1장

++, --, &, \*등은 우선순위는 모두 같다. 하지만 right to left의 결합성에 따라서…

\*(imsip++) == \*imsip++

2장

Int imsi[3]

\*(imsi + i); // 잘 수행

\*imsi++; // 컴파일 에러 - imsi라는 것은 배열명으로써 imsi[0]이 저장된 곳의 주소를 뜻하고 있을 뿐 변수가 아니기 때문에 imsi라는 곳에는 주소를 저장할 수 없다.

imsi = imsip; // imsi = imsi + 1; // imsi = &imsi[0]; - 같은 이유로 컴파일 에러

3장

Int \*\*imsip; // imsip는 1차원 포인터 배열에 대한 변수

Int (\*imsip2)[3] // 2차원 배열 포인터

Int (\*imsip3)[2][3] // 3차원 배열 포인터

대상체가 무엇인지 잘 판단하자.

Int imsi[3][2];

// 24 bytes, imsi가 가리키는 대상체는 imsi 배열 전체

// imsi[0]은 0번째 행 전체

// imsi[0][0]

Int (\*imsip)[2]; // 2차원 배열에 대한 포인터 변수, 변수가 하나 할당

// imsip라는 변수는 2차원 배열을 가리키고 있으며 메모리에 4바이트 딱 하나가 생성된다.

// 2차원 배열 포인터 변수는 행을 대상체로 한다.

// \*imsip + 1은 첫번째행의 1번째 배열 요소 : 주소 + 정수 => 주소

Int \*temp[2]; // 포인터를 가리킬 수 있는 요소 두 개를 가지고 있는 “포인터 배열”

// 이것은 포인터 변수가 두 개나 생성되고 메모리에도 당연히 두 변수에 대한 할당이 이루어진다. Temp[0] / temp[1]

// 주소를 저장할 수 있는 공간이 하나가 아닌 두 개가 만들어진다.

// temp[0]/temp[1] 은 1차원 배열만 받을 수 있다.

// temp자체는 무엇일까? Temp[0]/temp[1]을 대표하는 모 배열… 크기는 8

Sizeof(\*imsip) -> 한 행을 가리키고 있으므로 8

I

msip = imsi; // Okay

Temp = imsi; // 컴파일 에러

2차원 배열을 포인터로 접근했을 때 별표 하나는 행을 대상체로 하고 있기 때문에 절대 배열 요소 자체의 값을 취할 수 없다.

모든 주소의 sizeof는 항상 4이다.

15장

2차원 배열을 함수에 넘길 때 어떻게 선언하느냐?

Void view(int \*\*temp); // Wrong

// 포인터의 포인터는 포인터 배열을 다루기 위해서 사용되는 것이지 2차원 배열을 다루기 위해서는 사용되는 것이 아니다.

Int (\*imsip)[3];

Void view(int (\*)[3]);