# 재귀함수를 이해하기 위한 질문

* 재귀함수의 의미는 무엇인가? 도대체 이걸 왜 쓰나?
* 재귀함수를 만들기 위해서는 어떤 절차를 따라야 하나?

계산하는 방법에는 크게 2가지 방법이 있다.

1. 순차적인 방법
2. 귀납적인 방법

*** 순차적 계산법**

A를 계산한다.
A를 이용해서 B를 계산한다.
B를 이용해서 C를 계산한다.
C를 이용해서 D를 계산함으로써 원하는 결과를 얻는다.

예를 들어, 4의 약수를 구하라.

1로 나누어지면 count를 증가..

2로 나누어지면 count를 증가

단계적으로 나누어 접근하는 방법을 순차적인 방법이라 한다.

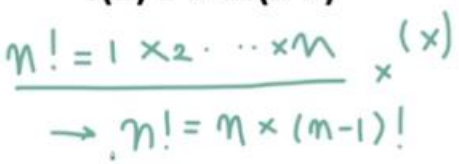
**즉, 반복문이 대표적인 순차적인 방법이다.**

## * 귀납적 계산법

구하려고 하는 값을 f(x)라고 하자.

f(x)를 구하기 위하여 또 다시 f(x)를 활용한다.

**f(n) = n x f(n-1)**



팩토리얼을 구하기 위해 팩토리얼을 쓴다.

f(5) = 5 x f(4) = 5 x 4 x f(3) .... = 5 x 4 x 3 x 2 x 1 x f(0)

**귀납적으로 정의해야 하는 것들은 반드시 멈추어야 하는 지시가 있어야 한다.**

0! = 1 (정의)

n ! 은 n x (n-1)! , 0! = 1 이라는 정의가 필요하다.

n^m 을 귀납적으로 정의하면 다음과 같다.

$n^m$ 을 귀납적으로 계산하여라

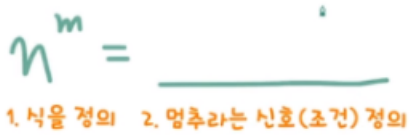
$$n^m = n^{m-1} \times n$$

(n을 m번 곱하는 것) = (n을 m-1번 곱하는 것) x n

"나" 를 계산하기 위해 또 다시 "나"를 활용한다.

# 귀납적 계산법의 깊은 이해

$n^m$ 을 귀납적으로 계산하여라




5^4을 구한다고 가정하자.

5를 네 번 곱한 값이 제대로 나올려면 **'5^3이 5를 세 번 곱한 값이다'** 라는 가정이 필요하다.

즉, 5의 4승이 5를 네 번 곱한 값이 나올려면 5의 3승은 5를 세 번 곱한 값이다라는 가정이 성립을 해야 한다.

5의 3승이 5를 세 번 곱한다면 한 번 더 5를 곱하여 5의 4승을 만들 수 있다.

5의 3승은 5의 2승 곱하기 5이다 -> 5의 2승은 5를 두 번 곱하다라는 가정으로 맞추어 짐.

5의 1승이 정말로 5를 한 번 곱할까? ... 여기서 함수의 관계가 제대로 나오게 된다.

**여기서 팩토리얼에서 보았던 재귀 호출의 흐름이 반복되는 것을 알 수 있다.**

5의 1승은 5^0 x 5... **'5의 0승은 1이다' 라고 정의가 되어 있음. (약속)**

$$n^m = n^{m-1} \times n,\ n^0 = 1$$

$5^4 = 5^3 \times 5$

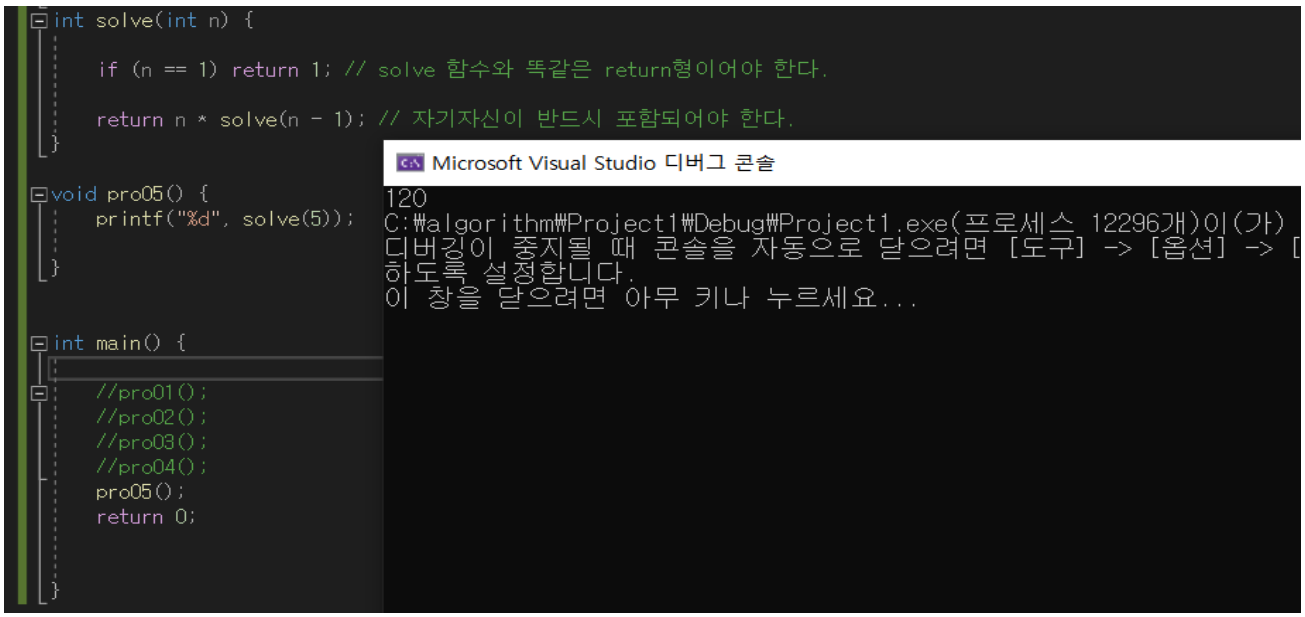
$5^3 = 5^2 \times 5$

$5^2 = 5^1 \times 5$

$5^1 = 5^0 \times 5$

$5^0 = 1$

- 팩토리얼 풀이




1. 여기서 main 함수가 가장 먼저 실행된다.
2. n이 1이 아니기 때문에 5 * solve(4) 가 가장 먼저 호출
3. 여기서 solve(4)가 역시 n이 1이 아니기 때문에 solve(3) 호출
4. 여기서 solve(3) 역시 n이 1이 아니기 때문에 solve(2) 호출
5. 여기서 solve(2) 역시 n이 1이 아니기 때문에 solve(1) 호출
6. 여기서 solve(1)은 1이므로 return 한다. 여기서부터 역으로 연산이 이루어진다.

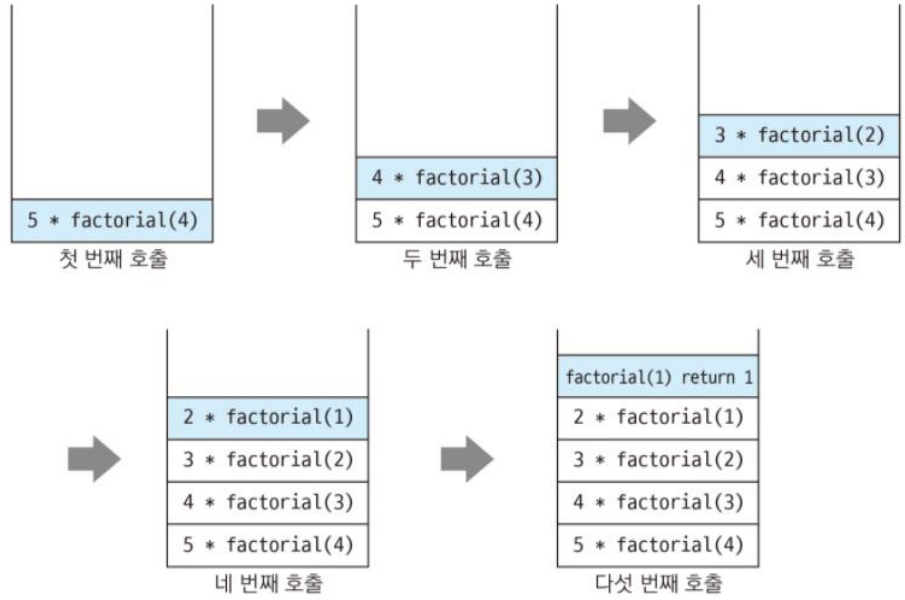
```
factorial(5) = return 5xfac(4)
factorial(4) = return 4xfac(3)
factorial(3) = return 3xfac(2)
factorial(2) = return 2xfac(1)
factorial(1) = return 1;
//여기서 기반조건이 충족된다. 재귀함수는 안에서부터 바깥으로 값을 반환해나간다.!
factorial(1) = return 1; => output=1;
factorial(2) = return 2xfac(1) => output = 2x1 = 2;
factorial(3) = return 3xfac(2) => output = 3x2 = 6;
factorial(4) = return 4xfac(3) => output = 4x6 = 24;
factorial(5) = return 5xfac(4) => output = 24x5 = 120;
```

재귀함수에서 가장 중요한 것 (재귀함수는 반복과 스택의 결합이다)

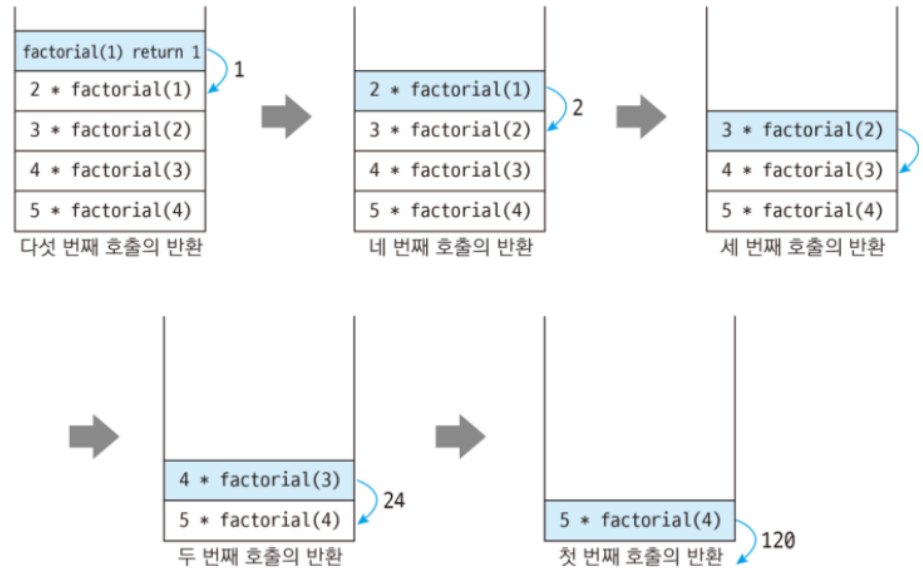
1. return 하는 식을 정의 하는 것 (식 세우기)
2. 재귀호출이므로 탈출조건(=기저조건)을 주는 것.

재귀함수가 스택인 이유 => 호출은 하는데 리턴을 못하기 때문

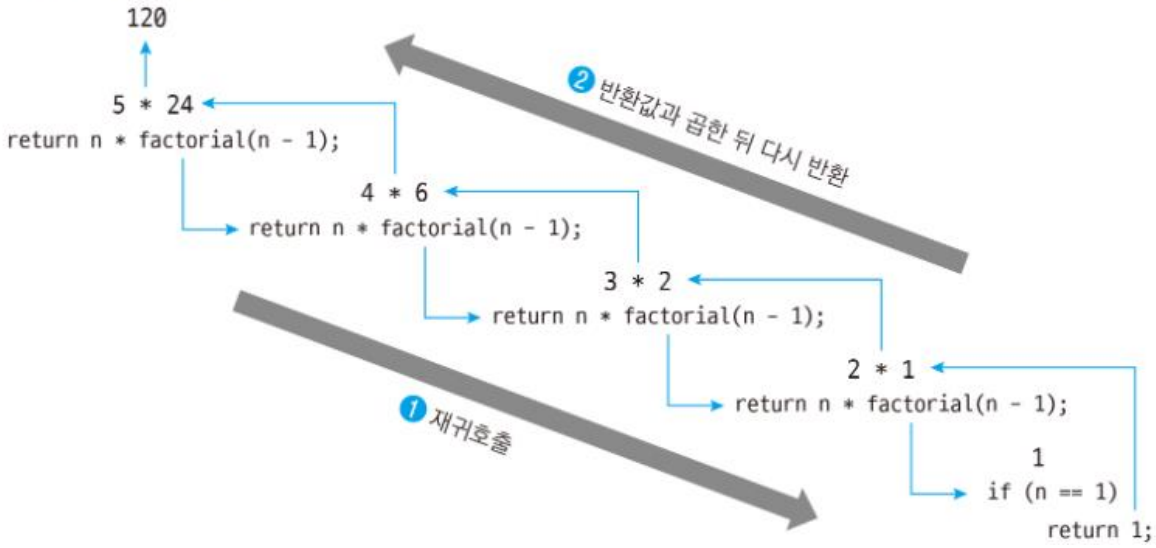
▼ 그림 67-3 팩토리얼 함수의 호출



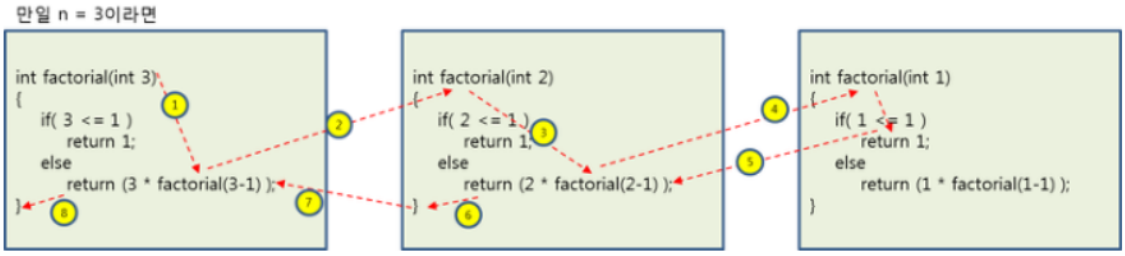

▼ 그림 67-4 팩토리얼 함수의 반환



▼ 그림 67-5 factorial 함수의 호출 순서와 계산 과정




구체적인 함수흐름




< 함수 흐름도 >

자기 자신을 호출한다고 하지만, 실제로 동작 순서는 위와 같다. 재귀함수를 이해할 때는 항상 위와 같은 그림을 연상하는게 정신건강에 이롭다.

이롭다고 한다~!

재귀함수의 효율성

**재귀함수의 효율성**이란 속도나 메모리 절약을 의미하는 것이 아닙니다. 오히려 잦은 **함수** 호출로 인한 속도 저하와 스택 오버플로우(overflow), 즉 다룰 수 있는 범위를 초과하는 문제가 발생할 수 있는데요. 논리적 사고를 그대로 옮기거나 코드를 단순화하기에 적합합니다.
2016. 7. 10.

## 재귀 VS 반복

- 보통 반복이 재귀 보다 성능면에선 효율이다.
  (재귀에서 종료조건이 잘못된 경우, 무한루프에 빠질 수 있다. Stack Overflow)
- 재귀는 반복보다 코드 면에서 간단하다. -> 가독성이 높다.
- 재귀는 종료조건이 반드시 명시되어야 한다.

왜 사용하는가?

재귀함수를 쓰는 이유
- 1. 알고리즘 자체가 재귀적으로 표현하기 자연스러울 때. (가독성 이야기)
- 2. 변수 사용을 줄여 준다.

1번은 알고 있었던 내용이나, 2번에 대한 설명은 오! 라는 감탄사를 가져다 주었다.

변수 사용을 줄여준다는건 변수가 잡는 메모리에 대한 이야기가 아니라, mutable state (변경 가능한 상태) 를 제거하여 프로그램 오류가 발생할 수 있는 가능성을 줄인다는 이야기다.

출처

https://velog.io/@violet/TIL%EC%9E%AC%EA%B7%80%ED%95%A8%EC%88%98%EC%97%90-%EB%8C%80%ED%95%9C-%EA%B0%84%EB%8B%A8%EC%A0%95%EB%A6%AC

https://leedr0730.medium.com/%EC%9E%AC%EA%B7%80%ED%95%A8%EC%88%98%EC%9D%98-%EC%9B%90%EB%A6%AC-3863bef83b9c

https://treeroad.tistory.com/entry/%EC%9E%AC%EA%B7%80%ED%95%A8%EC%88%98

https://medium.com/sjk5766/%EC%9E%AC%EA%B7%80%ED%95%A8%EC%88%98%EB%A5%BC-%EC%93%B0%EB%8A%94-%EC%9D%B4%EC%9C%A0-ed7c37d01ee0