**C프로그래밍 11차시**

**포인터**

&변수명 : 변수의 주소 [**참조연산자(&) -** 앰퍼샌드(ampersand)**]**

\*변수명 : 포인터 변수가 가리키는 주소의 내용

Ex) a = 3

&a = 020 (a가 3을 저장하고 있는 주솟값의 위치)

P = 020 (포인터 변수 P가 020이라는 주솟값을 저장하고 있음.)

\*p = 3 (주솟값 020 위치에 존재하는 값인 3을 가짐.)

%p (형식지정자) : 주솟값을 출력할 때 사용하는 형식지정자.

포인터 변수는 값을 1 더할때마다 주솟값이 더해짐.  
예를 들어, int 변수의 경우, 후술하겠지만 4byte이고, 따라서 int \*a = Null; 로 정의된 변수가 있을 때,   
00000000 -> 00000004의 주솟값을 가지게 됨.

Double의 경우, 00000008의 주솟값을 가질 것이고, char의 경우 00000001의 값을 가질 것이다.

**포인터 정의법**

[타입] \* [포인터의 이름] ;

항상 정의 시에 Null로 초기화해주는 것이 좋음. Ex) Int \*a = null;

1. Int \*p << 이 정의 방식이 가장 좋음.
2. Int \* p
3. Int\* p

**포인터 변수의 크기는 왜 일정한가?**

Char 1byte

Int 4byte

Short 2 byte

Long 4 byte

Long long 8 byte

Float 4 byte

Double 8 byte

포인터 변수 4 byte (32bit 기준.) 포인터 변수는 어떤 값이 들어가든 주솟값을 가지기에 4바이트로 일정하다**.(32비트 기준, 64비트라면 8바이트.)**

**포인터 배열 접근**

포인터 배열은, int arr[3] = {1, 2, 3}; int \*p = arr;으로 정의된 경우,

arr[0]의 주소가 020이면,  
arr[1]의 주소는 024, arr[2]의 주소는 028, arr[3]의 주소는 032 …. 으로, 자료형의 바이트 수 만큼 증가한다.  
이 순서의 배열은, 포인터 변수의 경우 +1을 한 것과 동일하므로, 포인터 변수 p를 이용해,  
arr[1]은 p+1, arr[2]는 p+2 등으로 생각하여 나타낼 수 있다.

그러므로, p와 arr이 같은 의미를 가지기에 a[n] == p[n]이라는 것을 알 수 있다.

**포인터 함수 응용**

Int main() {

Int a = 5;  
sub(&a);  
} //메인함수에서 sub함수 실행.

void sub(int \*a) {  
\*a = 7;  
} //sub함수. 포인터 변수를 지역변수인 매개변수로 선언해, a변수 주소에 접근하여 수정함.

처럼 사용하여 함수 내에서 주솟값을 받아와 직접 값에 접근해 수정하거나 어떠한 작업을 처리할 수 있다.