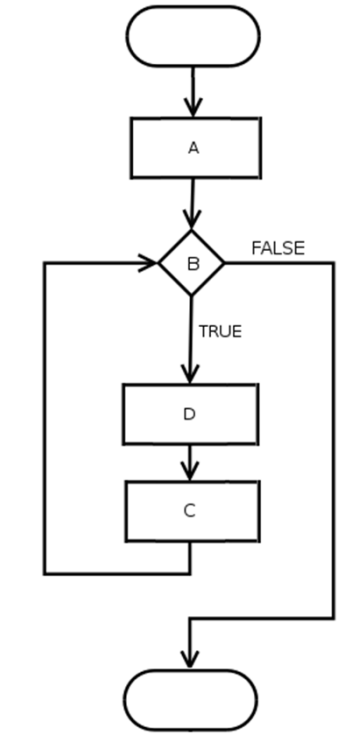
Clickseo.com

자료구조 & 알고리즘

for(A;B;C)
  D;

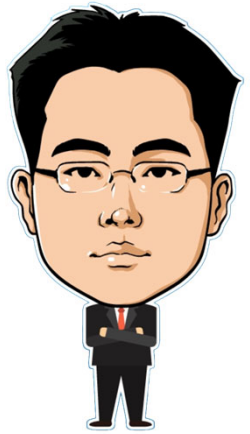


재귀 호출
(Recursive Call)

Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



목 차

- 재귀 함수
- 동적 프로그래밍

재귀 함수

- **재귀 함수**
  - 반복적.재귀적 용법
  - 피보나치 수열
- **동적 프로그래밍**

재귀 함수 (1/3)

- **재귀 함수**(Recursive Function)
  - **자기 자신의 함수를 호출 함으로써, 반복적인 처리를 하는 함수**
    - 재귀 함수 안에서 사용하는 변수는 **지역변수**(자동변수)
    - 재귀 함수의 인수들은 **값에 의한 전달**(pass by Value) 방식으로 전달된다.

주의: 반드시 탈출(종료) 조건 명시!!!

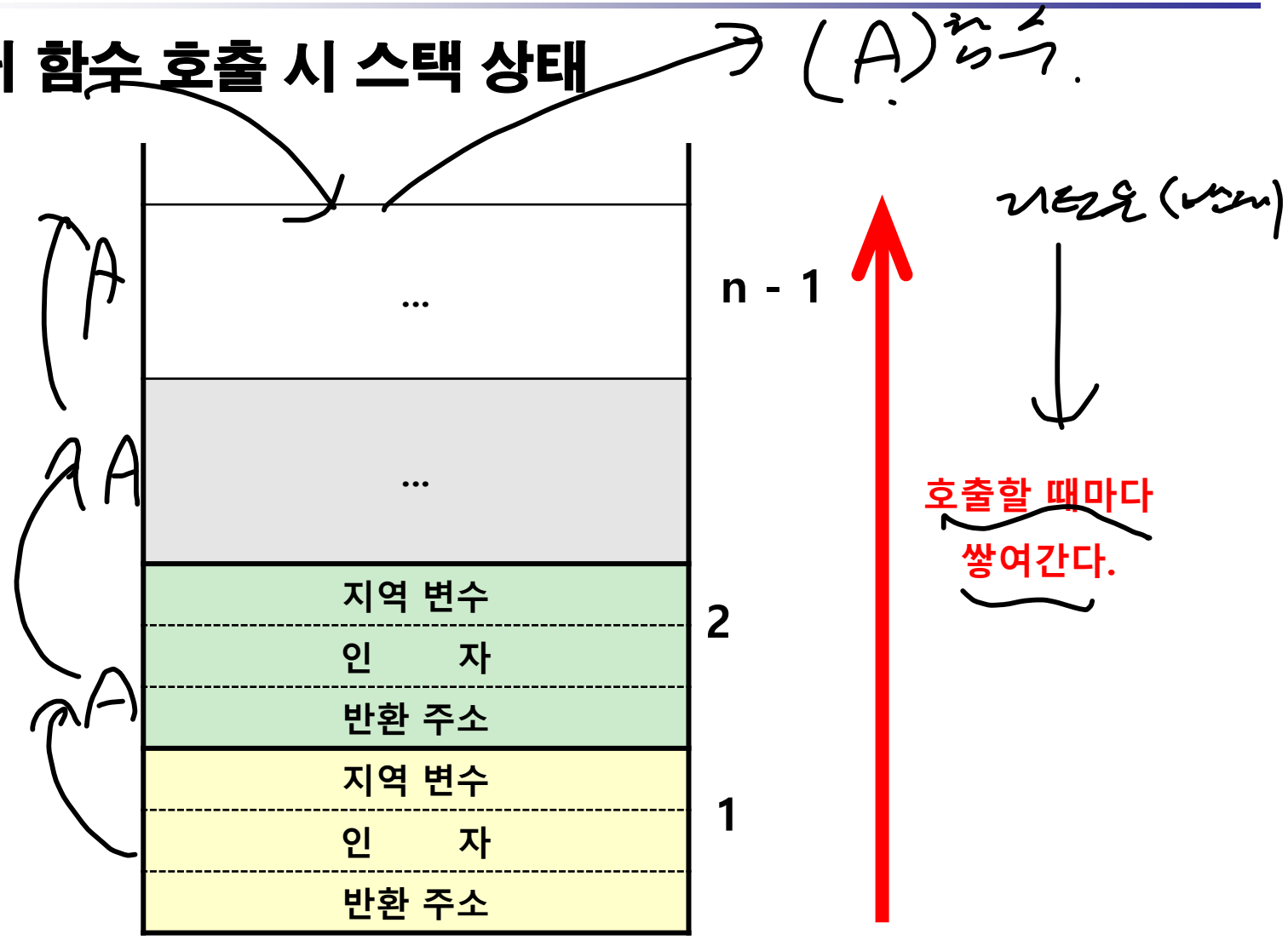
- **Stack Overflow 오류 발생 주의!!!**

- **장점**
  - 코드가 훨씬 간결 해지며, 프로그램을 보기가 쉽다.
  - 또한 프로그램 오류 수정이 용이하다.

- **단점**
  - 코드 자체를 이해하기 어렵다.
  - 또한 메모리 공간을 많이 요구한다.

재귀 함수 (2/3)

- 재귀 함수 호출 시 스택 상태



재귀 함수 (3/3)

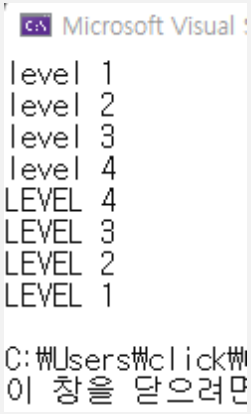
예제 2-1: 재귀 함수 | C | Python

```c
#include <stdio.h>
void OUTPUT(int num);

int main(void)
{
        OUTPUT(1);
        return 0;
}

// 재귀 함수
void OUTPUT(int num) {
    ① printf("level %d\n", num);
       // 재귀 함수 탈출(종료) 조건
    ② if(num < 4)
            OUTPUT(num+1);
    ③ printf("LEVEL %d\n", num);

        return;
}
```




```python
# 재귀 함수
def OUTPUT(num) :
    print(f'level {num}')
    # 재귀 함수 탈출(종료) 조건
  if num < 4 :
    OUTPUT(num+1)
    print(f'LEVEL {num}')


if __name__ == "__main__" :
    OUTPUT(1)
```

```
>>>
==========
level 1
level 2
level 3
level 4
LEVEL 4
LEVEL 3
LEVEL 2
LEVEL 1
>>>
```

재귀 함수

반복적 용법과 재귀적 용법

반복적.재귀적 용법 (1/3)

- **반복적 정의**
  - 반복 함수가 반복적으로 정의된다.
    - 함수 정의는 매개변수를 포함하나 함수 자체는 포함하지 않는다.

$$\text{factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * (n-1) * (n-2) \ldots 3 * 2 * 1 & \text{if } n > 0 \end{cases}$$

- **재귀적 정의**
  - 함수가 자기 자신을 포함한다.

$$\text{factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * \text{factorial}\ (n-1) & \text{if } n > 0 \end{cases}$$

$\Theta(n)$

반복적.재귀적 용법 (1/3)

예제 2-2: 반복적 용법과 재귀적 용법 | C

```c
#include <stdio.h>
int Factorial(int num);
int main(void)
{
        int     num;

        printf("임의의 정수: ");
        scanf_s("%d", &num);
        // scanf("%d", &num);

        printf("%d Factorial: %d \n", num, Factorial(num) );

        return 0;
}
```

```
Microsoft Visual Studio 디버그 콘솔
임의의 정수: 5
5 Factorial: 120

C:₩Users₩click₩OneDrive₩문서₩cClickseo₩x64₩
이 창을 닫으려면 아무 키나 누르세요...
```

```c
# 반복적 용법: Factorial
int Factorial(int num)  {
        int     res = 1;
        for (int i = 1; i <= num; i++)
                res = res * i;
        return res;
}
```

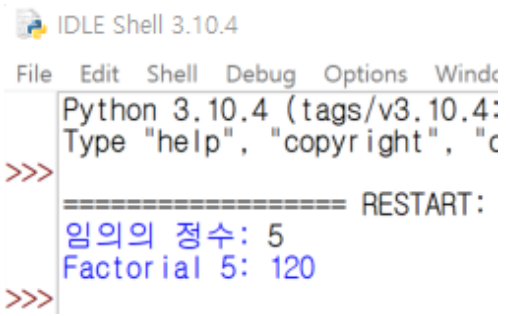
```c
# 재귀적 용법: Factorial
int Factorial(int num)        {
        if(num == 0)
                return 1;
        return num * Factorial (num - 1);
}
```

반복적.재귀적 용법 (3/3)

예제 2-3: 반복적 용법과 재귀적 용법 | Python

```
# 반복적 용법: Factorial
def Factorial(num) :
    res = 1
    for i in range(1, num+1) :
        res = res * i
    return  res

if __name__ == "__main__" :
    num = int(input('임의의 정수: '))
    print(f'Factorial {num}: {Factorial(num)}')
```




```
# 재귀적 용법: Factorial
def Factorial(num) :
    if num==0 :                 # 재귀 함수 탈출(종료) 조건
        return 1
    return num * Factorial(num-1)
```

재귀 함수

피보나치 수열

피보나치 수열 (1/3)

- **피보나치 수열**
  - **피보나치(Fibonacci)**
    - 1,200년 경에 활동한 이탈리아 수학자

"아주 간단한 문제지만...
동적 프로그래밍의 동기와 구현이 다 포함되어 있다."

1, 1, 2, 3, 5 ~

"토끼 한 마리가 매년 새끼 한 마리를 낳는다.
새끼는 한 달 후부터 새끼를 낳기 시작한다.
최초 토끼 한 마리가 있다고 하면...
한 달 후에 토끼는 두 마리가 되고 두 달 후에는 세 마리가 되고..."

$$f_n = f_{n-1} + f_{n-2} (n \geq 3)$$
$$f_1 = f_2 = 1 (n = 1,2)$$

피보나치 수열 (2/3)

- **피보나치 수열:** 재귀적 용법

```
Fibonacci(num)
{
        if (num = 1 or num = 2)
                then  return  1;
        else
                return (Fibonacci(num - 1) + Fibonacci(num - 2));
}
```

"엄청난 중복 호출이 존재한다."

--> 재귀적 알고리즘은 지수함수에 비례하는 시간이 든다.

피보나치 수열 (3/3)

연습문제 2-4: 피보나치 수열 -- 재귀적 용법 | C

```c
#include <stdio.h>
int Fibo(int num);

int main(void)
{
        int     num;

        printf("### 피보나치 수열 구하기 ### \n\n");
        printf("몇 번째 수열까지 출력할까요: ");
        scanf_s("%d", &num);  // scanf ("%d", &num);

        for( int i=1; i<=num; i++ ) {
            if(i%5)             printf("%8d", Fibo(i) );
            else                printf("%8d\n", Fibo(i) );
        }
        printf("\n");
        return 0;
}
```

```
Microsoft Visual Studio 디버그 콘솔
### 피보나치 수열 구하기 ###

몇 번째 수열까지 출력할까요: 35
       1       1       2       3       5
       8      13      21      34      55
      89     144     233     377     610
     987    1597    2584    4181    6765
   10946   17711   28657   46368   75025
  121393  196418  317811  514229  832040
 1346269 2178309 3524578 5702887 9227465

C:₩Users₩click₩OneDrive₩문서₩c Clickseo₩x64₩
이 창을 닫으려면 아무 키나 누르세요...
```

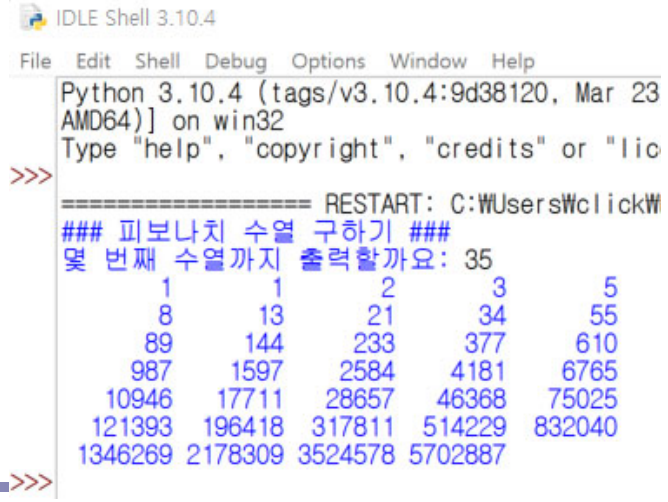
```c
// 재귀적 용법: 피보나치 수열
int Fibo(int num) {
        // 재귀 호출: 탈출 조건
        if(num == 1 || num == 2)
                return 1;
        return Fibo(num - 1) + Fibo(num - 2);
}
```

반복적.재귀적 용법: 피보나치 수열 (3/3)

연습문제 2-5: 피보나치 수열 -- 재귀적 용법 | Python

```
# 함수 정의: 재귀적 용법
def Fibo(num) :
    if num==1 or num==2 :
        return 1
    return Fibo(num-1) + Fibo(num-2)
```

```
if __name__ == '__main__' :
    print('### 피보나치 수열 구하기 ###')
    num = int(input('몇 번째 수열까지 출력할까요: '))
    for i in range(1, num) :
        if i%5 : print(f'{Fibo(i):8}', end='')
        else : print(f'{Fibo(i):8}')
```



동적 프로그래밍

- 재귀 호출

- **동적 프로그래밍**

  - 피보나치 수열

동적 프로그래밍 (1/2)

- **재귀적 해법**
  - 큰 문제에 닮음 꼴의 작은 문제가 깃든다.
  - **잘 쓰면 보약, 잘못 쓰면 맹독**
    - 관계중심으로 파악함으로써 문제를 간명하게 볼 수 있다.
    - **재귀적 해법을 사용하면 심한 중복 호출이 일어나는 경우가 있다.**
  - **재귀적 해법이 바람직한 예**
    - **계승(factorial)** 구하기
    - 퀵 정렬, 병합 정렬 등의 정렬 알고리즘
    - 그래프의 깊이 우선 탐색(DFS, Depth First Search)
  - **재귀적 해법이 치명적인 예**
    - **피보나치 수** 구하기
    - 행렬 곱셈 최적순서 구하기

동적 프로그래밍 (2/2)

- **동적 프로그래밍의 적용 조건**
  - **최적 부분구조(optimal substructure)**
    - 큰 문제의 최적 솔루션에 작은 문제의 최적 솔루션이 포함된다.
  - **재귀 호출 시 중복(overlapping recursive calls)**
    - 재귀적으로 구현했을 때 중복 호출로 심각한 비효율이 발생한다.

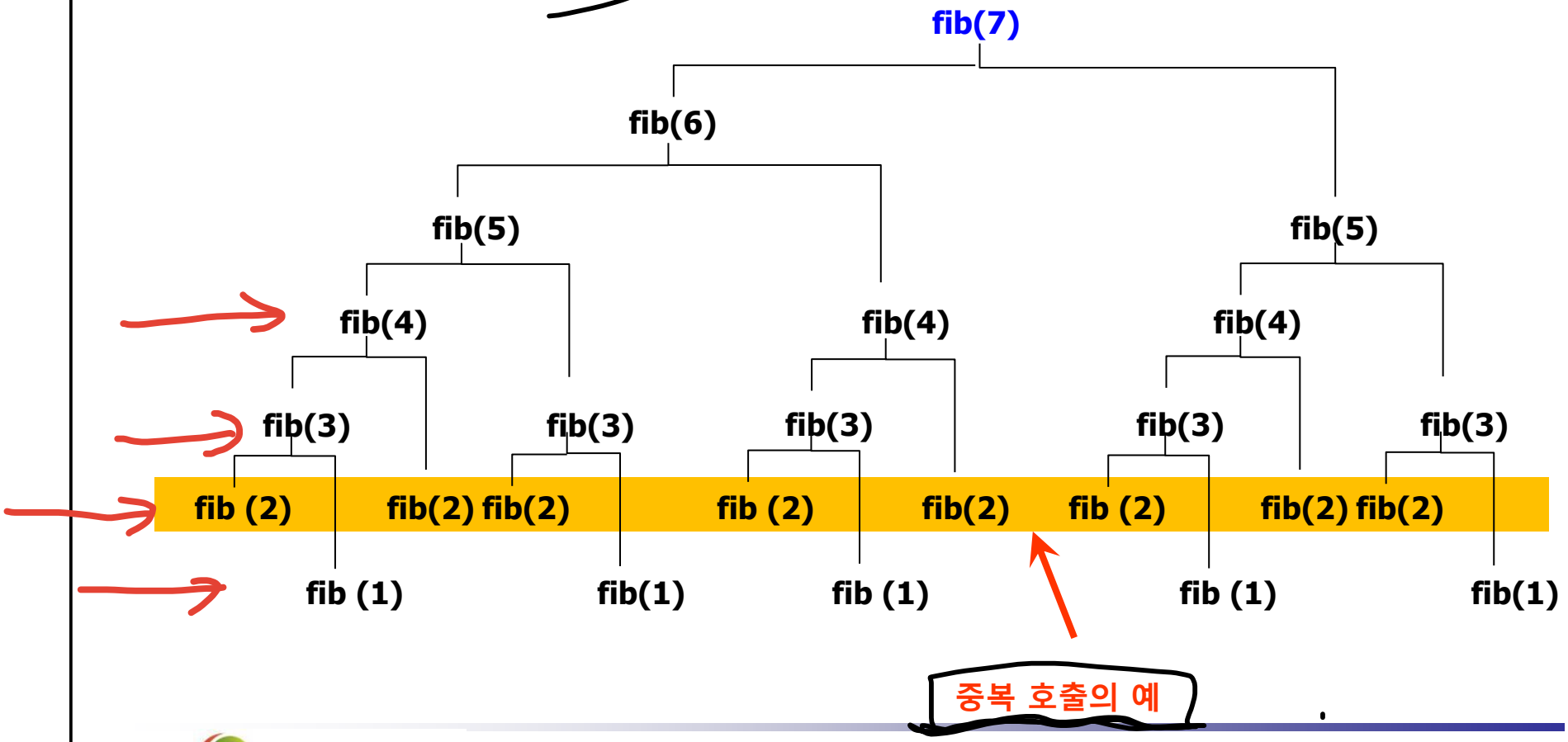
동적 프로그래밍이 그 해결책 !!!

동적 프로그래밍

피보나치 수열

피보나치 수열 (1/2)

- **피보나치 수열:** 재귀적 용법
  - 문제점: 중복 호출!!!



피보나치 수열 (2/2)

- **피보나치 수열:** 동적 프로그래밍 알고리즘

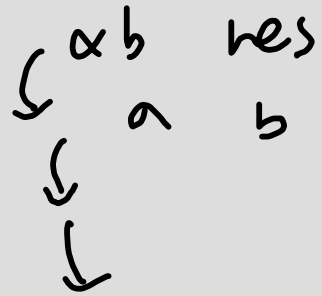
```
Fibonacci(n)
{
    f[1] ← f[2] ← 1;

    for i ← 3 to n
        f[i] ← f[i - 1] + f[i - 2];

    return f[n];
}
```

fibo

1	1	?	?	?	?	?



$\Theta(n)$

참고문헌

[1] Michael T. Goodrich 외 2인 지음, 김유성 외 2인 옮김, “C++로 구현하는 자료구조와 알고리즘”, 한티에듀, 2020.

[2] 주우석, “IT CookBook, C・C++ 로 배우는 자료구조론”, 한빛아카데미, 2019.

[3] 이지영, “C 로 배우는 쉬운 자료구조”, 한빛아카데미, 2022.

[4] 문병로, “IT CookBook, 쉽게 배우는 알고리즘: 관계 중심의 사고법”(개정판), 개정판, 한빛아카데미, 2018.

[5] Richard E. Neapolitan, 도경구 역, “알고리즘 기초”, 도서출판 홍릉, 2017.

[6] “프로그래밍 대회 공략을 위한 알고리즘과 자료 구조 입문”, 와타노베 유타카 저, 윤인성 역, 인사이트, 2021.

[7] “IT CookBook, 쉽게 배우는 자료구조 with 파이썬”, 문병로, 한빛아카데미, 2022.

[8] “이것이 취업을 위한 코딩 테스트다 with 파이썬”, 나동빈, 한빛미디어, 2020.