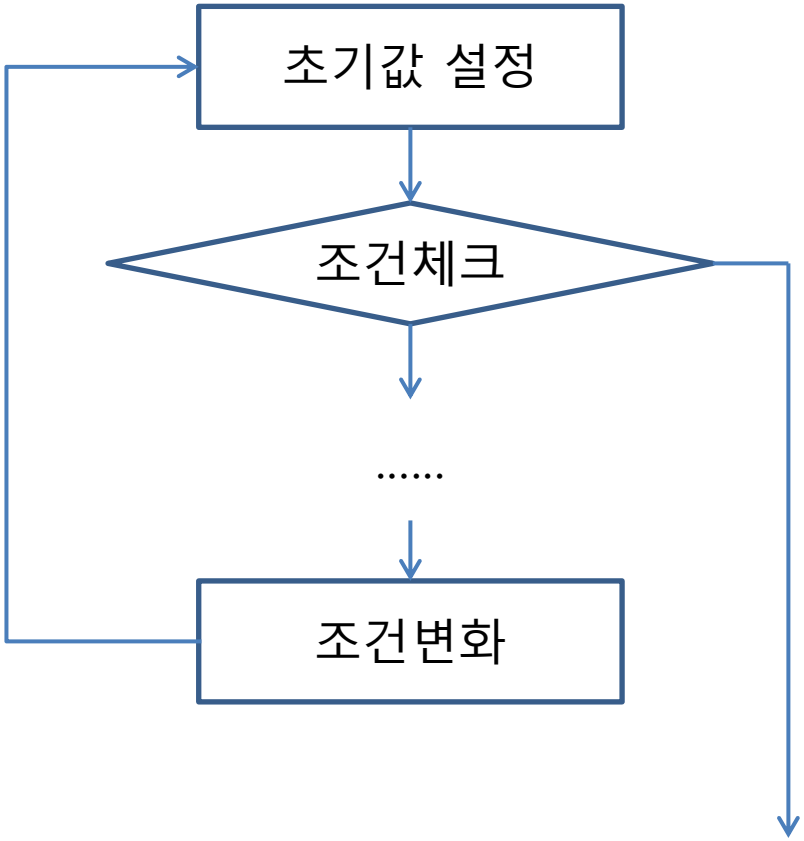
Chapter 02.
재귀

2주차

반복문의 구조와 재귀함수



[for문]

```
for (i = 0;i < 3;i++)
  printf("test");
```

[while문]

```
int i = 0;
while (i < 3) {
  printf("test");
}
```

[재귀함수]
"재귀함수를 호출하지 않고 return하는 경로"가 반드시 있어야 함!
(="조건체크")

```
#include <stdio.h>
void recursive_print(int n) {
  if (n <= 0)
    return;
  printf("test₩n");
  recursive_print(n - 1);
}
void main() {
  recursive_print(3);
}
```

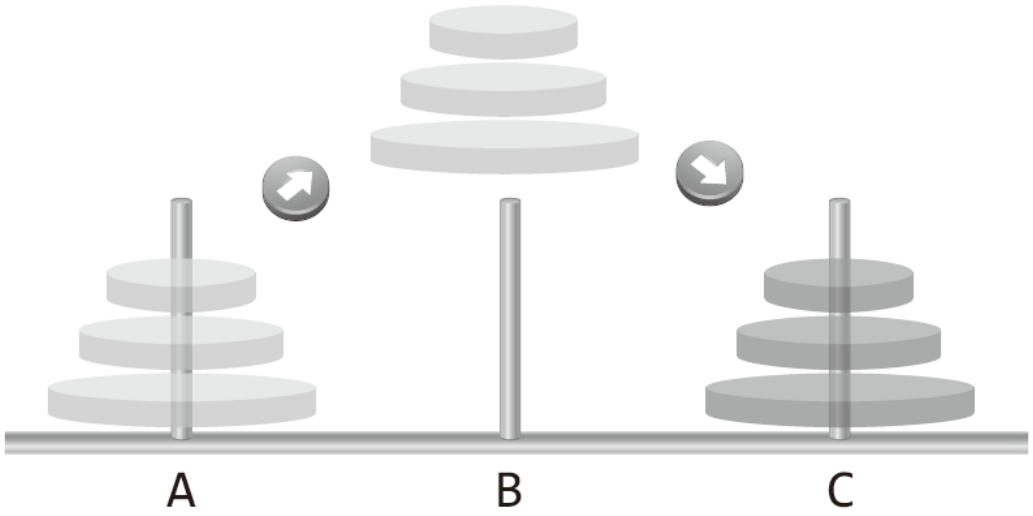
재귀함수의 디자인 사례
(재귀함수의 3가지 구성요소 확인하기!)

```
#include <stdio.h>

int Factorial(int n)
{
    if(n==0)
        return 1;
    else
        return n * Factorial(n-1);
}

int main(void)
{
        printf("1! = %d ₩n", Factorial(1));
        printf("2! = %d ₩n", Factorial(2));
        printf("3! = %d ₩n", Factorial(3));
        printf("4! = %d ₩n", Factorial(4));
        printf("9! = %d ₩n", Factorial(9));
        return 0;
}
```

문제의 성격은 재귀적이지만 구현상으로는 반복문으로 해결하기에도 문제없음.

하노이 타워 문제

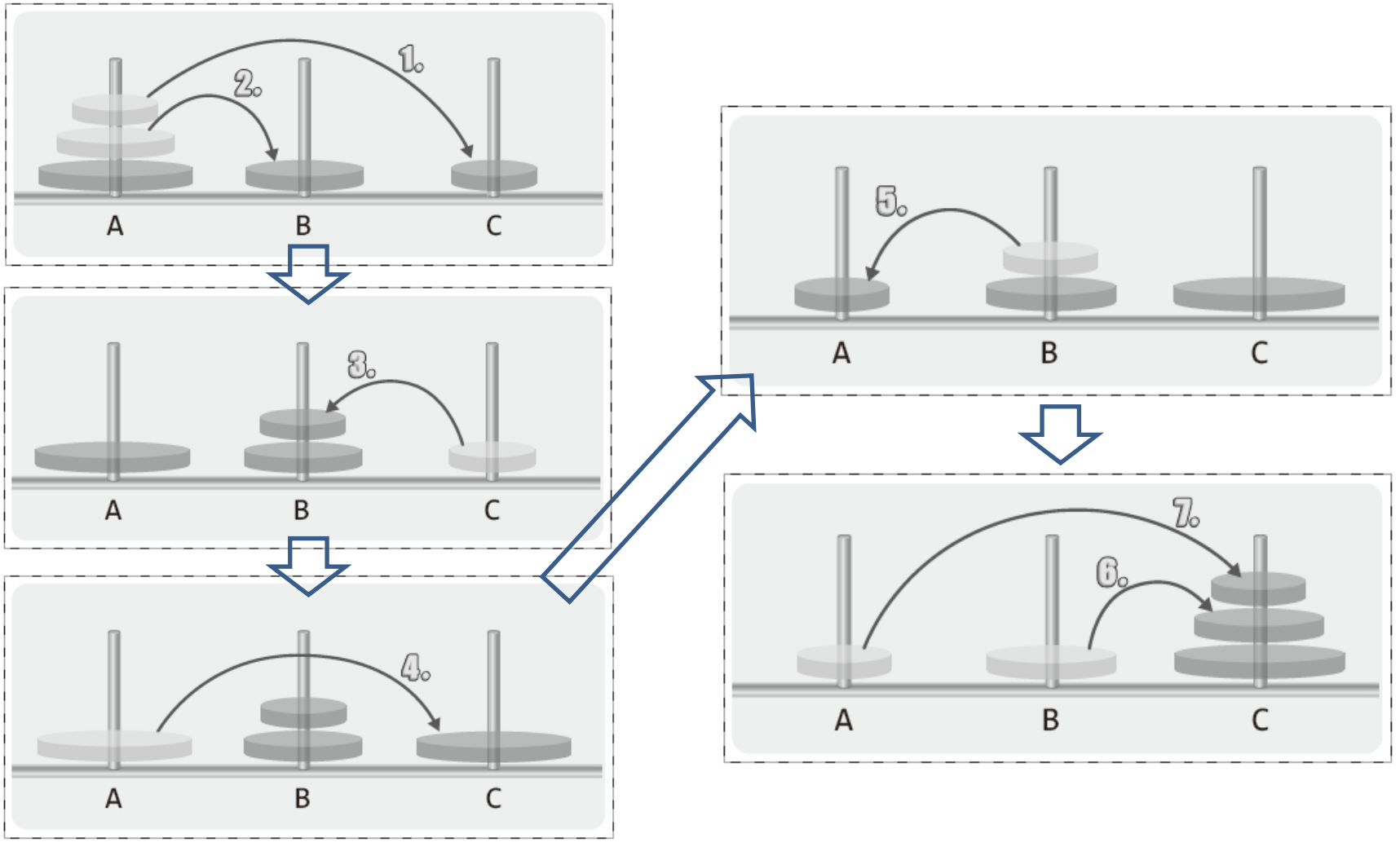
원반을 A에서 C로 이동하기



제약사항

- 원반은 한 번에 하나씩만 옮길 수 있습니다.
- 옮기는 과정에서 작은 원반의 위에 큰 원반이 올려져서는 안됩니다.

하노이 타워 3개 원반 해결
(원반 3개인 경우를 직관적으로 풀어보기)



하노이 타워의 반복패턴 연구
(규칙적인 패턴 정리하기)

- 목적: 큰 원반 n개를 A에서 C로 이동
- 재귀적인 알고리즘
  - 작은 원반 n-1개를 A에서 B로 이동
  - 가장 큰 원반 1개를 A에서 C로 이동
  - 작은 원반 n-1개를 B에서 C로 이동
- 멈추는 조건
  - 원반 1개일 때 : 직접 A에서 B로 이동하고 return.

하노이 문제의 해결
(막대A의 원반 3개를 막대B를 경유하여 막대C로 옮기기)

이동할 원반의 수가 1개라면
1개 옮기고 return.

```
#include <stdio.h>

void HanoiTowerMove(int num, char from, char by, char to) {
    if (num == 1)
        printf("원반1을 %c에서 %c로 이동 ₩n", from, to);
    else {
        HanoiTowerMove(num - 1, from, to, by);
        printf("원반%d을(를) %c에서 %c로 이동 ₩n", num, from, to);
        HanoiTowerMove(num - 1, by, from, to);
    }
}

int main(void) {
    HanoiTowerMove(3, 'A', 'B', 'C');
    return 0;
}
```