Java를 알고 C배우기

컴퓨터프로그래밍3

week 4-4 함수와 스택-재귀함수

2022.1학기

충남대 조은선

분할 정복

```
n! = n * (n-1) * (n-2) * ... * 1 (단, 1! = 1)
```

```
n! = n * (n - 1)!
```

- 분할 정복(Divide and Conquer)
  - 작은 문제 역시 큰 문제와 동일한 성격일 때 사용

예

1) 5!을 구하는 문제는 4!을 구하는 문제로, 이는 다시 3!을 구하는 문제로, …, 결국 1!을 구하는 문제로. 정의에 의해 1!은 1

2) 도미노 게임(100번째 막대기가 반드시 쓰러짐을 증명하라.)

cf. 수학적 귀납법(Mathematical Induction)과는 반대 순서

재귀 함수와 재귀 호출

- 재귀 호출(Recursive Call)
  - 실행 도중 자기 자신을 호출(Self Call)
- 재귀 함수
  - 자기 자신을 호출하는 함수
  - **base case 부분**과 **재귀 호출 부분**으로 구성됨
  - 호출할 때 마다 지역 참조 환경이 달라짐!

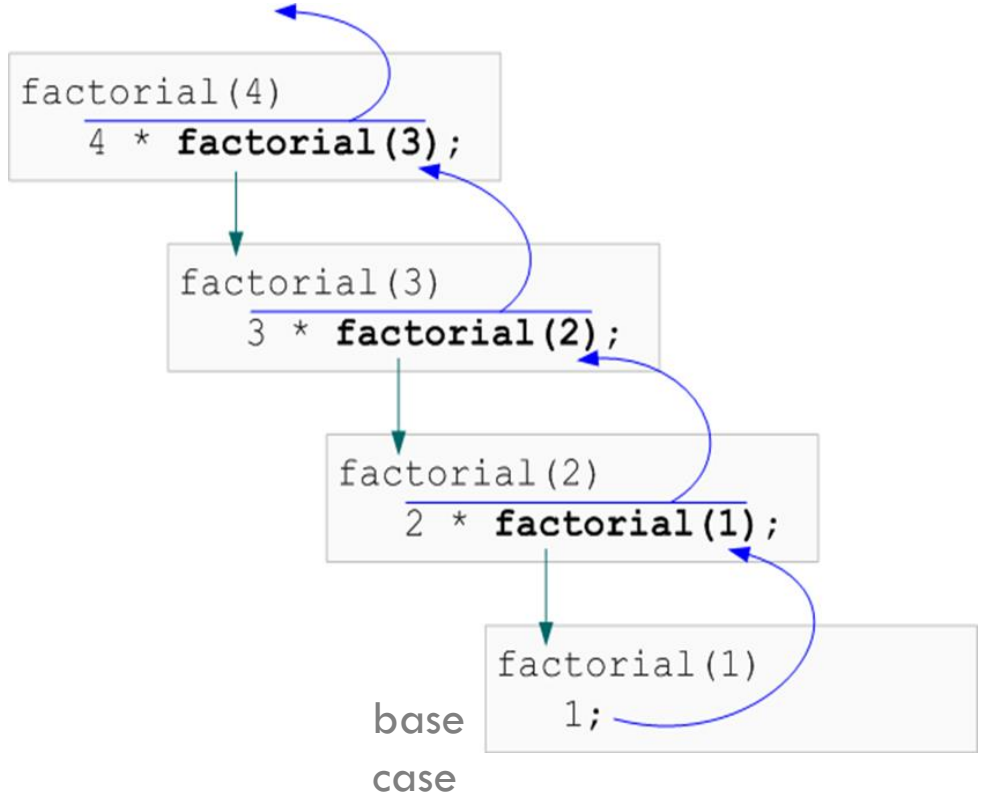
```
long factorial(int n) {
        if (n == 1) return 1;
        return n * factorial(n-1);
}
```

base case

재귀 호출

Base Case

- 베이스 케이스(Base Case)
  - 또는 Degenerate Case
  - 문제 크기가 충분히 작아져서 직접 해결할 수 있는 경우
  - 재귀 호출은 반드시 베이스 케이스에 도달해야 함
    - 그렇지 않으면 스택 오버플로우(Stack Overflow) 오류



재귀 함수와 반복문

- 반복문과 재귀함수의 관계
  - 재귀함수는 반복문을 표현가능
  - 그러나 재귀함수가 속도, 성능면에서 불리하다
    스택프레임 생성, 소멸
  - 재귀함수의 장점
    간명한 코드, 반복문으로 표현 힘들 때

새로 호출할 때는 새로운 참조 환경 구성

```
01  #include <stdio.h>
02
03  void fruit(int count);
04
05  int main(void)
06  {
07      fruit(1);
08
09      return 0;.
10  }
11
12  void fruit(int count)
13  {
14      printf("apple\n");
15      if (count == 3) return;
16      fruit(count + 1);
17  }
```

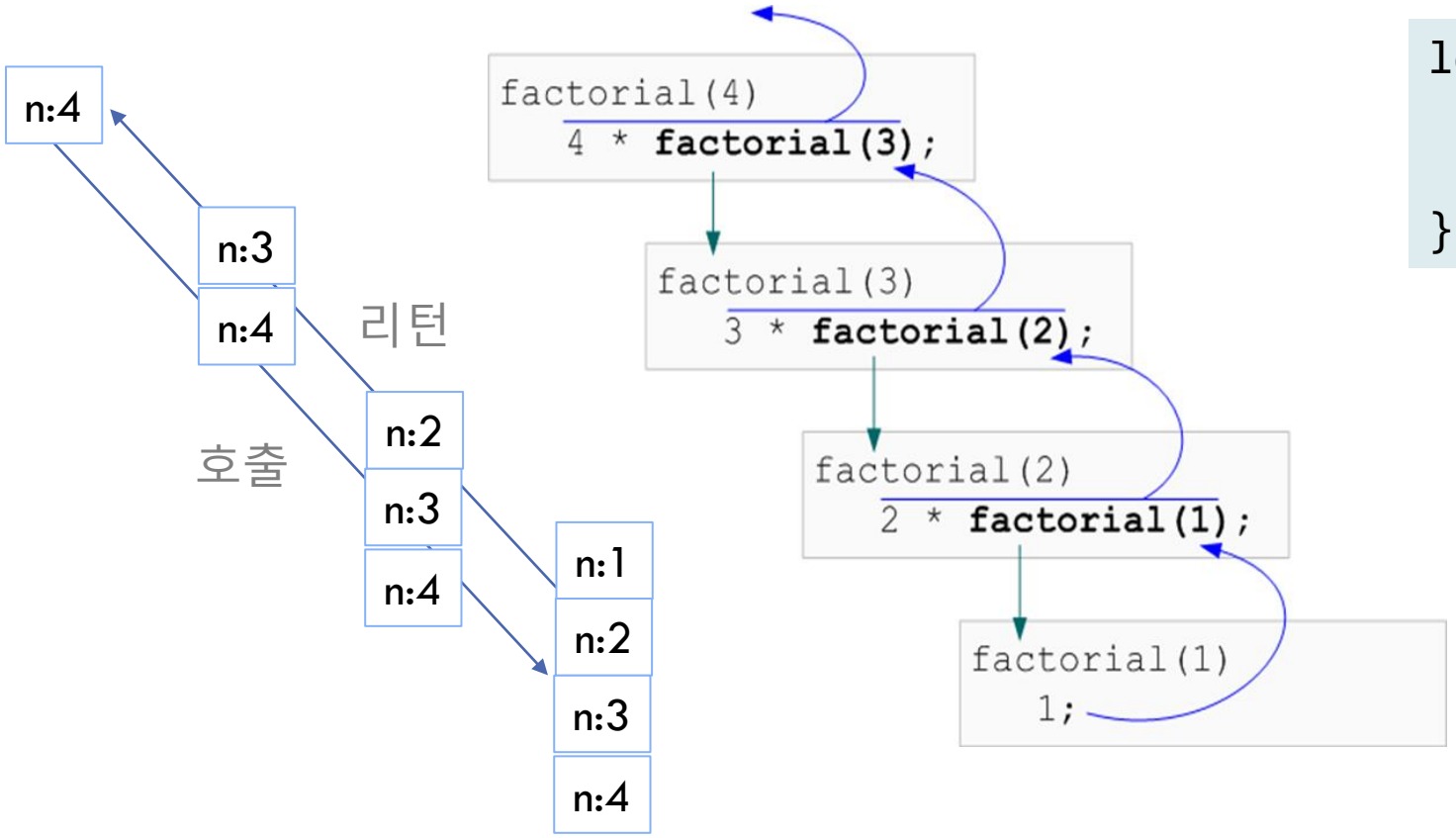
3번 반복

base 조건

재귀 호출

재귀함수와 참조환경

- 새로 호출할 때는 새로운 참조환경 (스택프레임) 구성




```
long factorial(int n) {
        if (n == 1) return 1;
        return n * factorial(n-1);
}
```

꼬리 재귀, 머리 재귀

- printf("%d ", n);
  recurse(n - 1);
  - 꼬리 재귀(Tail Recursion, End Recursion)
  - 먼저 일을 한 다음에 재귀 호출로 들어간다.

```
long fact(int n, int res) {
        if (n == 1) return res;
        return fact(n-1, res*n);
}
long factorial(int n) {
        return fact(n, 1);
}
```

- recurse(n - 1);
  printf("%d ", n);
  - 머리 재귀(Head Recursion)
  - 먼저 재귀 호출을 한 다음에 되돌아오면서 일을 한다.

```
long factorial(int n) {
        if (n == 1) return 1;
        return n * factorial(n-1);
}
```

Quiz

아래와 같은 fibonacci 수열을 생성하는 함수를 작성해보시오.

```
1, 1, 2, 3, 5, 8, 13, 21, ....
fibonacci(n) = fibonacci(n - 1) + fibonacci(n - 2)
단, fibonacci(2) = fibonacci(1) = 1
```