

자료구조 및 알고리즘

for(A;B;C)
D;

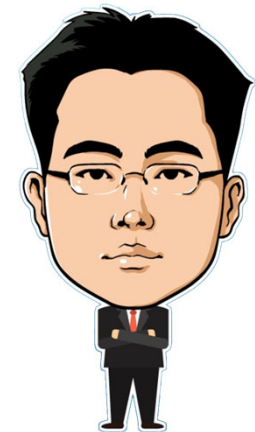


재귀
(Recursion)

Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



목 차



백문이불여일타(百聞而不如一打)

- 재귀 함수

- 피보나치 수열



재귀 함수



백문이불여일타(百聞而不如一打)

- 재귀 함수

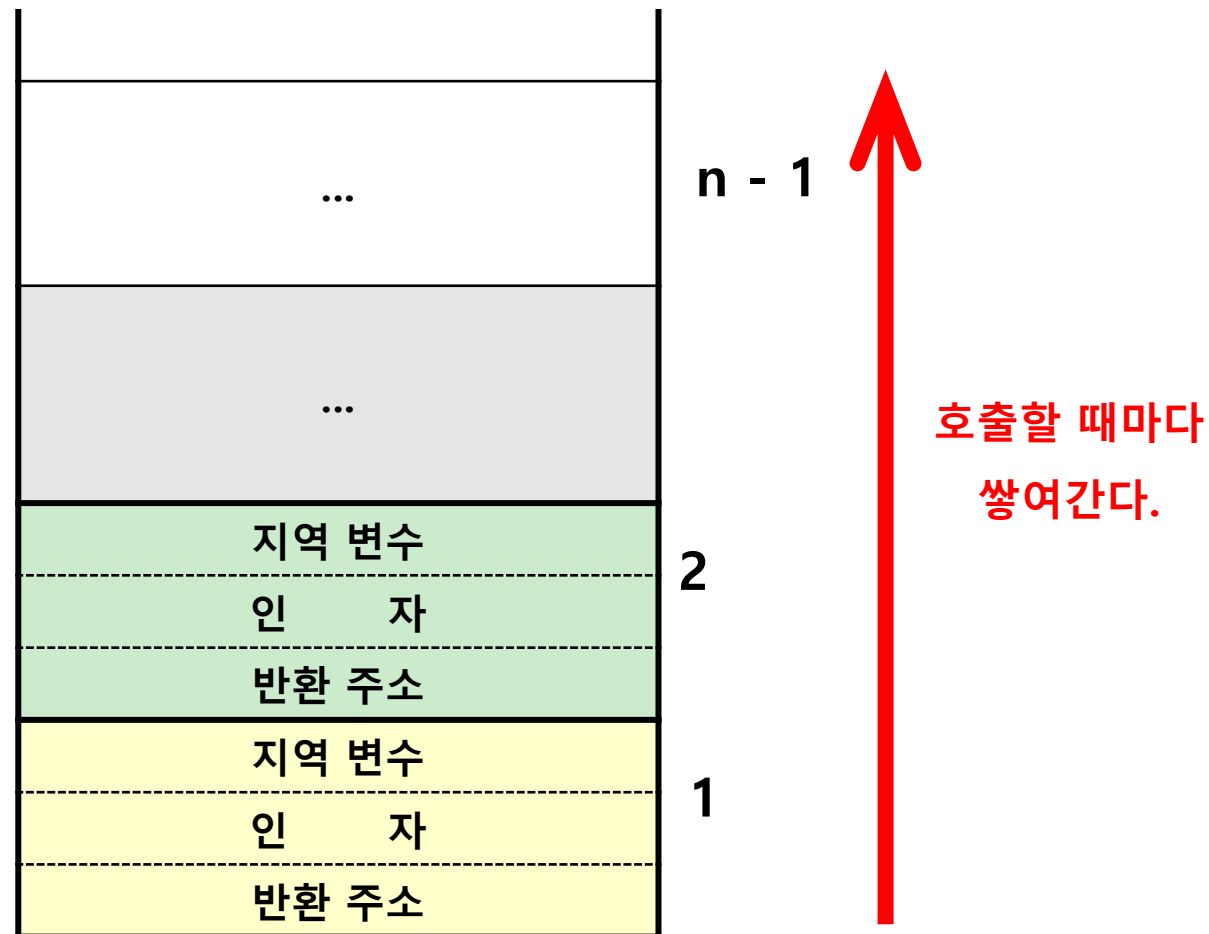
- 반복적.재귀적 용법

- 피보나치 수열



재귀 함수 (1/4)

- 함수 호출 시 동작 과정: 스택 구조



재귀 함수 (2/4)

● 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 (4/4)

예제 2-1: 재귀 함수

| Python

```
# 재귀 함수
def OUTPUT(num: int) -> None:
    print(f'level {num} ')
    # 재귀 함수 탈출(종료) 조건
    if num < 4 :
        OUTPUT(num+1)
    print(f'LEVEL {num} ')

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

```
IDLE Shell 3.11.2
File Edit Shell Debug O
Python 3.11.2 (tag
Type "help", "copy
>>>
=====
level 1
level 2
level 3
level 4
LEVEL 4
LEVEL 3
LEVEL 2
LEVEL 1
>>>
```



재귀 함수 (3/4)

예제 2-1: 재귀 함수

| C/C++

```
#include <iostream>
using namespace std;
```

```
void OUTPUT(int num);
```

```
int main(void)
```

```
{
```

```
    OUTPUT(1);
```

```
    return 0;
```

```
}
```

```
// 재귀 함수
```

```
void OUTPUT(int num) {
```

```
    cout << "level " << num << endl;
```

```
    // 재귀 함수 탈출(종료) 조건
```

```
    if (num < 4)
```

```
        OUTPUT(num + 1);
```

```
    cout << "LEVEL " << num << endl;
```

```
}
```

```
// #include <stdio.h>
```

Microsoft Visual Studio 디버그

level 1

level 2

level 3

level 4

LEVEL 4

LEVEL 3

LEVEL 2

LEVEL 1

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...

```
// printf("level %d\n", num);
```

```
// printf("LEVEL %d\n", num);
```

재귀 함수

반복적.재귀적 용법



반복적.재귀적 용법 (1/4)

● 재귀적 해법

- 큰 문제에 닮은 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

반복적.재귀적 용법 (2/4)

● 계승(Factorial) 구하기

○ 반복적 정의

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

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

○ 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 (4/4)

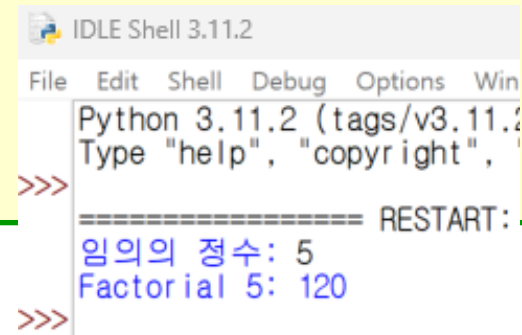
예제 2-3: 계승(Factorial) 구하기

| Python

계승(Factorial) 구하기: 재귀적 용법

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

```
if __name__ == '__main__' :  
    num = int(input('임의의 정수: '))  
    print(f'Factorial {num}: {Factorial(num)}')  
    # import math  
    # print(f'Factorial {num}: {math.factorial(num)}')
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Win  
Python 3.11.2 (tags/v3.11.2  
Type "help", "copyright",  
>>> ===== RESTART:  
임의의 정수: 5  
Factorial 5: 120  
>>>
```

계승(Factorial) 구하기: 반복적 용법

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



반복적.재귀적 용법 (3/4)

예제 2-3: 계승(Factorial) 구하기

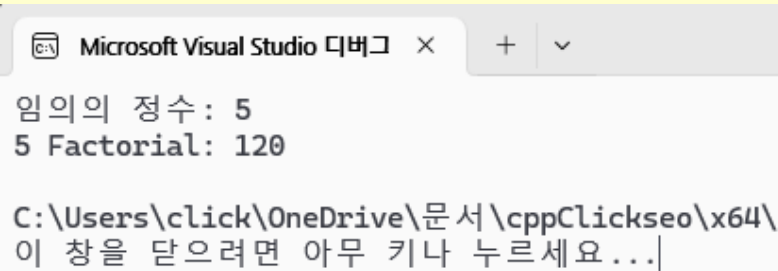
| C/C++

```
#include <iostream>                                     // #include <stdio.h>
using namespace std;
int Factorial(int num);
int main(void)
{
    int num;
    cout << "임의의 정수: ";
    cin >> num;

    cout << num << " Factorial: " << Factorial(num) << endl;
    // printf("임의의 정수: ");
    // scanf_s("%d", &num);
    // scanf("%d", &num);
    // printf("%d Factorial: %d \n", num, Factorial(num) );
    return 0;
}
```

```
// 계승(Factorial) 구하기: 재귀적 용법
int Factorial(int num) {
    if (num == 0) // 재귀함수 탈출(종료) 조건
        return 1;
    return num * Factorial(num - 1);
}
```

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



Microsoft Visual Studio 디버그 × + ▾

임의의 정수: 5
5 Factorial: 120

C:\Users\click\OneDrive\문서\cppClickseo\x64\I
이 창을 닫으려면 아무 키나 누르세요...

재귀 함수

반복적.재귀적 용법: 알고리즘 분석



반복적.재귀적 용법: 알고리즘 분석 (1/3)

- **수학적 알고리즘: 1부터 10까지의 합 구하기**

- 1부터 10까지의 합을 구하는 문제를 해결하는 세 가지 알고리즘

1. 1부터 10까지의 숫자를 직접 하나씩 더한다.

$$1 + 2 + 3 + \dots + 10 = 55$$

2. 두수의 합이 10이 되도록 숫자들을 그룹화하여, 그룹의 계수에 10을 곱하고 남은 숫자 5를 더한다.

$$(0 + 10) + (1 + 9) + (2 + 8) + (3 + 7) + (4 + 6) + 5 = 10 \times 5 + 5 = 55$$

3. 공식을 이용하여 계산할 수도 있다.

$$10 \times (1 + 10) / 2 = 55$$

최선의 알고리즘은 어떻게 찾을 것인가?

반복적.재귀적 용법: 알고리즘 분석 (3/3)

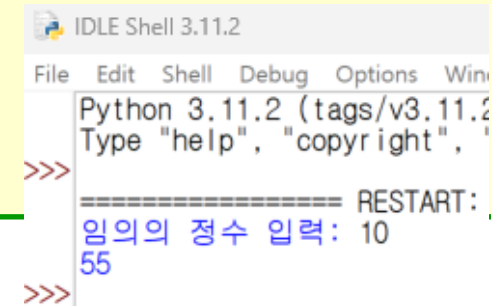
예제 2-2: 1부터 N까지의 합

| Python

1부터 N까지의 합: 재귀적 용법

```
def SUM(num:int) -> int:  
    if(num < 1): return 0  
    return num + SUM(num-1)
```

```
if __name__ == '__main__':  
    num = int(input('임의의 정수 입력: '))  
    print(f'{SUM(num)}', end='')
```



1부터 N까지의 합: 반복적 용법

```
def SUM(num: int) -> int:  
    tot = 0  
    for i in range(1, num+1):  
        tot += i
```

```
    return tot          # return sum(range(1, num+1))    # O(n)
```

```
    # return num * (num+1) // 2    # O(1)
```

알고리즘 분석

$$sum(n) = 1 + 2 + 3 + \dots + (n-1) + n$$

반복적.재귀적 용법: 알고리즘 분석 (2/3)

예제 2-2: 1부터 N까지의 합

| C/C++

```
#include <iostream>
using namespace std;
int SUM(int num);
int main(void)
{
    int num;
    cout << "임의의 정수 입력: ";
    cin >> num;

    cout << "1부터 " << num << "까지의 합: " << SUM(num) << endl;
    // printf("1부터 %d까지의 합: %d \n", num, SUM(num));
    return 0;
}
```

```
// #include <stdio.h>
```

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

```
// 1부터 N까지의 합: 재귀적 용법
int SUM(int num) {
    if (num < 0)
        return 0;
    return num + SUM(num - 1);
}
```

Microsoft Visual Studio 디버그 x + v

임의의 정수 입력: 10
1부터 10까지의 합: 55
1부터 10까지의 합: 55

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...

$$sum(n) = 1 + 2 + 3 + \dots + (n - 1) + n$$

```
// 1부터 N까지의 합: 반복적 용법
int SUM(int num) {
    int tot = 0;
    for (int i = 1; i <= num; ++i)
        tot += i;
    return tot;
    // return num * (num + 1) / 2;
}
```

// $O(n)$ 알고리즘 분석
// $O(1)$

피보나치 수열



백문이불여일타(百聞而不如一打)

- 재귀 함수

- 피보나치 수열

- 동적 프로그래밍

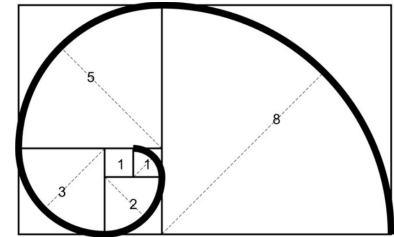


피보나치 수열 (1/4)

● 피보나치 수열(Fibonacci Sequence)

○ 피보나치(Fibonacci)

- 1,200년 경에 활동한 이탈리아 수학자



“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...”

// 재귀적 용법: 피보나치 수열

Fibonacci(num)

{

if (num = 1 or num = 2)
then return 1;

else

return (**Fibonacci**(num - 1) + **Fibonacci**(num - 2));

}

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

$$f_1 = f_2 = 1 (n = 1, 2)$$

“아주 간단한 문제지만...

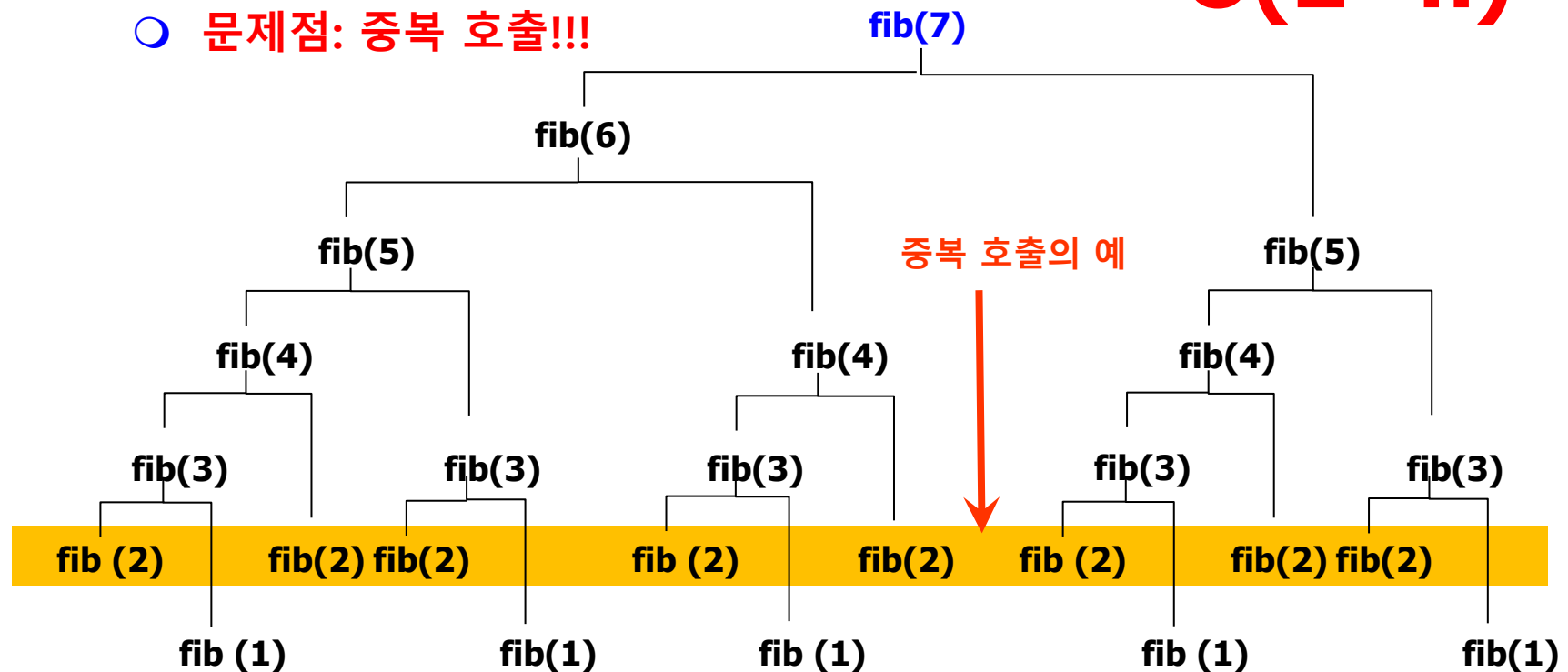
동적 프로그래밍의 동기와 구현이 다 포함되어 있다.”

피보나치 수열 (2/4)

- 피보나치 수열: 재귀적 용법

$O(2^n)$

- 문제점: 중복 호출!!!



“엄청난 중복 호출이 존재한다.”

--> 재귀적 알고리즘은 **지수함수에 비례**하는 시간이 든다.

피보나치 수열 (4/4)

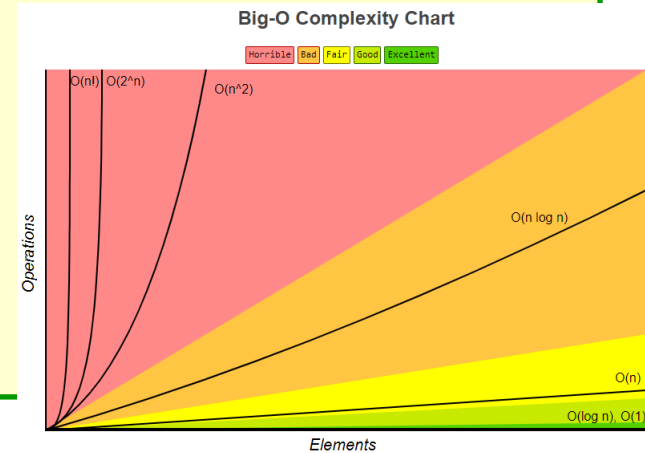
예제 2-4: 피보나치 수열 -- 재귀적 용법

| Python

피보나치 수열: 재귀적 용법

```
def Fibo(num: int) -> int :  
    # 재귀 함수: 탈출 조건  
    if num==1 or num==2 :  
        return 1  
    return Fibo(num-1) + Fibo(num-2)  # O(2^n)
```

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



IDLE Shell 3.11.2

File Edit Shell Debug Options Window Help

Python 3.11.2 (tags/v3.11.2:878ead1, Feb Type "help", "copyright", "credits" or "li

```
>>> ##### RESTART: C:\Users\Wclick\>>> ### 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>>> 수행시간: 3.1221530437469482초>>>
```

피보나치 수열 (3/4)

예제 2-4: 피보나치 수열 -- 재귀적 용법 | C/C++

```
#include <iostream> // #include <stdio.h>
using namespace std;
int Fibo(int num);
int main(void)
{
    int    num;

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

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

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

Microsoft Visual Studio 디버거 × + ▾

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\cppClickseo\x64\ 이 창을 닫으려면 아무 키나 누르세요...

Microsoft Visual Studio 디버거 × + ▾

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

35번째 피보나치 수열 계산 시간: 0.198

C:\Users\click\OneDrive\문서\cppClickseo\x64\ 이 창을 닫으려면 아무 키나 누르세요...

피보나치 수열

동적 프로그래밍



피보나치 수열: 동적 프로그래밍 (1/11)

● 동적 프로그래밍의 적용 조건

○ 최적 부분 구조(Optimal Substructure)

- 큰 문제의 해답에 그보다 작은 문제의 해답이 포함되어 있다.
 - 최적 부분 구조를 가진 문제의 경우에는 재귀 호출을 이용하여 문제를 풀 수 있다.

○ 재귀 호출 시 중복(overlapping recursive calls)

- 재귀적으로 구현했을 때 중복 호출로 심각한 비효율이 발생한다.

동적 프로그래밍이 그 해결책 !!!

- 위의 두 성질이 있는 문제에 대해 적절한 저장 방법으로 중복 호출의 비효율을 제거한 것을 동적 프로그래밍이라고 한다.

피보나치 수열: 동적 프로그래밍 (2/11)

- 피보나치 수열: 동적 프로그래밍

Fibonacci(n)

```
{  
    f[1] ← f[2] ← 1;  
    for i ← 3 to n  
        f[i] ← f[i - 1] + f[i - 2];  
  
    return f[n];  
}
```

fibonacci

1	1	?	?	?	?	?
---	---	---	---	---	---	---

$\Theta(n)$

Fibonacci(n)

```
{  
    first ← second ← 1;  
    for i ← 3 to n  
        res ← first + second;  
    return res;  
}
```


피보나치 수열: 동적 프로그래밍 (8/11)

예제 2-5: 피보나치 수열 -- 동적 프로그래밍

(1/3) | Python

피보나치 수열: 동적 프로그래밍(내장 클래스: list)

```
def Fibo(num: int) -> int:
    if num==1 or num==2 :
        return 1
    sList = [1, 1]
    for i in range(2, num) :
        sList.append(sList[i-1] + sList[i-2])
    return sList[i]
```

```
Fibonacci(n)
{
    f[1] ← f[2] ← 1;
    for i ← 3 to n
        f[i] ← f[i-1] + f[i-2];
    return f[n];
}
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>>
===== RESTART: C:\Users\clickw
### 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
수행시간: 0.0986166000366211초
>>>
```

피보나치 수열: 동적 프로그래밍 (9/11)

예제 2-5: 피보나치 수열 -- 동적 프로그래밍

(2/3) | Python

피보나치 수열: 동적 프로그래밍(일반 변수)

```
def Fibo(num: int) -> int:
```

```
    if num==1 or num==2 :
```

```
        return 1
```

```
    first, second = 1, 1
```

```
    for i in range(3, num+1) :
```

```
        first, second = second, first + second
```

```
    return second
```

Fibonacci(n)

{

 first ← second ← 1;

 for i ← 3 to n

 res ← first + second;

 return res;

}

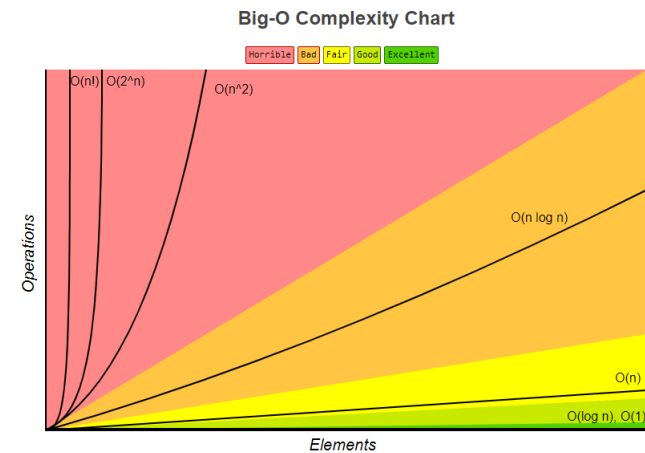
```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>>
===== RESTART: C:\Users\WclickW
### 피보나치 수열 구하기 ###
몇 번째 수열까지 출력할까요: 35
      1      1      2      3      5
      8     13     21     34     55
     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
수행시간: 0.16396713256835938초
>>>
```

피보나치 수열: 동적 프로그래밍 (10/11)

예제 2-5: 피보나치 수열 -- 동적 프로그래밍

(3/3) | Python

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



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  
Type "help", "copyright", "credits" or "li  
>>>  
===== RESTART: C:\Users\WclickW  
### 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  
>>>
```

피보나치 수열: 동적 프로그래밍 (11/11)

● 피보나치 수열: 성능 평가

○ 피보나치 수열

- 각 수는 앞선 두 수의 합인 일련의 수열이다.
- 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

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

$$f_1 = f_2 = 1 (n = 1, 2)$$

- 피보나치 수열을 재귀적 용법과 비 재귀적 용법으로 각각 구현하여, 두 가지 방식의 성능을 비교하였다.

```
IDLE Shell 3.11.2
File Edit Shell Debug Windows Help
Python 3.11.2 (tags/v3.11.2:97addcf, Feb 13 2023)
Type "help()" for help, "copyright()" for copyright, "credits()" for credits or "license()" for license.

>>> ##### RESTART: C:\Users\Wclick\
### 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

수행시간: 3.1221530437469482초
>>>
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Windows Help
Python 3.11.2 (tags/v3.11.2:97addcf, Feb 13 2023)
Type "help()" for help, "copyright()" for copyright, "credits()" for credits or "license()" for license.

>>> ##### RESTART: C:\Users\Wclick\
### 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

수행시간: 0.0986166000366211초
>>>
```

피보나치 수열: 동적 프로그래밍 (3/11)

예제 2-5: 피보나치 수열 -- 동적 프로그래밍

(1/4) | C/C++

```
#include <iostream>                // #include <stdio.h>
using namespace std;              // #include <stdlib.h>
                                  // malloc, calloc, realloc, free

int Fibo(int num);

int main(void)
{
    int    num;

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

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

Microsoft Visual Studio 디버그 × + ▾

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\cppClickseo\x64\ 이 창을 닫으려면 아무 키나 누르세요...

피보나치 수열: 동적 프로그래밍 (4/11)

예제 2-5: 피보나치 수열 -- 동적 프로그래밍

(2/4) | C/C++

```
// 피보나치 수열: 동적 프로그래밍 -- (1) 동적 배열: 동적 메모리 할당
int Fibo(int num){
    // 재귀 함수: 탈출 조건
    if(num == 1 || num == 2)
        return 1;

    // 동적 메모리 할당
    // int* pArr = (int*)realloc(NULL, num * sizeof(int))
    // int* pArr = (int*)malloc(num * sizeof(int));
    // int* pArr = (int*)calloc(num, sizeof(int));
    int* pArr = new int[num];
    if(pArr == nullptr) { // if (pArr == NULL) {
        cout << "동적 메모리 할당 실패!!!" << endl;
        // printf("메모리 할당 실패!!!\n");
        exit(100);
    }
    // pArr[0] = pArr[1] = 1;
    *pArr = *(pArr + 1) = 1;

    int i, temp;
    for(i=2; i < num; ++i)
        *(pArr + i) = *(pArr + i - 1) + *(pArr + i - 2);

    temp = *(pArr + i - 1);
    delete[] pArr; // free(pArr);

    return temp;
}
```

Microsoft Visual Studio 디버그 × + v

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

35번째 피보나치 수열 계산 시간: 0.008

C:\Users\click\OneDrive\문서\cppClickseo\x64\ 이 창을 닫으려면 아무 키나 누르세요...

```
Fibonacci(n)
{
    f[1] ← f[2] ← 1;
    for i ← 3 to n
        f[i] ← f[i-1] + f[i-2];
    return f[n];
}
```

피보나치 수열: 동적 프로그래밍 (5/11)

예제 2-5: 피보나치 수열 -- 동적 프로그래밍

(3/4) | C++

// 피보나치 수열: 동적 프로그래밍 -- (2) 동적 배열: vector 클래스

```
#include <vector>
int Fibo(int num) {
    if (num == 1 || num == 2)
        return 1;

    // 동적 배열: vector 클래스
    vector<int> vArr(num);
    vArr[0] = vArr[1] = 1;

    for (int i = 2; i < num; ++i)
        vArr[i] = vArr[i - 1] + vArr[i - 2];

    return vArr[num - 1];
}
```

Microsoft Visual Studio 디버그

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

35번째 피보나치 수열 계산 시간: 0.009

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...

```
Fibonacci(n)
{
    f[1] ← f[2] ← 1;
    for i ← 3 to n
        f[i] ← f[i-1] + f[i-2];
    return f[n];
}
```

피보나치 수열: 동적 프로그래밍 (6/11)

예제 2-5: 피보나치 수열 -- 동적 프로그래밍

(4/4) | C/C++

// 피보나치 수열: 동적 프로그래밍 -- (3) 일반 변수

```
int Fibo(int num) {  
    if(num == 1 || num == 2)  
        return 1;  
  
    // 일반 변수: first, second, res  
    int first = 1, second = 1, res = 0;  
    for(int i=2; i<num; ++i) {  
        res = first + second;  
        first = second;  
        second = res;  
    }  
    return res;  
}
```

Microsoft Visual Studio 디버그 x + v

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

35번째 피보나치 수열 계산 시간: 0.007

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...

```
Fibonacci(n)  
{  
    first ← second ← 1;  
    for i ← 3 to n  
        res ← first + second;  
    return res;  
}
```


피보나치 수열: 동적 프로그래밍 (7/11)

● 피보나치 수열: 성능 평가

○ 피보나치 수열

- 각 수는 앞선 두 수의 합인 일련의 수열이다. $f_n = f_{n-1} + f_{n-2} (n \geq 3)$
- 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

$$f_1 = f_2 = 1 (n = 1, 2)$$

- 피보나치 수열을 재귀적 용법과 비 재귀적 용법으로 각각 구현하여, 두 가지 방식의 성능을 비교하였다.

```
Microsoft Visual Studio 디버그 x + v
### 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
35번째 피보나치 수열 계산 시간 : 0.198

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

```
Microsoft Visual Studio 디버그 x + v
### 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
35번째 피보나치 수열 계산 시간 : 0.007

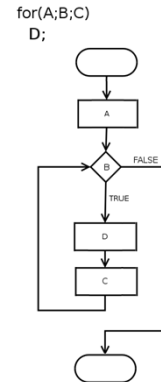
C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

재귀적 용법

비재귀적 용법

참고문헌

- [1] "이것이 자료구조+알고리즘이다: with C 언어", 박상현, 한빛미디어, 2022.
- [2] "C++로 구현하는 자료구조와 알고리즘(2판)", Michael T. Goodrich, 김유성 외 2인 번역, 한빛아카데미, 2020.
- [3] "IT CookBook, 쉽게 배우는 자료구조 with 파이썬", 문병로, 한빛아카데미, 2022.
- [4] 문병로, "IT CookBook, 쉽게 배우는 알고리즘: 관계 중심의 사고법"(3판, 개정판, 한빛아카데미, 2024.
- [5] "코딩 테스트를 위한 자료 구조와 알고리즘 with C++", John Carey 외 2인, 황선규 역, 길벗, 2020.
- [6] "이것이 취업을 위한 코딩 테스트다 with 파이썬", 나동빈, 한빛미디어, 2020.
- [7] "SW Expert Academy", SAMSUNG, 2025 of viewing the site, <https://swexpertacademy.com/>.
- [8] "BAEKJOON", (BOJ) BaekJoon Online Judge, 2025 of viewing the site, <https://www.acmicpc.net/>.
- [9] "programmers", grepp, 2025 of viewing the site, <https://programmers.co.kr/>.
- [10] "goormlevel", goorm, 2025 of viewing the siteh, <https://level.goorm.io/>



이 강의자료는 저작권법에 따라 보호받는 저작물이므로 무단 전제와 무단 복제를 금지하며,
내용의 전부 또는 일부를 이용하려면 반드시 저작권자의 서면 동의를 받아야 합니다.

Copyright © Clickseo.com. All rights reserved.

