TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018.03.02
5 일차

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 신민철
akrn33@naver.com

--배열에 있는 값 거꾸로 집어넣어서 출력하기 ----

```
#include <stdio.h>
void rev_order(int *arr1, int *arr2, int size)
{
	int i, j;
	for(i = size - 1, j = 0; i >= 0; i--, j++)
	{
		arr2[j] = arr1[i];//arr2 첫번째에 arr1 끝을 넣음
	}
}
void print_arr(int *arr, int size)
{
	int i;
	for(i = 0; i < size; i++)
	{
		if(i != size - 1)//i 가 배열의 크기와 같지 않으면 실행
			printf("%d ", arr[i]);//arr i 번째 출력
		else
			printf("%d\n", arr[i]);
	}
}
int main(void)
{
	int arr[12] = {3, 77, 10, 7, 4, 9, 1, 8, 21, 33};
	int reverse_order[12] = {0};
	int size = (sizeof(arr) / sizeof(int)) - 1;
	print_arr(arr, size);
	rev_order(arr, reverse_order, size);
	print_arr(reverse_order, size);
	return 0;
}
```

--문자형배열 받아서 짝수값만 출력 ------

```
#include <stdio.h>

void print_even_arr_elem(char *str)
{
	int i;

	for(i = 0; str[i]; i++)
	{
		if(!(str[i] % 2))//str[i]가 2 로 나누어지면
		{
			printf("res = %d : %c\n", str[i], str[i]);
		}
	}
}
int main(void)
{
	char str[32] = "Hello Embedded World!\n";
	print_even_arr_elem(str);

	return 0;
}
```

---정수를 입력받아서 비트연산

```
#include <stdio.h>
void compute_int_bit(char *val, int size)
{
	int i, j;
	unsigned char comp = 0;
	printf("val = ");
	for(i = size - 1; i >= 0; i--)
	{
		comp = 128;
		for(j = 1; j <= 8; j++)
		{
			if(i == size - 1 && j == 1)
			{
				printf("%d ", val[i] & comp ? 1 : 0);
			}
			else
			{
				printf("%d", val[i] & comp ? 1 : 0);
			}
			comp >>= 1;
		}
		printf(" ");
	}
}

int main(void)
{
	int size = sizeof(int);
	int val = 2004016;

	compute_int_bit((char *)&val, size);

	return 0;
}
```

--정수를 난수로 입력받아서 저글링

```
#include <stdio.h>
void pointer_juggling(void)
{
	int n1 = 2, n2 = 4, n3 = 7;
	int *tmp = NULL;
	int *n_p[3] = {&n1, &n2, &n3};
	int **npp = NULL;
	int i = 0, j = 0;;
	printf("n1 = %d\n", *n_p[0]);
	printf("n2 = %d\n", *n_p[1]);
	printf("n3 = %d\n\n", *n_p[2]);

	for(;;)
	{
		npp = &n_p[i];
		tmp = *npp;
		*npp = n_p[i + 1];
		n_p[i + 1] = n_p[i + 2];
		n_p[i + 2] = tmp;

		printf("n1 = %d\n", *n_p[0]);
		printf("n2 = %d\n", *n_p[1]);
		printf("n3 = %d\n\n", *n_p[2]);

		if(j++ == 3)
			break;
	}
}

int main(void)
{
	pointer_juggling();
	return 0;
}
```

---------- 행렬의 덧셈 뺄셈 곱셈
1_6.c

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

void init_2x2_matrix(int (*mat)[2])
{
	int i, j;

	for(i = 0; i < 2; i++)
	{
		for(j = 0; j < 2; j++)
		{
			// 1 ~ 4
			mat[i][j] = rand() % 4 + 1;
		}
	}
}

void print_mat(int (* mat)[2])
{
	int i, j;

	printf("mat:\n");

	for(i = 0; i < 2; i++)
	{
		for(j = 0; j < 2; j++)
		{
			printf("%4d", mat[i][j]);
		}
		printf("\n");
	}
	printf("\n");
```

```
}

void add_2x2_matrix(int (*A)[2], int (*B)[2], int (*R)[2])
{
	int i, j;

	for(i = 0; i < 2; i++)
	{
		for(j = 0; j < 2; j++)
		{
			R[i][j] = A[i][j] + B[i][j];
		}
	}
}

void sub_2x2_matrix(int (*A)[2], int (*B)[2], int (*R)[2])
{
	int i, j;

	for(i = 0; i < 2; i++)
	{
		for(j = 0; j < 2; j++)
		{
			R[i][j] = A[i][j] - B[i][j];
		}
	}
}

void mult_2x2_matrix(int (*A)[2], int (*B)[2], int (*R)[2])
{
	/* a b   e f
	   c d   g h

	   ae + bg      af + bh
	   ce + dg      cf + dh */
```

```
	R[0][0] = A[0][0] * B[0][0] + A[0][1] * B[1][0];
	R[0][1] = A[0][0] * B[0][1] + A[0][1] * B[1][1];
	R[1][0] = A[1][0] * B[0][0] + A[1][1] * B[1][0];
	R[1][1] = A[1][0] * B[0][1] + A[1][1] * B[1][1];
}

int main(void)
{
	int mat1[2][2] = {0};
	int mat2[2][2] = {0};
	int res[2][2] = {0};

	// It's for use rand() function.
	srand(time(NULL));

	init_2x2_matrix(mat1);
	print_mat(mat1);

	init_2x2_matrix(mat2);
	print_mat(mat2);

	add_2x2_matrix(mat1, mat2, res);
	print_mat(res);

	sub_2x2_matrix(mat1, mat2, res);
	print_mat(res);

	mult_2x2_matrix(mat1, mat2, res);
	print_mat(res);

	return 0;
}
```

----------홀수번째 짝수번째 곱

```
#include <stdio.h>
int mult_even_odd_sum(int *arr, int size)
{
	int i, esum = 0, osum = 0;
	for(i = 0; i < size; i++)
	{
		if(i % 2)
		{
			osum += arr[i];
		}
		else
		{
			esum += arr[i];
		}
	}
	return osum * esum;
}
int main(void)
{
	int arr[12] = {3, 77, 10, 7, 4, 9, 1, 8, 21, 33};
	int size = sizeof(arr) / sizeof(int) - 1;
	int res;
	res = mult_even_odd_sum(arr, size);
	printf("res = %d\n", res);
	return 0;)
}
```

malloc() Function

malloc()은 무엇을 하는가?
Memory 구조상 heap 에 data 를 할당함
data 가 계속해서 들어올 경우
얼만큼의 data 가 들어오는지 알 수 없음
들어올 때마다 동적으로 할당할 필요성이 있음

free()은 무엇을 하는가?

Memory 구조상 heap 에 data 를 할당 해제함
malloc()의 반대 역할을 수행함
----------malloc

```
#include<stdio.h>

int main(void)
{
	char* str_ptr = (char * )malloc(sizeof(char) * 20);

	printf("Input String : ");
	scanf("%s",str_ptr);

	if(str_ptr != NULL)
		printf("string = %s \n",str_ptr);

	free(str_ptr);

	return 0;
}
```

```
------------------------------calloc
#include<stdio.h>
int main(void)
{
	int*  num_ptr = (int*)calloc(2,sizeof(int));

	printf("Input Integer:");
	scanf("%d%d",&num_ptr[0],&num_ptr[1]);

	if(num_ptr!=NULL)
		printf("Integer = %d, %d\n",num_ptr[0],num_ptr[1]);

	free(num_ptr);

	return 0;
}
--------------------------------typedef
#include<stdio.h>

	typedef int     INT;
	typedef int *   PINT;


int main(void)
{

	INT num = 3;
	PINT ptr = &num;

	printf("num = %d\n",*ptr);
	return 0;
}
```

----------typedef2

```
#include<stdio.h>

typedef int INT[5];

int main(void)
{
	int i;
	INT arr = {1, 2, 3, 4, 5};

	for(i = 0; i < 5; i++)
	{
		printf("arr[%d] = %d\n",i,arr[i]);
	}
	return 0;
}
```

구조체는 왜 사용하는가?

자료를 처리하다보니 하나로 묶어야 편함

문자열과, 숫자를 한 번에 묶어서 관리하고 싶을때 등

패스플래닝 자율주행차량 어떤경로로 가야하는가 계산할 때 데이터하ㄴ

어디에 연결될것인가 할 때 유용하게 쓰임

```
#include<stdio.h>

#define NAME_LEN	30
#define ID_LEN		15

typedef struct __id_card{
	char name[NAME_LEN];
```

```
	char id[ID_LEN];
	unsigned int age;
}id_card;

int main(void)
{
	int i;
	id_card arr[2] = {
	{"Marth Kim","800903-1012589",34},
	{"July Eun","830708-1023417",31}
	};
	for(i = 0; i < 2; i++)
	{
		printf("name = %s, id = %s, age = %d\n",
		arr[i].name,arr[i].id,arr[i].age);
	}
	return 0;
}
--------------------------------typedefstruct2
#include<stdio.h>

typedef struct __id_card{
	char name[30];
	char id[15];
	unsigned int age;
}id_card;

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

int main(void)
{
	int i;
```

```
	city info = {
	{"Marth Kim","800903-1012589",34},"Seoul"
	};
	for(i = 0; i < 2; i++)
	{
		printf("city = %s,name = %s, id = %s, age = %d\n",
		info.city,info.card.name,info.card.id,info.card.age);
	}
	return 0;
}
```

----------structtypedef3

```
#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;
}

-----------------------typedefstructchange
#include<stdio.h>
#include<stdlib.h>
typedef struct __data{
	int val;
	struct __data* data_ref;
}data;

int main(void)
{
	int i;
	data* data_p;
	data d1 = {3, NULL};
	data d2 = {7, NULL};

	d1.data_ref = &d2;
	d2.data_ref = &d1;

	data_p = &d1;
	for(i = 1; i <=10; i++)
	{
		printf("%3d",data_p->val);
		(data_p->val)++;
		data_p = data_p->data_ref;
		if(!(i%2))
```

```
		printf("\t");
	}
	return 0;
}
--------------------------------enumeration
#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;
}
```

----------functionpointer(함수포인터)==객체

```
#include<stdio.h>
void aaa(void){
	printf("aaa called\n");
}

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

int main(void){
	bbb(aaa);
	return 0;
}
```

---

void (*)(int) 반환이 없고 인자로 int 를 받는 함수포인터

void (*p)(void):
void 를 리턴하고 void 를 인자로 취하는
함수의 주소값을 저장할 수 있는 변수 p

```
int aaa(int, int);
int (*p)(int, int);
void(* signal(int signum, void(* handler)
```

----------클래스 객체에 있는 메소드를 사용하는것과 비슷한 코드.

```
#include <stdio.h>

typedef struct test_class
{
```

```
	int in1;
	int in2;
	double dn1;
	double dn2;

	int (*int_op)(int, int);
	double (*double_op)(double, double);
} tc;

int iadd(int n1, int n2)
{
	return n1 + n2;
}

int imul(int n1, int n2)
{
	return n1 * n2;
}

double dadd(double n1, double n2)
{
	return n1 + n2;:wq
}

int main(void)
{
	int res;
	double dres;
	tc tc_inst = {3, 7, 2.2, 7.7, NULL, NULL};

	tc_inst.int_op = iadd;
	res = tc_inst.int_op(tc_inst.in1, tc_inst.in2);
	printf("res = %d\n", res);

	tc_inst.int_op = imul;
```

```
	res = tc_inst.int_op(tc_inst.in1, tc_inst.in2);
	printf("res = %d\n", res);

	tc_inst.double_op = dadd;
	dres = tc_inst.double_op(tc_inst.dn1, tc_inst.dn2);
	printf("dres = %lf\n", dres);

	return 0;
}
```

----------puncpointer.c

```
#include<stdio.h>

void aaa(void){
	printf("aaa called\n");
}

int number(void){
	printf("number called\n");
	return 7;
}

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

void ccc(void(*p)(void)){
	printf("ccc: Ican call aaa!\n");
	p();
}
```

```
int (* ddd(void))(void){
	printf("ddd:Ican call number\n");
	return number;
}

int main(void){
	int res;
	bbb()();
	ccc(aaa);
	res = ddd()();
	printf("res = %d\n", res);
	return 0;
}
```

----------funcpointer2.c

```
#include<stdio.h>

void aaa(void){
	printf("aaa called\n");
}

void(*bbb(void(*p)(void)))(void){
	p();
	printf("bbb called\n");
	return aaa;
}

int main(void){
	bbb(aaa)();
	return 0;
}
```

```
//void(*bbb(void(*p)(void)))(void)          -->          void(*)(void)bbb(void(*p)(void)){

----------funcpointer3.c
#include<stdio.h>
int(* aaa(void))[2]{
        static int a[2][2] = {10, };
        printf("aaa called\n");
        return a;
}

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

int main(void){
        int (*ret)[2];
        int(*(*(*p[][2])(void))(void))[2] ={{bbb,bbb},{bbb,bbb}};
        int(*(*(*(*p1)[2])(void))(void))[2] = p;
        ret = ((*(*(*(*p1)[2])))()());
        printf("%d\n",*ret[0]);
        return 0;
}

----------q1.c/cafe.naver.com/hestit/788
#include<stdio.h>

float pof_test1(int n1, int n2)
{
        return (n1 + n2) * 0.23573;
}

int pof_test2(float n1, double n2, int n3)
{
```

```
	return (n1 + n2 + n3) / 3.0;
}

//int(*)(float, double, int) pof_test_main(float(*)(int,int))
//int(* pof_test_main(float (*)(int,int)))(float, double, int)

int (* pof_test_main(float (*p)(int,int)))(float, double, int)
{
	float res;
	res = p(4,3);
	printf("internal ptm res = %f\n", res);
	return pof_test2;
}

int main(void)
{
	int res;
	res = pof_test_main(pof_test1)(3.7, 2.4, 7);
	printf("pof_test_main res = %d\n",res);
	return 0;
}

----------------------------q2.c
#include<stdio.h>

//int(*)(int, int)
int pof1(int n1, int n2)
{
	return n1 + n2;
}

//int(*)(int, int) subpof1(int)
int (* subpof1(int n))(int, int)
{
	printf("n = %d\n",n);
```

```
	return pof1;
}

//float (*)(int, double)
float pof2(int n1, double n2)
{
	return n1 * n2;
}

//int(*)(int, int) (*)(int) pof_test_main(float (*)(int, double))
//int(*)(int, int) (* pof_test_main(float (*)(int,double)))(int)

//int(*(* pof_test_main(float (*)(int, double)))(int))(int,int)
//int(*)(int, int) (* pof_test_main(float (*)(int, double)))(int)
//int(*)(int, int) (*)(int) pof_test_main(float (*)(int, double))

int(*(* pof_test_main(float (*p)(int, double)))(int))(int,int)
{
	float res;
	res = p(3,7.7);
	printf("res = %f\n",res);
	return subpof1;
}

//int p (int, int(int))
int main(void)
{
	int res;
	res = pof_test_main(pof2)(3)(7,3);
	printf("res = %d\n",res);
	return 0;
}
```

```
-------------------------func_p2.c
#include <stdio.h>

// int (*)[2] aaa(void)
// return: int(*)[2]
// name: aaa
// parameter: void
int (* aaa(void))[2]
{
	static int a[2][2] = {{10, 20}, {30, 40}};
	printf("aaa called\n");
	return a;
}

// int (*(* bbb(void))(void))[2]

// case 1: criteria: function name
// int (*(*)(void))[2] bbb(void)
// int (*)[2](*)(void) bbb(void)

// case 2: first '(*' last ')~~'
// int (*)[2](* bbb(void))(void)
// int (*)[2](*)(void) bbb(void)

// return: int(*)[2](*)(void)
// name: bbb
// parameter: void

// int (*)[2] aaa(void)
// int (*)[2] (*)(void) bbb(void)
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)
{
	// int (*)[2] ret;
	int (*ret)[2];
	// int (*)[2] (*)(void) bbb(void)
	// int (*)[2] (*)(void) (*)(void) p[][2]
	int (*(*(*p[][2])(void))(void))[2] = {{bbb, ccc}, {ddd, eee}};
	// 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] ()() = eee()()
	// eee()() = aaa()
	// ret = address of array a
	ret = ((*(*(*(*p1)[3])))()());
```

```
	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;
}
```

----------memcpy.c

```
//되도록이면 memmove 를 쓰는것이 해킹에 안정적임
#include<stdio.h>
#include<string.h>

int main(void)
{
	char src[30]= "This is amazing";
	char *dst = src + 3;

	printf("before memmove = %s\n",src);
	memcpy(dst, src, 3);
	printf("after memmove = %s\n", dst);

	return 0;
}
```

----------memmove.c

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

int main(void)
{
	int i = 0;
	int src[5] = {1, 2, 3, 4, 5};
	int dst[5];

	memmove(dst, src, sizeof(src));
```

```
	for(i = 0; i < 5 ; i++)
	printf("dst[%d] = %d\n",i,dst[i]);

	return 0;

}
----------------------marg.c
#include<stdio.h>

int main( int argc, char **argv, char **env)
{
	int i;

	printf("argc = %d\n", argc);

	for(i = 0; i < argc; i++)
	{
		printf("argc[%d] = %s\n", i, argv[i]);
	}

	for(i = 0; env[i]; i++)
		printf("env[%d] = %s\n", i, env[i]);

	return 0;
}

------------------------strlen.c
#include<stdio.h>
#include<string.h>

int main(int argc, char** argv)
{
	char *str = "This is the string";
	int len = strlen(str);
	printf("len = %d\n",len);
```

```
	return 0;

}
```

----------strncmp.c

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

int main(int argc, char** argv)
{
	char src[20] = "made in korea";
	char dst[20] = "made in china";
	if(!strncmp(src, dst, 8))
	printf("src, dst 는 서로 같음 \n");
	else
	printf("src, dst 는 서로 다름 \n");
	return 0;
}
```

----------strncpy.c

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

int main(int argc, char** argv)
{
	char src[20] = "abcdef";
	char dst[20];
	strncpy(dst, src, 3);
	printf("dst = %s\n",dst);
	return 0;
}
```

----------stack.c

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

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

```
{
        printf("Stack is empty!!!\n");
        return 0;
}
```