10주차 C프로그래밍 정리노트

10주차 C프로그래밍 정리노트 8조 김다은, 유시연

포인터는 변수로, 메모리의 **위치**를 가리킨다.

포인터 선언 방법

    자료형 * 변수명;

    int * ptr;

!. 포인터 변수의 크기는 4바이트이기 때문에 char *나 double *로 선언한다.

예제 10 - 1

예제를 작성하면서 ptr의 값과 *ptr의 값이 다르게 나옴을 알 수 있었다.

ptr 은 i의 주소값이 나오지만, *ptr의 경우엔 i의 값인 10이 나온다.

예제 10 - 2

포인터에 대해 잘 이해하고 있는지 확인할 수 있었다.

*ptr을 사용하여 바로 직접적으로 i의 값을 바꿀 수 있다는 걸 알게되었다.

!.Swap 함수 : 두 변수의 값을 서로 교환하는 함수.

예제 10 -3

기존에 나온 예시대로 Swap() 함수를 사용하면 값이 서로 바뀌지 않는다. 바꾸기 위해선 어떻게 해야할까?

A. 지금은 int a와 int b만 서로 교환하는 것으로 되어있지만 이것을 그들의 주소값 &a &b를 변경하는 것으로 바꾸어봤다. 그렇게 했더니 문제가 바라던대로 서로의 값이 바뀐 채 나왔다.

A2. 주소값으로 변경해준 이유는 단순히 앞에서 배웠던 것과 연관이 있을거라 생각했기에 바꿨던 것인데, 주소값을 변경하는 것이 맞는 이유에 대해 생각해봐야겠다.

A3. 주소값으로 변경해야만 하는 이유는 포인터의 특성처럼 주소값의 내용을 바꿔야 온전한 값이 바뀌는 것이 아닌가 싶다.

A4. 만약 주소값으로 변경하지 않으면 a와 b의 값들은 stack에 쌓인다고 한다. 그렇게 되면 return하면서 값이 모두 사라져 최종적으로 값의 변경이 없다고 한다.

즉, ptr과 * ptr이 다른 것처럼 주소를 넘겨줘야만 return으로 stack의 내용들이 사라진다고 해도 그 주소의 내용을 바꿀 수가 있다고 한다.

call by value와 call by reference

예제 10-3에서 값이 바뀌지 않았던 이유가 바로 call by value형식이었기 때문이다.

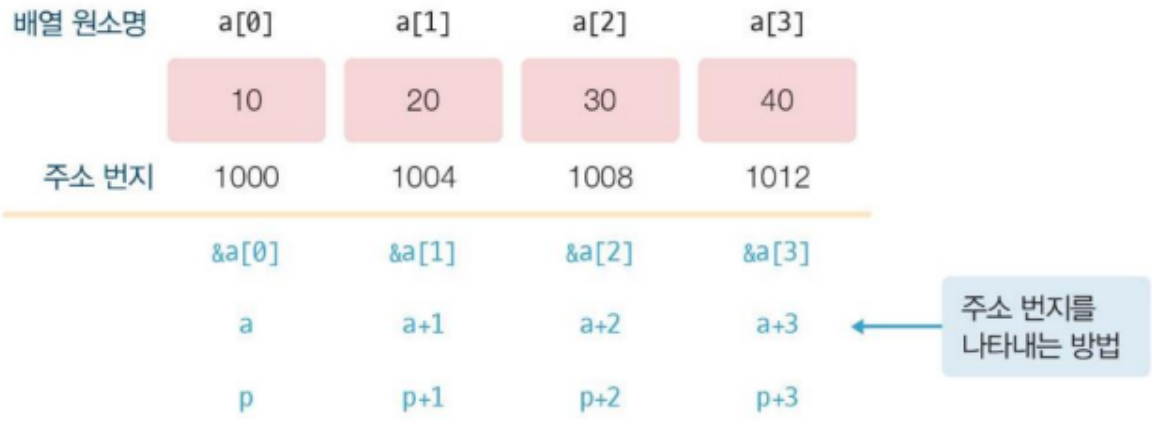
값을 스택에 복사하는 것이기 때문에 인자 값을 바꾸더라고 main()함수는 영향을 받지 않는다.

반대로 call by reference는 포인터를 이용하여 함수에 변수 자체를 전달할 때 사용한다. 즉, 변수의 주소값을 직접적으로 건드리는 것이기에 변수 자체의 값을 변경하게 되는 것이다.

예제 10-5

예제 10-5를 통해 call by value와 call by reference의 차이를 재차 확인하여 알 수 있었다.

포인터와 배열의 대응 관계



■ 배열의 주소를 표현하는 방법

표 10-1 배열 a의 주소를 표현하는 방법

구분	첫 번째 주소	두 번째 주소	세 번째 주소	네 번째 주소
방법 1	&a[0]	&a[1]	&a[2]	&a[3]
방법 2	a	a+1	a+2	a+3
방법 3	p	p+1	p+2	p+3

■ 배열의 값을 표현하는 방법

표 10-2 배열 a의 값를 표현하는 방법

구분	첫 번째 값	두 번째 값	세 번째 값	네 번째 값
방법 1	a[0]	a[1]	a[2]	a[3]
방법 2	*a	*(a+1)	*(a+2)	*(a+3)
방법 3	*p	*(p+1)	*(p+2)	*(p+3)
방법 4	p[0]	p[1]	p[2]	p[3]

예제 10 - 6

포인터와 배열에 대해 더 잘 알 수 있었다.

for문을 이용한 배열의 주소값과 배열 값 표현을 알 수 있었기에 활용할 수 있을 것 같다는 생각이 들었다.

p++: 포인터 p가 가리키는 곳의 다음 주소를 의미

*p++: 포인터 p가 가리키는 곳의 다음 주소 안에 들어 있는 값을 의미

a++: 배열명 a는 상수이므로 증감 연산자를 사용할 수 없다.