Q1. 이중 포인터

변수	표현법	변수	표현법
&num	ptr1, *ptr2	&ptr2	-
num	*ptr1, **ptr2	ptr2	&ptr1
&ptr1	ptr2	*ptr2	&num, ptr1
ptr1	&num, *ptr2	**ptr2	num, *ptr1
*ptr1	num, **ptr2		

정수형 변수 num에 값 333을 저장한 후, num의 주소를 정수형 포인터 ptr1에 저장한다. 또, ptr1의 주소를 ptr2에 저장한다. ptr1이 가리키는 주소에 있는 값이라는 의미로 *ptr1을 사용하여 num에 접근할 수 있으며, 한 수준 더 들어가 **ptr2로도 접근할 수 있다. 또한 ptr1의 값, 즉 num의 주소는 *ptr2로 접근할 수 있다.

Q2. 삼중 포인터

변수	표현법	변수	표현법	변수	표현법
&num	ptr1, *ptr2, **ptr3	&ptr2	ptr3	ptr3	&ptr2
num	*ptr1, **ptr2, ***ptr3	ptr2	&ptr1, *ptr3	*ptr3	&ptr1, ptr2
&ptr1	ptr2, *ptr3	*ptr2	&num, ptr1, **ptr3	**ptr3	&num, ptr1, *ptr2
ptr1	&num, *ptr2, **ptr3	**ptr2	num, *ptr1, ***ptr3	***ptr3	num, *ptr1, **ptr2
*ptr1	num, **ptr2, ***ptr3	&ptr3	-		

Q1과 동일하나 한 수준이 더 추가되어 ptr2의 주소를 저장하는 삼중 포인터 ptr3이 존재한다. 간접 참조 연산자 세 개를 사용하여 ***ptr3로 세 수준 아래 값 num에, **ptr3로 ptr1에 접근할 수 있으며 *ptr3로 ptr2에 접근이 가능하다.

Q3. 이중 포인터 및 포인터 배열

변수	표현법	변수	표현법
&num1	arr[0], ptr[0]	num1	*arr[0], *ptr[0]
&num2	arr[1], ptr[1]	num2	*arr[1], *ptr[1]
&num3	arr[2], ptr[2]	num3	*arr[2], *ptr[2]

num1, 2, 3의 값을 각각 지정하고, 각각의 주소를 포인터 배열 arr에 저장하여 초기화한다. 또한, 이 배열의 주소를 이중 포인터 ptr2에 저장한다. 배열의 첫 번째 인덱스의 주소를 나타내는 배열의 이름 arr은, 배열의 주소를 저장한 ptr과 같은 값을 가진다. 기존 배열의 사용과 같이 arr[index]로 배열의 각 요소에 접근할 수 있으며, 이는 변수의 주소이므로 간접 참조를 통하여 *arr[index]로 원래 변수의 값에 접근할 수 있다. 또한, arr과 ptr은 같은 값을 가리키므로 ptr[index]로도 동일하게 배열 요소에 접근할 수 있다.

Q4. 포인터 배열

문자열 포인터 배열 arr은 힙에 저장되는 문자열 상수 “Alpha”, “Beta” … “Epsilon”의 주소를 배열 요소로 저장한다. 따라서 arr[index]로 각 요소에 접근할 수 있으며, 이는 각 요소가 가리키는 문자열의 첫 번째 주소이므로, 이를 포맷 형식 %s로 출력하면 해당 문자열이 출력된다.

Q5. 함수 포인터

함수 포인터의 선언은 함수의 반환형 지정, 포인터 연산자 및 함수 포인터 이름 지정, 괄호 안에 매개변수 자료형 지정으로 이루어진다. 함수 포인터 선언 시 지정한 반환형과 매개변수의 개수, 자료형은 함수 포인터에 저장할 함수와 일치해야 한다. 함수의 이름 또한 배열과 마찬가지로 함수의 메모리 주소 값을 나타내므로, 함수 포인터에 함수의 주소를 저장할 때는 배열과 같은 방식으로 사용한다. 포인터에 의한 함수 호출 시 원래 함수를 호출하는 것과 같이 함수 포인터로 호출하여 사용할 수 있다.

Q6. 함수 포인터(2)

func에서 매개변수로 함수 포인터를 받고, 연산할 숫자 두 개를 추가로 입력받아 함수 포인터로 호출한 함수에 매개변수로 전달한다. main에서는 입력 모드에 따라 func에 서로 다른 함수의 주소를 전달하여 각 요청에 맞는 함수를 호출한다.

Q7. exchangeNum에서 void형 포인터 2개와 자료형 지정 문자열을 인자로 받아, 자료형 문자열을 통해 void형 포인터에 저장된 변수 타입을 확인한 후 각 자료형에 적합한 매크로 함수를 호출한다. 포인터는 그 자료형에 무관하게 메모리 값을 저장하므로, 항상 4바이트로 일정한 크기를 가진다. 따라서 자료형을 알지 못하는 변수의 주소라도 void형으로 저장할 수 있다. 그러나 자료형을 알 수 없으므로 이를 역참조하여 수행하는 연산은 일체 불가능하다.