11 함수의 활용과 심화

11-1. 함수의 인자 전달 II

11-1-1. 배열을 함수의 인자로

11-1-2. 구조체를 함수의 인자로

11-2. 함수와 포인터

11-2-1. 함수의 주소

11-2-2. 함수 포인터의 선언과 사용

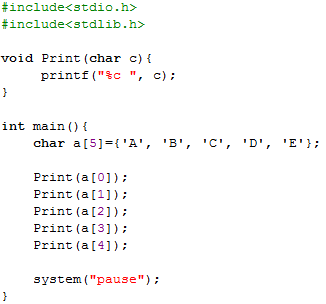
지금까지 함수의 기본 개념과 반환형 등을 알아보았다. 앞 단원을 잘 학습했다면 우린 어떤 수를 인자로 받아서 특정한 작업을 하는 함수를 만들 수 있다. 이제는 함수의 기본 개념에서 나아가 배열, 구조체를 함께 쓰는 방법들에 대해 배울 것이다.

11-1. 함수의 인자 전달 II

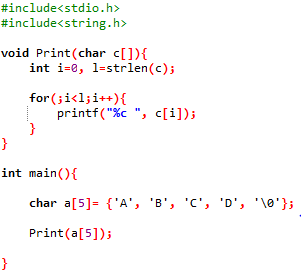
앞 장의 함수의 인자 전달 I 단원에서 우리는 함수에서 인자를 전달하는 두 가지 방법론에 대해 알아보았고, 그 중 참조에 의한 전달(Call by reference)가 매우 유용하게 사용될 수 있다는 사실을 학습하였다. 이번 장에서는 이를 활용해 함수의 인자로써 배열과 구조체를 어떻게 다룰 수 있는지 예제를 통해 학습하는 시간을 가져 보도록 한다.

11-1-1. 배열을 함수의 인자로

어떤 정보를 처리할 때, 값 하나하나를 처리하기도 하지만, 여러 개의 정보가 나열된 배열을 통째로 처리해야 할 때도 있다. 함수에서는 인자로 값 하나뿐만 아니라 배열도 받을 수 있다. 예시를 보여줄 함수로, 배열 안에 있는 값을 출력하는 Print 함수를 만들자. 먼저, 문자 배열 char a[5]를 선언하고, 아래와 같이 초기화 해준다.

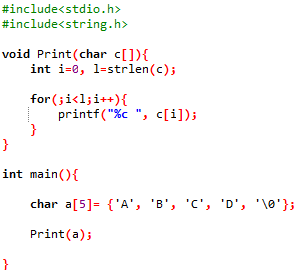


하지만, Print 함수가 값을 일일이 받아서 출력하는 것보다, 배열 자체를 받아서 한꺼번에 출력해주는 것이 훨씬 효율적이다. 그렇다면 배열 a를 Print 함수의 인자로 받아야 하는데, 어떻게 받을 수 있을까? 아래 코드를 실행시켜보자.



일단, 전 코드와 비교해 가장 중요한 차이점은 Print 함수 안에 char c[]이다. 문자를 받을때는, char c만 써줘도 되지만, 지금은 배열을 받을 것이기 때문에 배열을 받는다는 표시로 [] 까지 붙여줘야 한다. 계속해서 보자.

Print 함수에서 char c[] 형태를 받는다고 되어있어서 메인 함수에서 호출할 때 a[5]로 써 주었다. 실행을 하게 되면 실행되지 않고 오류가 뜬다. 이유는 a[5]가 배열의 이름이 아니기 때문이다. Print 함수에서 char c[]라고 표기한 것은 ‘난 이제 배열을 받을거야’ 라는 말이지, 배열을 써줄 때 크기까지 적어달라는 것이 아니다. 따라서 배열의 이름인 a 만 써줘야 한다. 아래 코드로 다시 한 번 실행시켜보자.



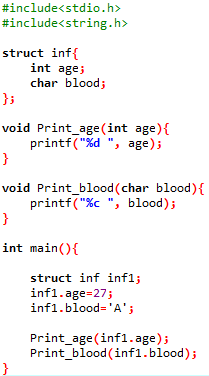
한 가지 보자면, 위 두 코드는 string 헤더와 그 안의 strlen 함수를 사용하였다. string.h가 문자열에 대한 함수들을 저장해놓은 헤더라는 것과 strlen 함수가 문자열의 길이를 나타내주는 함수라는 것을 ‘문자와 문자열’ 단원에서 학습하였다. 혹시라도 까먹었다면 돌아가서 복습하고 오자.

참고로, 배열의 한 요소만 함수에 전달하게 되면, 배열 자체는 전달되지 않는다. 당연하다. 필통에서 연필을 꺼내줬다고 필통 안에 있는 모든 학용품들을 알 수 있는 것은 아니기 때문이다.

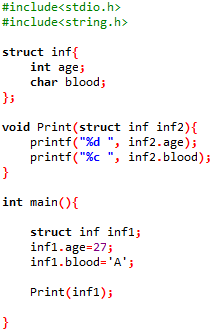
11-1-2. 구조체를 함수의 인자로

구조체 역시 함수와 맞물려서 사용될 수 있다. 예를 들어, 개인 정보는 정수(나이), 문자(혈액형), 문자열(이름, 전화 번호) 등의 데이터 형들이 있는 구조체이다. 개인 정보를 관리하는 함수가 있다고 하면, 구조체를 함수 안으로 받아와야 한다. 구조체는 배열과 비슷하긴 해도 다르기 때문에 함수에서 받을 때도 배열과는 조금 다르다. 구조체와 함수가 어떻게 쓰이는지 계속 알아가 보자.

구조체 안의 값을 인자로 받는 방법은 이미 배웠으므로 어렵지 않을 것이다.

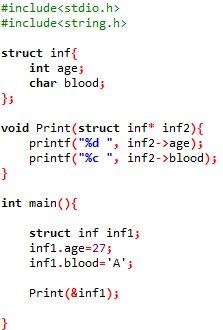


나이와 혈액형 정보를 갖는 구조체 inf를 만들고, 메인 함수에서 구조체 변수 inf1을 선언, 초기화 해준다. 값을 인자로 받으려고 했더니, 구조체 안의 데이터 형이 다르다. 따라서 출력하는 함수를 두 개를 만들고, 각각의 함수를 호출해주었다. 이해하기 쉬운 코드이지만, 출력하는 함수를 두 개 만든 것부터 비효율적이다. 따라서, 값을 보내기 보다는 구조체를 통째로 보내주는 것이 더 효율적인 출력 방법일 것이다. 다음 코드를 실행해보자.



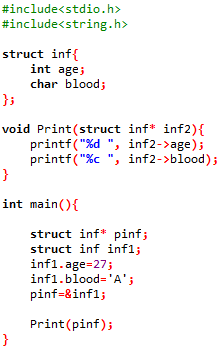
구조체 자체를 받기 위해서 Print 함수의 매개변수 부분을 위와 같이 써주었다. 데이터 형이 struct inf 이므로 ‘struct inf inf2’라는 표현이 이상하지 않을 것이다. 또, Print 함수를 호출할 때는 값이 아니라 구조체의 이름인 inf1을 써주었다. 결과적으로 구조체가 전달이 되어서, 나이와 혈액형이 하나의 함수에서 출력된다.

한편, 우리는 주소와 포인터를 이용해서 구조체에 접근하는 방법도 배웠다. 다음 코드를 통해, 함수 안에서 주소와 포인터로 어떻게 구조체를 활용하는지 보자.



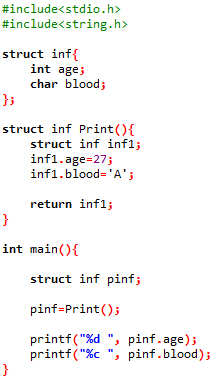
전의 코드와 다르게 이번에는 구조체의 주소를 통해 값에 접근하였다. 초기화하는 부분까진 같지만, Print 함수에서 받아야 하는 것이 구조체가 아니라 구조체의 주소이므로, 주소를 넣어줄 구조체 포인터를 만들어줘야 한다. 따라서 struct inf\* inf2 라고 돼있는 것이다. 당연한 것이지만, 주소를 넣어줘야 하므로 함수를 호출해줄 때 &inf1과 같이 써주어야 주소가 전달된다.

아래와 같은 방법으로도 코드를 짤 수 있다. 아예 구조체 포인터 변수를 선언해서 그 변수 안에 주소를 저장한 것이다. 차근차근 읽어보고 해석하면 이해가 될 것이다.

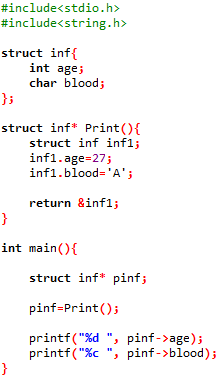


중간 과제: 이제 구조체를 인자로 보내 보았으니 반대로 구조체를 반환하는 함수를 만들어보자. 이전 장에서 함수를 배우면서 반환형에 대해서도 공부했을 것이다.

다음으론 구조체를 반환하는 함수를 한 번 만들어보자. 아래 코드를 실행시켜보자!



구조체를 반환하는 함수는 간단하다. 지금까지 써오던 void Print()의 void 대신에 struct inf를 써주기만 하면 된다. 물론 반환하는 return도 있어야 한다. 살펴보면, 메인 함수에서 비어있는 구조체 pinf를 만들어 놓고, Print 함수를 호출해서 반환해주는 구조체를 pinf에 넣는 것이다. 결과적으로 나이와 혈액형이 잘 출력된다. 그런데, 구조체를 통째로 반환하려니 뭔가 비효율적인 느낌이 든다. (비유). 그렇다면 주소를 이용하여 접근하는 방법은 어떨까? 배열의 첫 주소만 알면 배열의 모든 값에 접근할 수 있듯이 구조체 역시 시작 위치만 알면 구조체 안의 모든 값들에 접근할 수 있다. 때문에 아래 코드와 같이 접근하는 것이 더 효율적일 것이다.



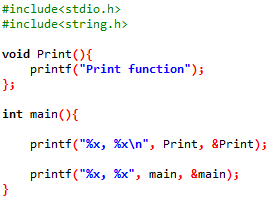
천천히 살펴보자. 이번에는 구조체를 받는 것이 아니라 구조체의 주소를 받을 것이기 때문에 Print 함수의 형태가 좀 바뀌게 된다. 반환하는 것이 주소이므로, \*를 써주어야 한다. 그리고 반환할 때도 구조체 inf1의 주소를 보내줘야 하므로 &를 써준다. 이렇게 반환된 주소를 구조체 포인터 변수 pinf에 넣으면 이제 주소로 접근이 가능해진다. 마찬가지로 포인터 변수를 사용했으므로 값에 접근할 때 -> 연산자를 사용한다.

11-2. 함수와 포인터

11-2-1. 함수의 주소

이번에는 함수 포인터라는 개념에 대해 알아보려고 한다. 엄밀히 말하자면 함수 포인터는 함수의 활용이라기보단 포인터의 활용이다. 포인터 단원에서 배열 포인터와 구조체 포인터에 대해서 알아봤을 것이다. 마찬가지로 포인터에는 함수 포인터라는 것이 있다. 말 그대로 함수를 가리키는 포인터인 것이다. 배열의 이름이 배열의 시작 주소를 표현하듯이 함수의 이름도 함수의 주소를 표현한다.

다음 코드는 Print 함수와 main 함수의 주소를 출력하는 코드이다.



함수의 이름을 넣고 출력했더니 주소를 출력했을 때와 같은 결과가 출력된다. 즉, 함수의 이름이 함수의 주소도 나타낸다는 것을 알았다.

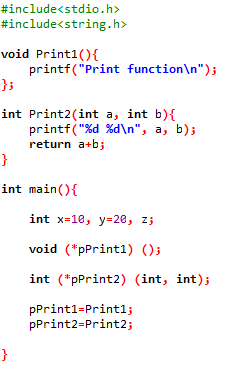
11-2-2. 함수 포인터의 선언과 사용

그렇다면 본격적으로 함수 포인터에 대해 알아보자. 함수 포인터의 문법은 배열 포인터의 문법과 비슷하다.

(함수의 반환형) ((\*함수 포인터 이름)) (인수 자료형 목록);

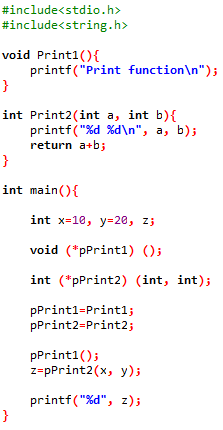
int (\*fpoint) (int, char);

배열 포인터와 비슷하게 괄호 안에 \*가 있고 포인터 이름이 있다. 다른 점은 배열 포인터가 뒤에 배열의 크기를 나타냈다면, 함수 포인터는 뒤에 인수들의 자료형을 적어줘야 한다는 것이다. 함수 포인터를 선언할 때는 함수의 반환형과, 인수의 종류와 개수를 맞춰서 써주어야 한다. 아래 코드를 통해 함수 포인터가 어떻게 선언되는지 보자.



Print1 함수의 경우 반환할 것이 없는 void형 함수이고 인수로 받는 것도 없다. 따라서 반환형이 void이고 인수가 없는 함수 포인터 pPrint1을 선언해서 Print1 함수의 주소를 넣는다. Print2 함수의 경우는 int 형 데이터를 반환하고, int형 데이터 두 개를 인수로 가지기 때문에, pPrint2 형태의 함수 포인터를 선언하고, 주소를 저장해야 한다.

그렇다면 이 함수 포인터들이 어떻게 사용될까? 전 코드에 이어 다음 코드를 완성해보자.



void형 함수인 Print1 함수의 경우는 값을 리턴하는 것이 아니므로 그냥 함수만 호출해주면 된다. pPrint1(); 의 결과로 Print function 이 출력된다. 반면, pPrint2(x, y)는 정수를 반환해주는 함수이므로 반환 값을 담을 상자가 필요하다. 그 상자가 바로 z이다. 한편 Print2 함수가 받은 인수 2개를 출력하기도 하므로 25번째 줄의 결과로 x, y가 출력된다. 마지막으로 Print2 함수의 결과인 z를 출력하는 것으로 코드가 끝난다.

함수 포인터를 이용하면, 함수를 직접 호출하는 것보다 처리 속도가 빠르다. 따라서, 자주 사용하는 함수의 경우 함수 포인터를 이용하면 프로그램을 더욱 빨리 돌릴 수 있다.

지금까지 앞의 포인터 I, II 장과 함수 장에서 배운 여러 개념들을 모두 종합하여 활용하는 함수 포인터의 개념에 대해 알아보았다. 조금 어려운 개념이고, 이게 어디에 쓰일까 싶은 것이긴 하지만 독자 여러분이 이를 잘 숙지한다면 한 차원 더 수준 높은 프로그래머가 될 수 있을 것이다. 수고했어요~