char**(*) (char*, char**));
interpretation : d is a pointer to function that takes int &, char**(*)(char *, char *) as arguments and returns pointer to pointer to void
d 는 int &, char**(*)(char *, char *)를 인자로 갖고,
void 를 가리키는 포인터의 포인터를 리턴하는 함수에 대한 포인터다.

float (*(*e[10])(int &))[5];
interpretation : e is an array of 10 pointer to function that takes an int & as an argument and returns pointer to array of 5 float
e 는 int &를 인자로 갖고, float 을 5 개 저장하고있는 배열의 포인터를 리턴하는
함수에 대한 포인터를 10 개 저장하고 있는 배열이다.

int *(*(*fp1)(int))[10];
interpretation : fp1 is a pointer to function that takes an int as an argument and returns pointer to array of 10 pointer to int

fp1 은 int 를 인자로 갖고, int 를 가리키는 포인터 10 개를 저장하고있는
배열의 포인터를 리턴하는 함수에 대한 포인터다.

char *(*(**foo[][8])())[];
interpretation : foo is an array of array of 8 pointer to pointer to function returns pointer to array of pointer to char
foo 는 char 를 가리키는 포인터를 저장하고 있는 배열의 포인터를 리턴하는
함수에 대한 포인터의 포인터 8 개를 저장할 수 있는 배열에 대한 배열이다.

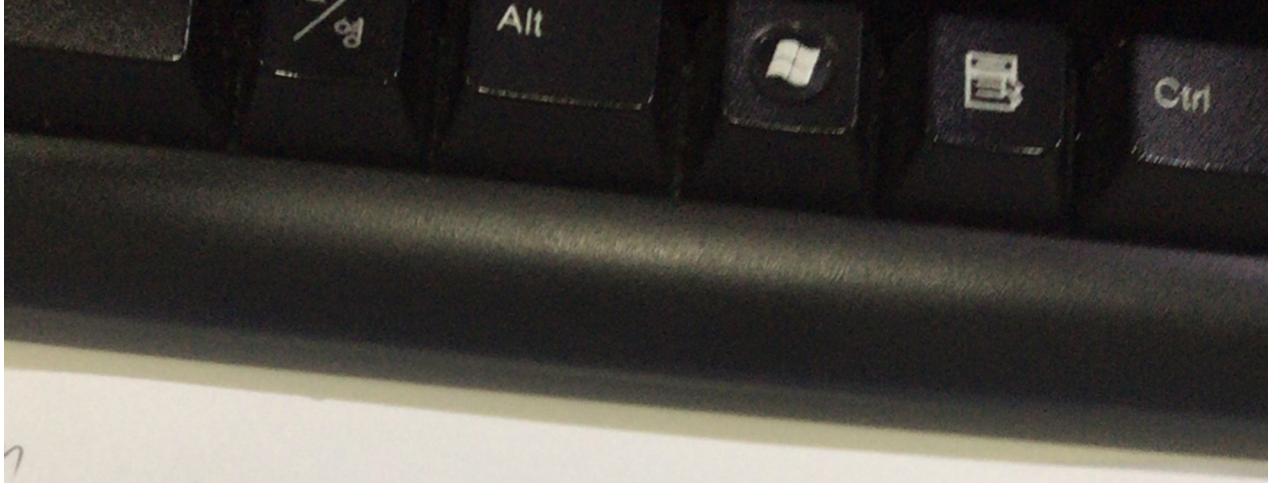
void bbb(void(*p)(void));
interpretation : bbb is a function that takes a (void(*p)(void) as an argument and returns void
bbb 는 인자로 (void(*p)(void)를 갖고, void 를 리턴하는 함수다.
void(*bbb(void))(void);
interpretation : bbb is function that takes a void as an argument and returns
pointer to function that takes a void as an argument and returns void
bbb 는 인자로 void 를 갖고, void 를 인자로 갖고 void 를 리턴하는 함수에 대한 포인터를 리턴하는 함수다.

void(*bbb(void(*p)(void)))(void);
interpretation : bbb is a function that takes a void(*p)(void) as an argument and
returns pointer to function that takes void as an argument and returns void
bbb 는 인자로 void(*p)(void)를 갖고, void 를 인자로 갖고
void 를 리턴하는 함수에 대한 포인터를 리턴하는 함수다.

int (*(*bbb(void))(void))[2];
interpretation : bbb is a function that takes a void as an argument and returns
pointer to function that takes a void as an argument and returns pointer to array of 2 int
bbb 는 인자로 void 를 갖고, void 를 인자로 갖고 int 를 2 개 저장하고있는 배열의 포인터를 리턴하는
함수에 대한 포인터를 리턴하는 함수다.

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

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

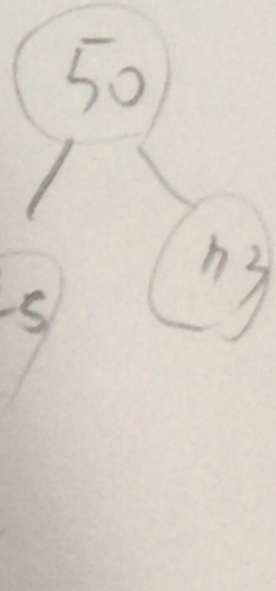
32

d l r

ns

45

d



ns