Day5

- 과정이름: TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

1. 피보나치 재귀함수 풀어쓴 예제 fib.c

2. 팩토리얼 재귀함수 예제 factorial.c

3. 팩토리얼 재귀함수 풀어쓴 예제 factorial2.c

4. 배열이 필요한 이유

- 배열의 정의
- 배열
  DataType 배열명[길이] = {0};
  2 차원배열 → 이중배열
  3 차원배열 → 삼중배열

(예제) array.c

```
#include <stdio.h>

int main(void)
{
	int i;
	int num[7];

	for(i =0; i < 7; i++)
	{
		num[i] = i+1;
		printf("num[%d] = %d₩n", i, num[i]);
	}

	return 0;
}
```

(예제) array2.c

```
#include <stdio.h>

int main(void)
{
	int i;
	int num1_arr[] = {1, 2, 3, 4, 5};
	int num2_arr[3] = {1, 2, 3};

	int len1 = sizeof(num1_arr)/sizeof(int);
	int len2 = sizeof(num2_arr)/sizeof(int);

	printf("num1_arr length = %d₩n", len1);
```

```
	printf("num2_arr length = %d₩n", len2);

	for(i = 0; i < len1; i++)
	{
		printf("num1_arr[%d] = %d₩n", i, num1_arr[i]);
	}

	for(i=0; i < len2; i++)
	{
		printf("num2_arr[%d] = %d₩n", i, num2_arr[i]);
	}

	return 0;
}
```

(예제) array3.c

```
#include <stdio.h>

int main(void)
{
	int i;
	int num1_arr[7] = {1, 2, 3};

	for(i=0; i < 7; i++)
	{
		printf("num1_arr[%d] = %d₩n", i, num1_arr[i]);
	}

	return 0;
}
```

5. char 배열이 필요한 이유

- 문자열의 1byte문자 조작을 위해 필요
- 마지막 data에 Null Character가 필요
- Null Character는 '₩0'으로 문자열의 끝을 표시

```
#include <stdio.h>

int main(void)
{
	char str1[5] = "AAA";
	char str2[] = "BBB";
	char str3[] = {'A', 'B', 'C'};
	char str4[] = {'A', 'B', 'C', '₩0'};

	printf("str1 = %s₩n", str1);
	printf("str1 = %s₩n", str2);
	printf("str1 = %s₩n", str3);
```

```
	printf("str1 = %s₩n", str4);

	str1[0] = 'E';
	str1[1] = 'H';
	printf("str1 = %s₩n", str1);

	return 0;
}
```

6. 다중(고차원)배열을 사용하는 이유

- c언어에서 배열에 차원이라는 개념은 존재하지 않음
- 2 차원 배열의 경우 행렬 표현에 용이
- 2 차원 배열의 경우 [x][y]로 x명의 y개 과목을 관리

```
#include <stdio.h>

int main(void)
{
	int arr[4][4];
	int i, j;

	for(i=0; i < 4; i++)
	{
		for(j=0; j < 4; j++)
		{
			if(i == j)
				arr[i][j] = 1;
			else
				arr[i][j] = 0;
		}
	}


	for(i=0; i < 4; i++)
	{
		for(j=0; j < 4; j++)
		{
			printf("%d", arr[i][j]);
		}

		printf("₩n");
	}

	return 0;

}
```

- 메모리에서 배열의 상태

[0]	[0]
	[1]
	[2]
	[3]
[1]	[0]
	[1]
	[2]
	[3]
[2]	[0]
	[1]
	[2]
	[3]
[3]	[0]
	[1]
	[2]
	[3]

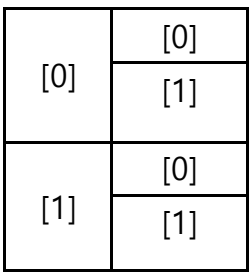
- 고차원 배열

```
int arr[2][2][3] = {
                        {{1, 2, 3}, {1, 2, 3}},
                        {{1, 2, 3}, {1, 2, 3}},
                    };
```

[0]	[0]	[0]
		[1]
		[2]
	[1]	[0]
		[1]
		[2]
[1]	[0]	[0]
		[1]
		[2]
	[1]	[0]
		[1]
		[2]

(예제) multidimArray.c

```
#include <stdio.h>

int main(void)
{
	int arr[2][2] = {{10, 20}, {30, 40}};
	int i, j;

	for(i=0; i < 2; i++)
	{

		for(j=0; j < 2; j++)
		{

			printf("arr[%d][%d] = %d₩n", i, j, arr[i][j]);
		}
	}


	return 0;

}
```



- 배열의 내부
  배열의 대표 주소는 배열의 이름
  배열 = 주소

- 컴퓨터의 주소
  가상메모리, 페이징(paging) 확인

- 2 차원 배열과 1 차원 배열의 관계
  2 차원 배열은 1 차원 배열의 묶음

```
#include <stdio.h>

int main(void)
{
	int arr[3][4];

	printf("arr address = %lu₩n", arr);
```

```
	printf("arr[0] address = %lu₩n", arr[0]);
	printf("arr[1] address = %lu₩n", arr[1]);
	printf("arr[2] address = %lu₩n", arr[2]);

	return 0;
}
```

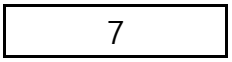
(예제) pointer2.c

```
void add_arr(int *arr)
{
	int i;

	for(i = 0; i < 3; i++)
	{
		arr[i] += 7;
	}


}

void print_arr(int *arr)
{
	int i;
	int end;
	end = sizeof(arr)/sizeof(int);

	for(i=0; i<= end; i++)
	{
		printf("arr[%d] = %d₩n", i, arr[i]);
	}
}

int main(void)
{
	int i;
	int arr[3] = {1, 2, 3};
	add_arr(arr);
	print_arr(arr);

	printf("real₩n");

	for(i=0; i < 3; i++)
	{
		printf("arr[%d] = %d₩n", i, arr[i]);
	}

	return 0;
```

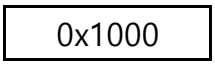
```
}
```

- 포인터의 개념
  주소를 저장하는 메모리 공간(변수)
  포인터의 크기는 시스템지원 최대크기 byte로 설정

- 포인터의 선언
  int num = 7;
  int *pointer1 = #
  (int형의 데이터를 가리키는 포인터 변수)



num(0x1000)



pointer1

(예제) pointerError.c

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

int main(void)
{
	int *ptr = (int *)malloc(sizeof(int)*10);	//malloc하지 않으면 segmentation Error 발생

	printf("ptr = %p₩n", ptr);
	printf("ptr value = %d₩n", *ptr);

	*ptr = 27;

	printf("ptr value = %d₩n", *ptr);

	return 0;
}
```

- Segmentation fault가 나는 이유

. 가상메모리주소(user 사용주소) ↔ 메모리주소 변환(paging 메커니즘) ↔ 물리주소

. 가상메모리는 리눅스의 경우 32 비트 버전과 64 비트 버전으로 나뉨

. 32 비트 시스템 - 4GB => 1:3 = 커널:유저
커널은 HW, CPU, SW와 같은 정보관리, 유저영역은 변수와 같은 시스템에 크게 영향주지 않는내용으로 구성

. 64 비트 시스템 - 8GB => 1:1 = 커널 : 유저
포인터 메모리 할당 하지 않는 경우 쓰레기값 0xCCCCC...CCC를 생성
32 비트 메모리영역 1:3 경계 0xc0000000 을 넘어가는 주소를 침범
64 비트 경우 1:1 경계 0x77...77 을 넘어가는 주소를 침범

.접근하면 안되는 메모리 영역 접근 => page fault발생 => 인터럽트 발생 => page handler발생(페이지 제어기)=> 가상메모리에 대한 paging 처리 => 물리메모리 할당 => user쪽에서 들어온 요청 kernel에서 거절 => Segmentation Fault발생

. 커널에서 들어온 요청일 경우 물리 메모리 할당

64bit
2^63 kernel 영역
2^63 user

32bit
1G kernel
3G user

(예제) null 포인터 지정

```
#include <stdio.h>

int main(void)
{
	int *ptr = NULL;

	printf("ptr = %p₩n", ptr);
	printf("ptr value = %d₩n", *ptr);

	return 0;
}
```

(예제) 포인터를 통한 기존 변수값 변경(pointer3.c)

```
#include <stdio.h>

int main(void)
{
	int num = 3;

	*(&num) += 30;

	printf("num = %d₩n", num);

	return 0;
}
```

(예제) 배열과 포인터에 문자열 할당(pointer4.c)

```
#include <stdio.h>

int main(void)
{
	char str1[33] = "Pointer is important!";
	char *str2 = "Pointer is important!";

	printf("str1 = %s₩n", str1);
	printf("str2 = %s₩n", str2);

	return 0;
}
```

- Pointer에 대한 Pointer

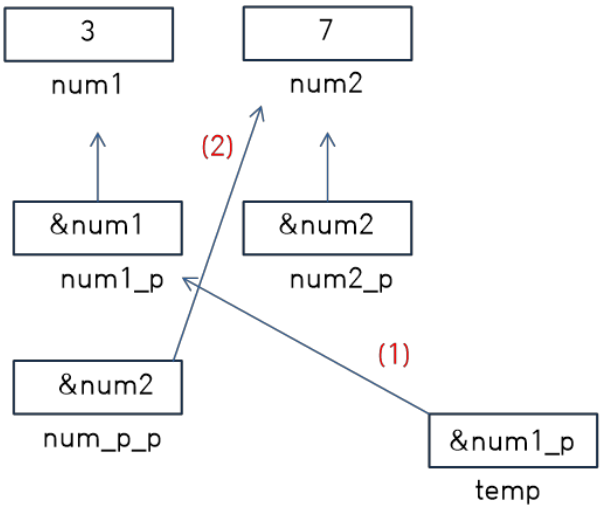
(예제) doublePointer.c

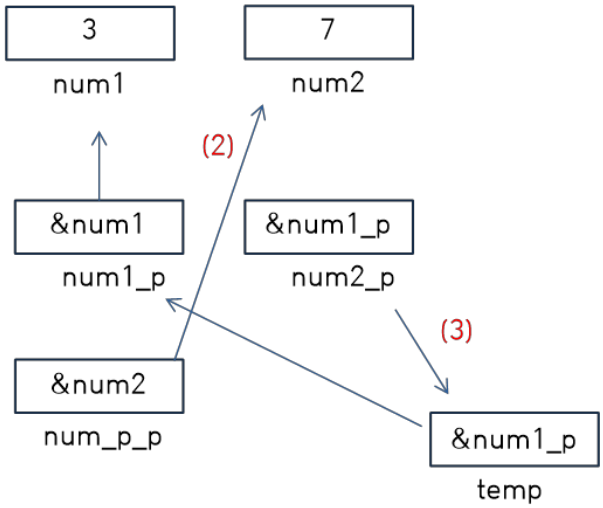
```
 #include <stdio.h>

int main(void)
{
	int num = 3;
	int *p = &num;
	int **pp = &p;

	printf("num = %d₩n", num);
	printf("*p = %d₩n", *p);
	printf("**pp = %d₩n", **pp);

	return 0;
}
```

(예제) pointer 주소를 이용한 swap(doublePoinet2.c)

```
#include <stdio.h>

int main(void)
{
	int num1 = 3, num2 = 7;
	int *temp = NULL;
	int *num1_p = &num1;
	int *num2_p = &num2;
	int **num_p_p = &num1_p;

	printf("*num1_p = %d₩n", *num1_p);
	printf("*num2_p = %d₩n", *num2_p);

	temp = *num_p_p;		-------(1)
	*num_p_p = num2_p;		------(2)
	num2_p = temp;			-------(3)

	printf("num1_p = %d₩n", *num1_p);
	printf("num2_p = %d₩n", *num2_p);

	return 0;
}
```

< 관계도 >





(예제) 배열포인터 예제 (arrayPointer.c)

```
#include <stdio.h>

int main(void)
{
	int i, j, n1, n2, n3;
	int a[2][2] = {{10, 20}, {30, 40}};
	int *arr_ptr[3] = {&n1, &n2, &n3};
	int (*p)[2] = a;

	for(i=0; i < 3; i++)
	{
		*arr_ptr[i] = i;
	}

	for(i=0; i <3; i++)
	{
		printf("n%d = %d₩n", i, *arr_ptr[i]);
	}

	for(i=0; i <2; i++)
	{
		printf("p[%d] = %d₩n", i, *p[i]);
	}

	return 0;
}
```

*숙제
1. 배운내용 복습
2. 문제은행
cafe.naver.com/hestit/79/
2, 5, 7, 8 제외

cafe.naver.com/hestit/104
2, 4 번 풀이

*별도의 문제
삼각형의 넓이 구하는 문제
case1) 밑변, 높이
case2) 밑변, 밑변과 다른 변이 이루는 각도

2 가지 경우로 모두 구현해본다.
(삼각함수 복습)
* 제공된 교재의 279 페이지 퀴즈를 풀어보기
2, 4 번 제외