C프로그래밍 실습

**#6 Functions**

* Organizing functions

함수는 C프로그램의 블록이다: 자주 사용하는 프로그램문을 함수로 만들 수 있다. 그리고 재사용도 가능하다

프로그램 코드 재사용성 때문에 생산성이 증가한다.

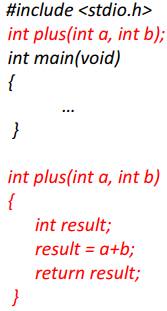
함수안에 프로그램 코드가 조직된다.

보통의 프로그램을 짤 때 다른 프로그래머들과 협력하기 좋다.

* Creating and Using a Simple Function

사용할 함수를 먼저 선언해야 한다.

1. Data type of return value (output): 대부분의 함수는 값을 return한다. (1 or none)
2. Function name: ex) printf, scanf…
3. Parameters or arguments: none or one or more input이 있다.이때 관련있는 이름과 data type을 써야한다.

함수 선언 방법 (prototype the function)

* [return data type] [function name] (parameters);

만약 return value가 없을 때, argument가 없을 때

void 키워드를 사용할 수 있다.

함수 선언 위치

Function이 사용되기 전에

보통은 int main(void) 함수 전에 사용한다.

함수 작성은 main function의 바깥에 한다.

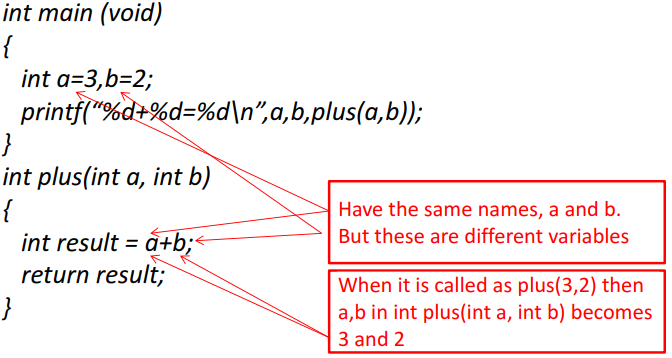
문장 블록과 함께 함수를 선언

함수를 멈추고 싶을 때 그리고 값을 도출하고 싶을 때 Return을 사용한다.

함수 사용, 호출하기: 어떤 block의 안에서 적절한 인자와 함수의 이름을 선언한

형태로 작성하여 함수를 호출 할 수 있다.

* Defining a Function with one or more arguments

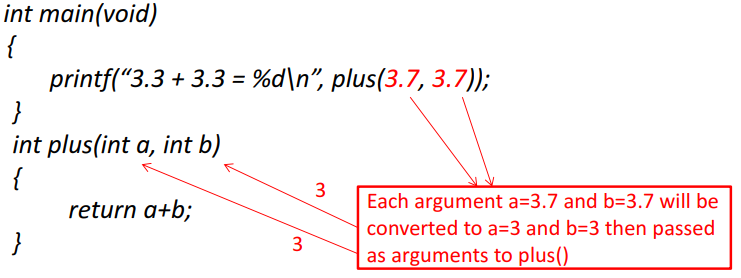
argument는 data value들을 함수 블럭으로 옮긴다.

이때 main함수의 변수 a, b와 plus함수의 인자 a, b는 같은 이름이지만 다른 변수이다. 만약 plus(b,a)로 main함수에서 호출하면, plus함수에서 int a에는 b의 값에 해당하는 2가, int b에는 a의 값에 해당하는 3이 들어간다.

* void dubs(int x, int y, int z); / void dibs(int x,y,z) (x)
* Calling a Function with an Argument: Actual Arguments

함수블록을 호출할 때 argument의 데이터형은 반드시 함수를 선언할 때의 데이터형과 일치해야 한다.

즉, C컴파일러는 다른 데이터형의 값을 선언된 데이터형으로 변환한다. (type conversion)



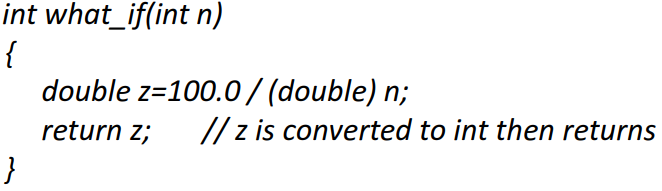
>>> 3.3+3.3 = 6

* Returning a Value from a Function with return

함수의 결과값을 return하고싶을때는 return키워드를 사용한다

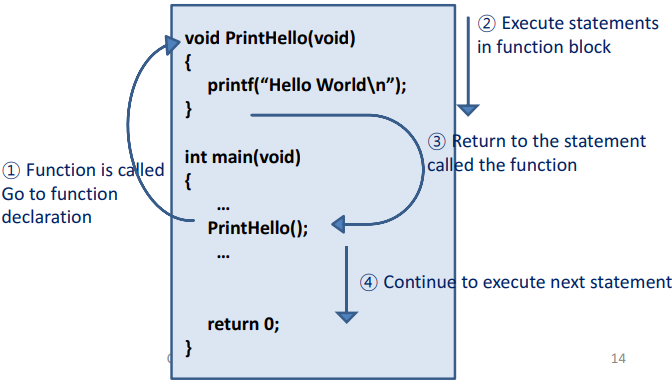
이때 오로지 하나의 데이터 값만 return 할 수 있다.

Return 데이터 형이 맞지 않으면, 처음에 선언했던 함수의 return 데이터형으로 형변환이 일어난다. (type conversion)



* Omission of Function Declaration

만약 function block이 호출되기 전에 코딩된다면, 해당 function block이 생략될 수 있다.

웬만하면

declare => main function => function block 형태로 코딩하자

* Recursion (Recursive Function): 재귀함수

함수가 자기 자신을 호출하는 것을 말한다.

Loop와 비슷하게 recursion은 반드시 어떤 지점에서 종료되어야 한다.

(무한히 반복될 수 있기 때문에)

대부분 if문을 사용해서 자기 자신을 호출할 때의 조건을 설정한다.

recursion의 경우에, if문에서 자기자신을 호출한다.

반대의 경우에는 if문은 함수 호출에 대해 return해서, recursion을 종료한다.

* Recursion Fundamentals

함수호출에서 각각의 level은 자기 자신의 데이터 변수를 갖는다.

Level1의 int n은 level2에서의 변수와 다르다.

Recursion은 level4까지 총 4개의 각기 다른 int n 변수를 만든다.

* Tail(End) Recursion

recursion의 가장 심플한 형태

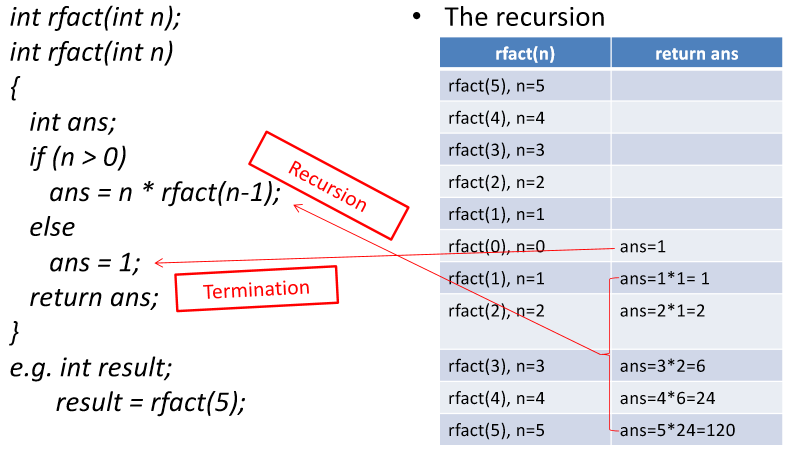
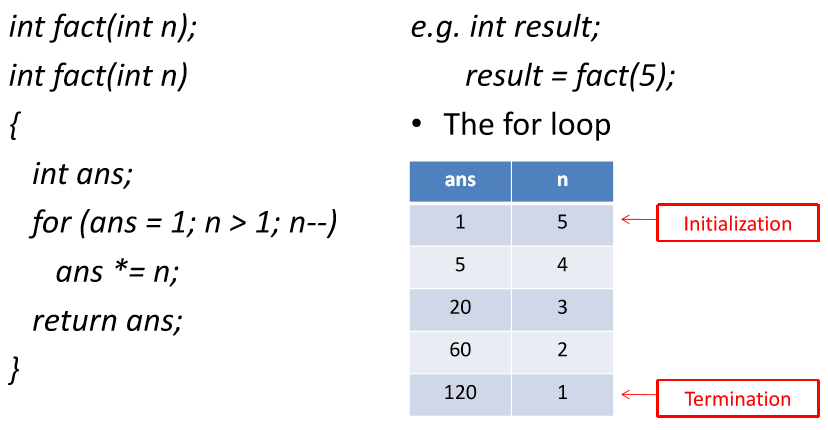
recursion이 자기 자신을 호출하는 것은 함수의 가장 마지막에 있다.

return의 바로 전 or 함수 블록의 제일 끝 줄

tail recursion은 “loop”처럼 실행된다.

Ex) factorial: 5! = 5\*4\*3\*2\*1 = 120

[loops] [recursion]



* Loop VS Recursion

보통은 loop가 좋은 선택이다.

recursion보다 더 작은 메모리 공간이 필요하다: recursion은 여러 번 변수를 복사한다

recursion보다 overhead가 적게 일어난다: 각각의 함수에서 overhead를 초래한다.

하지만 때때로 변환할 간단한 형태의 loop가 없다.

recursion만을 선택한다.

recursion은 어떤 프로그래밍 문제에서는 가장 간단한 해결책이다.

EX) Fibonacci number: loop보다 recursion이 더 간단한 경우이다.