

# 遞回 Recursion

by 多拉 A 夢

來看看小時候學的遞回...

小時候的我們...

$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

小時候的我們...

$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_1 = F_0 + 1 = 0 + 1 = 1$$

小時候的我們...

$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_2 = \underbrace{F_1}_{\text{~~~~~}} + 1$$

$$= \underbrace{(F_0 + 1)}_{\text{~~~~~}} + 1 = 2$$

小時候的我們...

$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = F_2 + 1$$

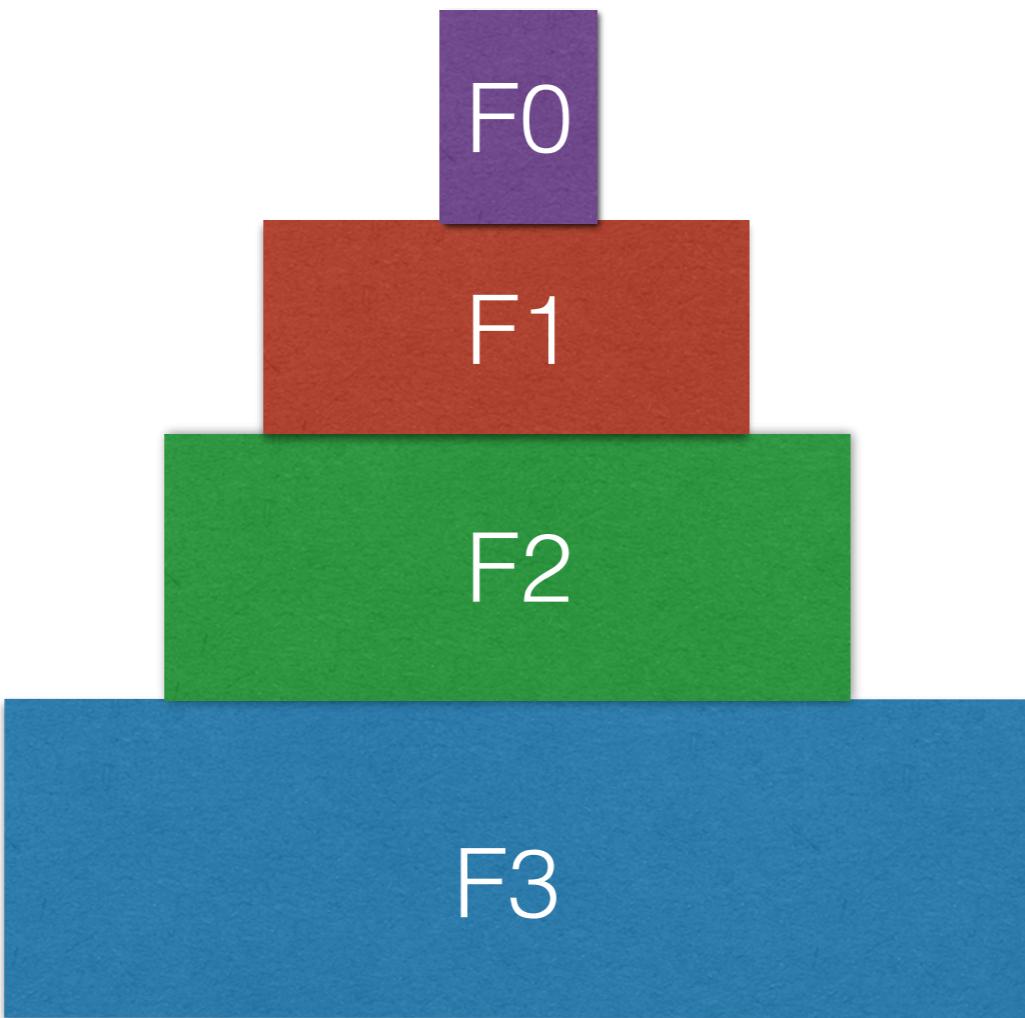
$$= (\cancel{F_1} + 1) + 1$$

$$= ((\cancel{F_0} + 1) + 1) + 1 = 3$$

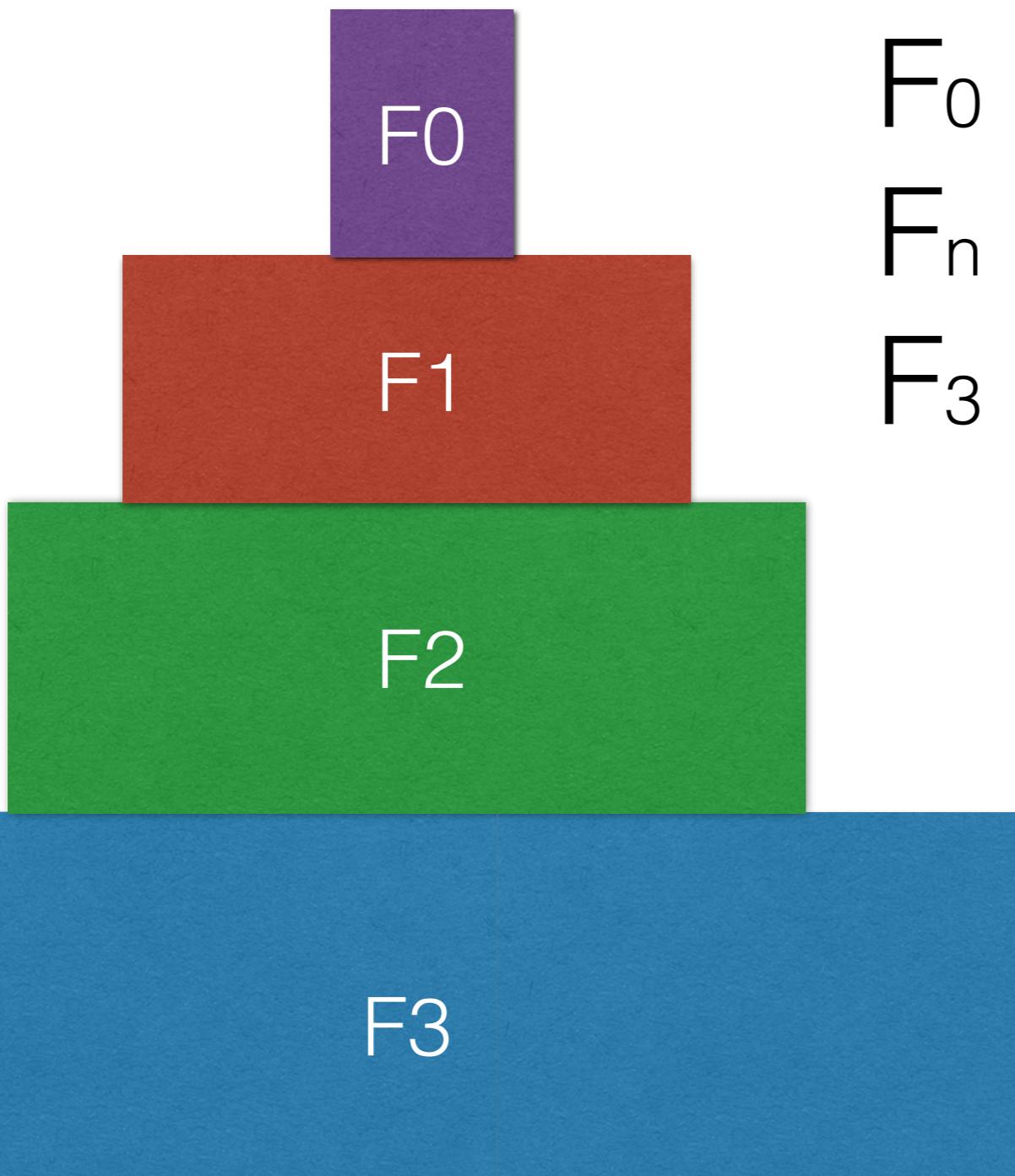
數字太難看？

還是很難想像：D？

讓遞回變成一座塔：D



# 讓遞回變成一座塔：D

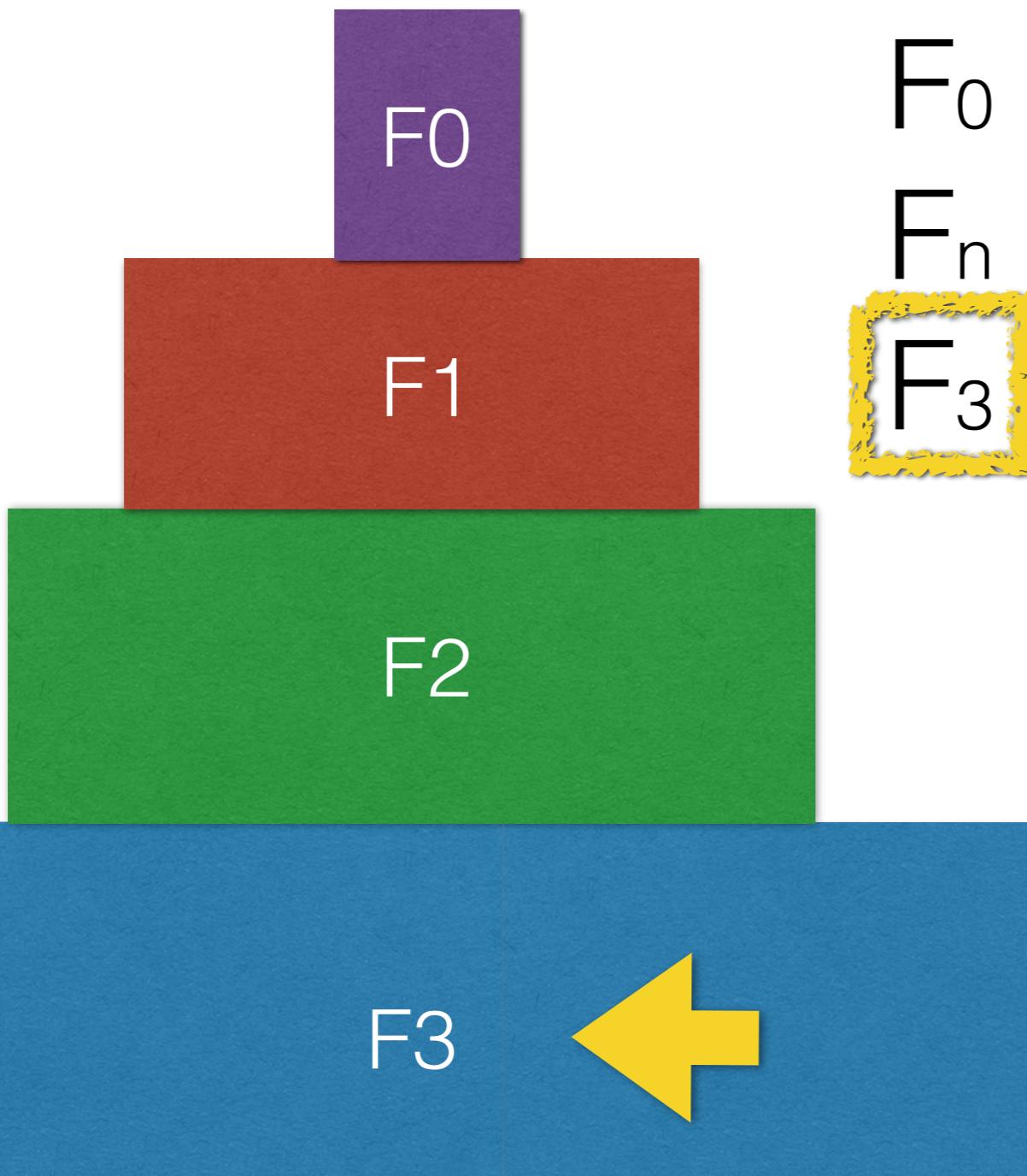


$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = ?$$

# 讓遞回變成一座塔：D

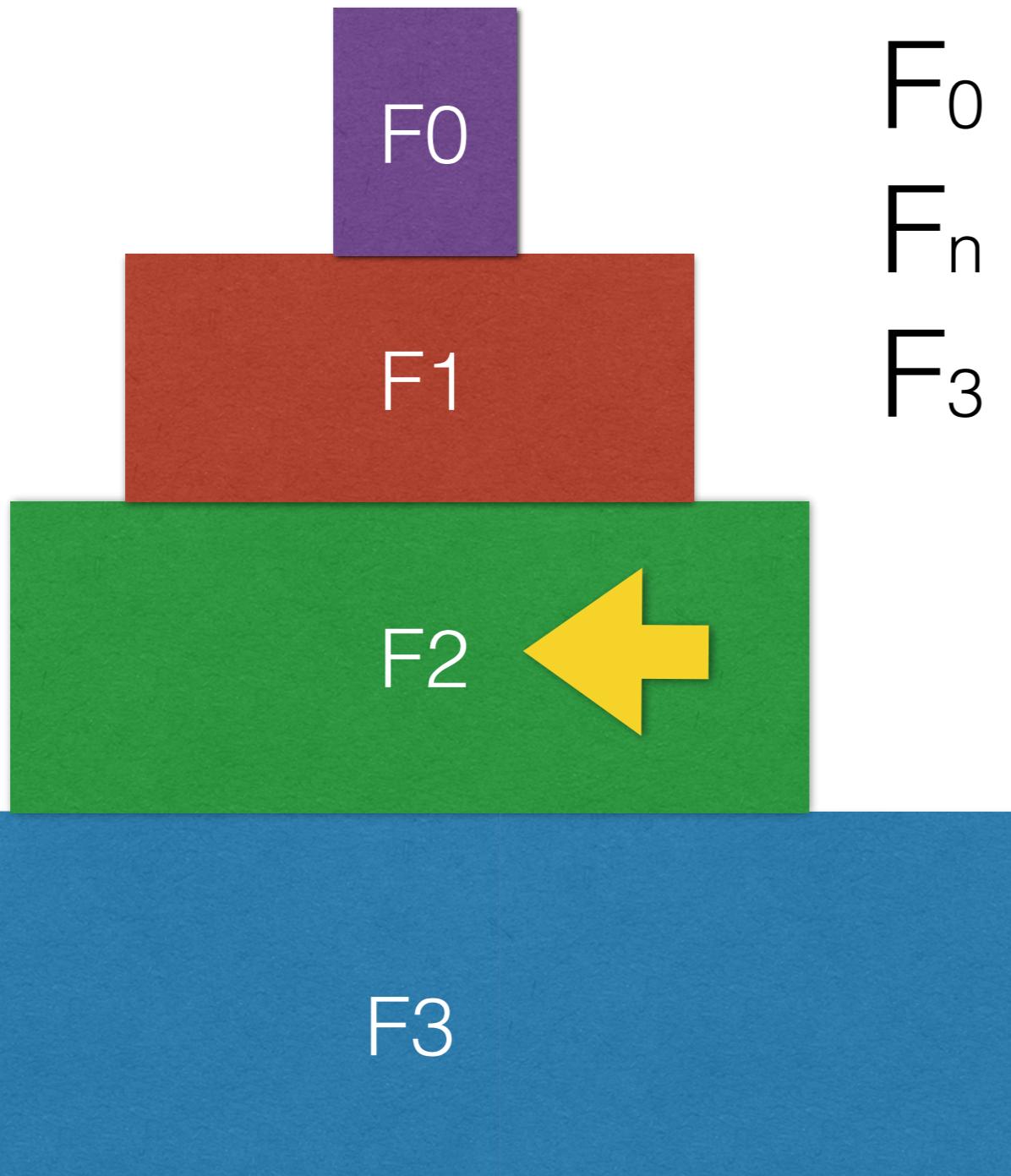


$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = \underline{F_2} + 1$$

# 讓遞回變成一座塔：D



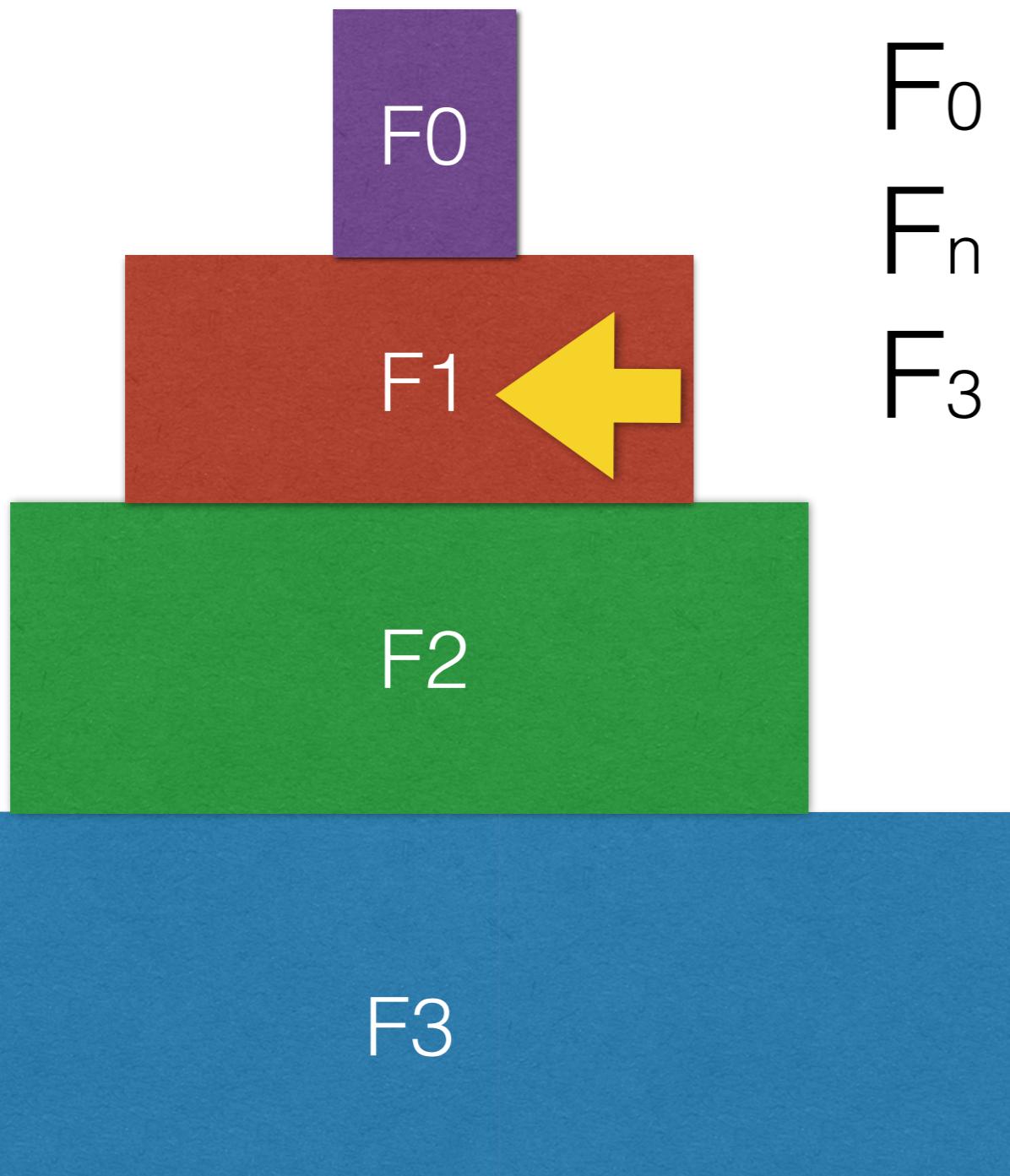
$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = \boxed{F_2} + 1$$

$$\textcolor{red}{F_2} = \underline{F_1} + 1$$

# 讓遞回變成一座塔：D



$$F_0 = 0$$

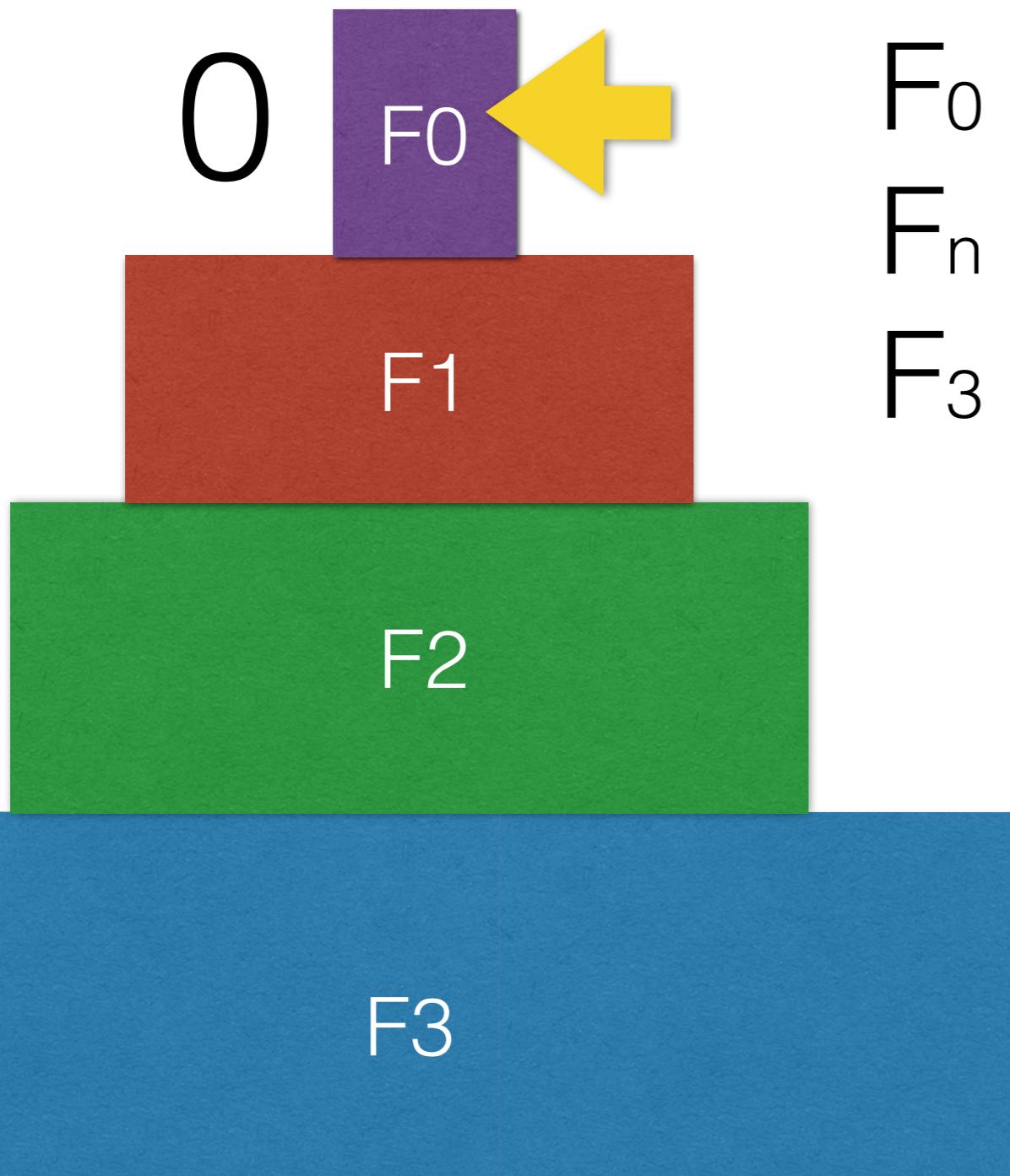
$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = \underbrace{F_2}_{\text{---}} + 1$$

$$\underbrace{F_2}_{\text{---}} = \boxed{F_1} + 1$$

$$\underline{F_1} = \underline{F_0} + 1$$

# 讓遞回變成一座塔：D



$$F_0 = 0$$

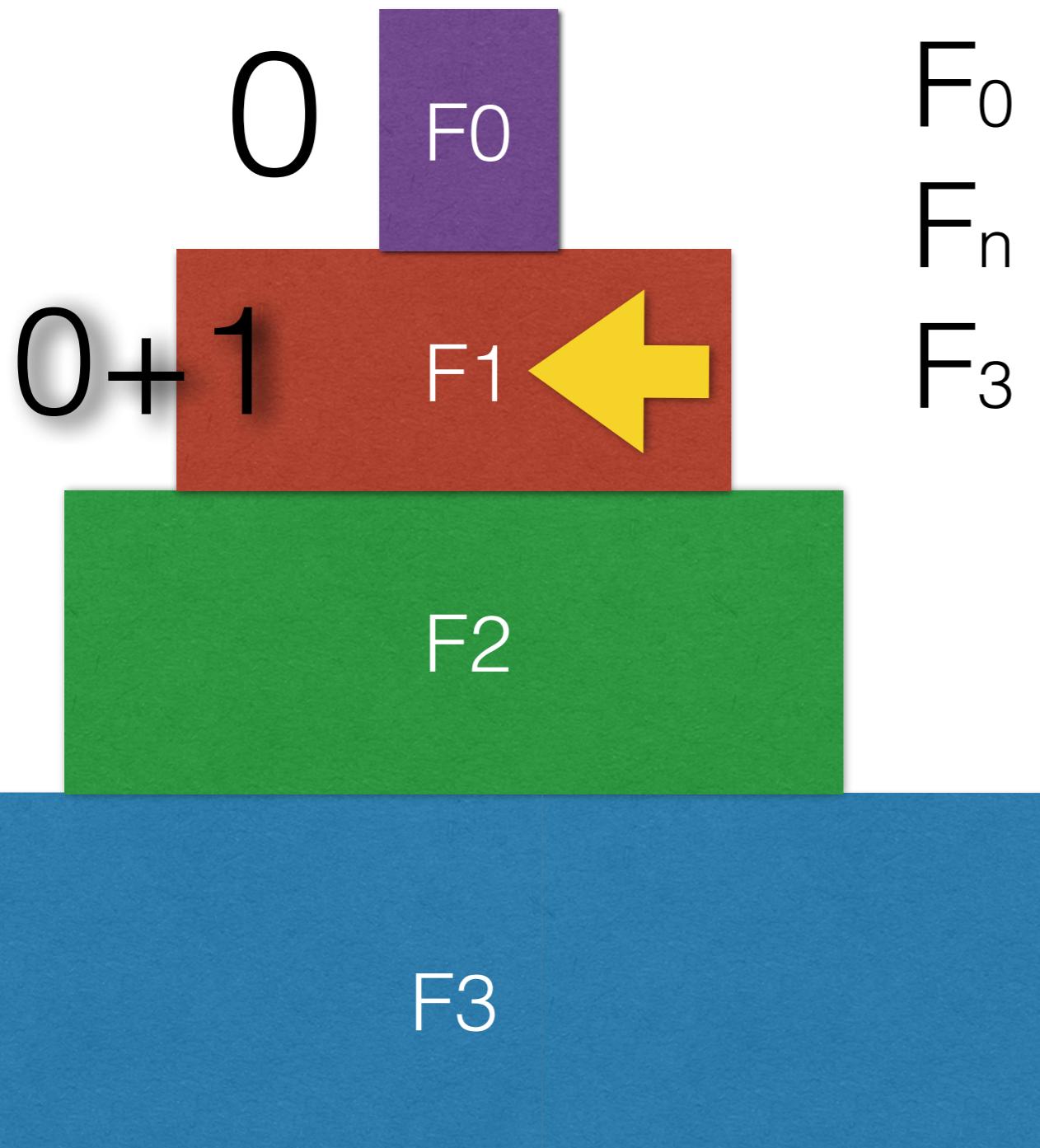
$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = \underbrace{F_2}_{\text{~~~~~}} + 1$$

$$\underbrace{F_2}_{\text{~~~~~}} = \frac{F_1}{\text{~~~~~}} + 1$$

$$\underline{F_1} = \boxed{F_0} + 1$$

# 讓遞回變成一座塔：D



$$F_0 = 0$$

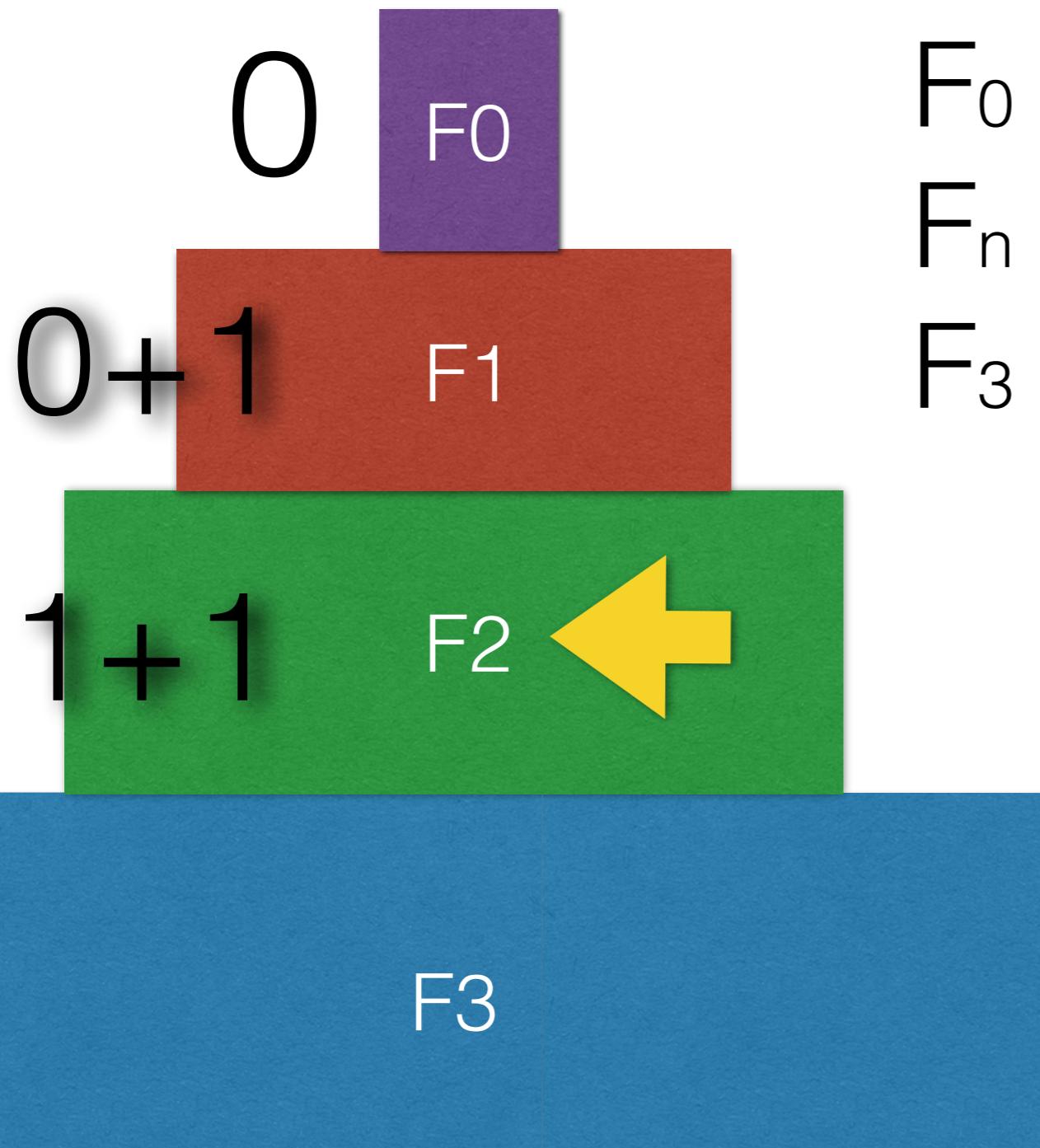
$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = \underbrace{F_2}_{\textcolor{red}{\sim\sim\sim}} + 1$$

$$\textcolor{red}{F_2} = \underbrace{F_1}_{\textcolor{red}{\sim\sim\sim}} + 1$$

$$\boxed{\textcolor{green}{F_1} = \textcolor{green}{0} + 1}$$

# 讓遞回變成一座塔：D



$$F_0 = 0$$

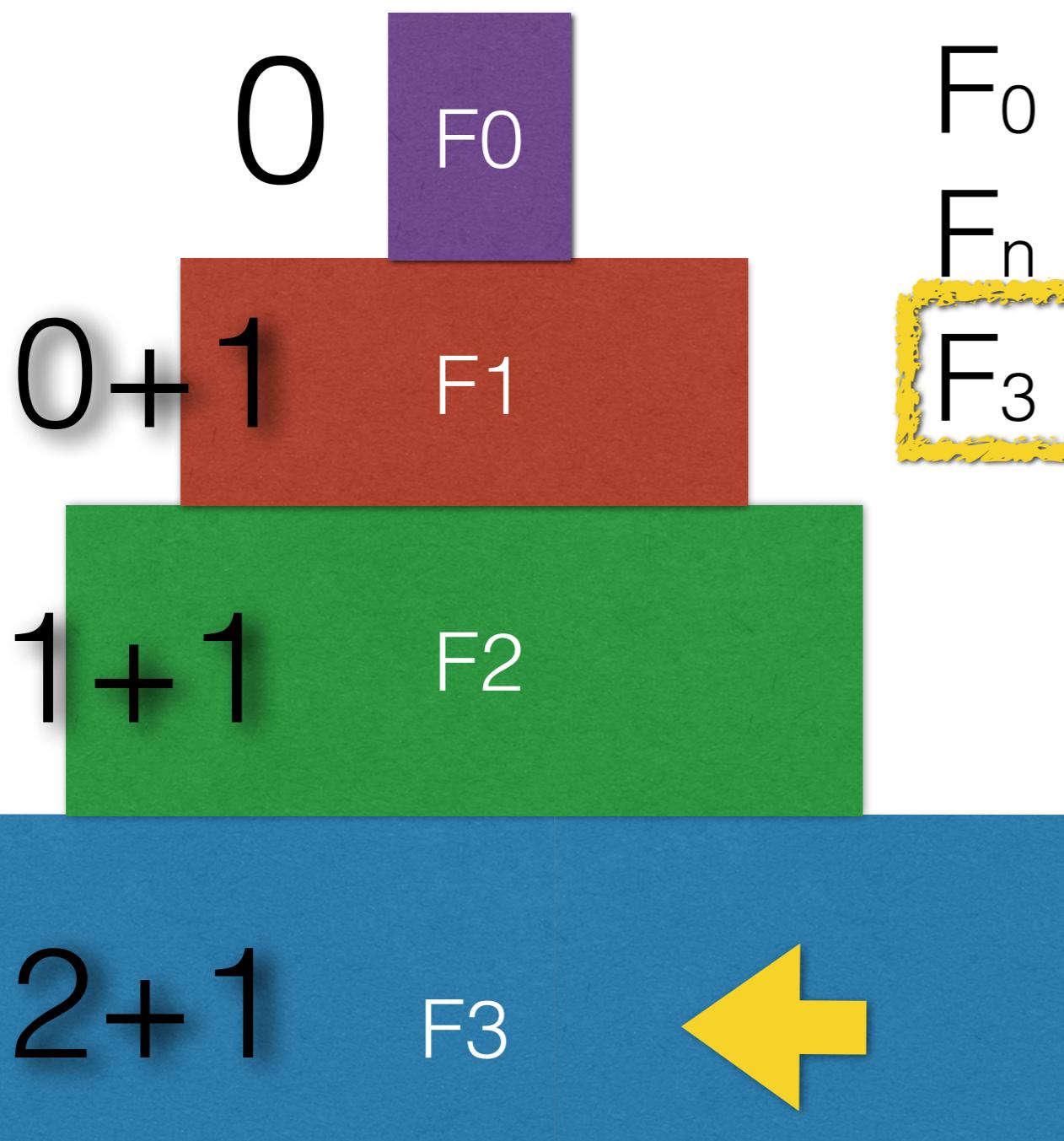
$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = F_2 + 1$$

$$F_2 = 1 + 1$$

$$\underline{F_1 = 0 + 1}$$

# 讓遞回變成一座塔：D



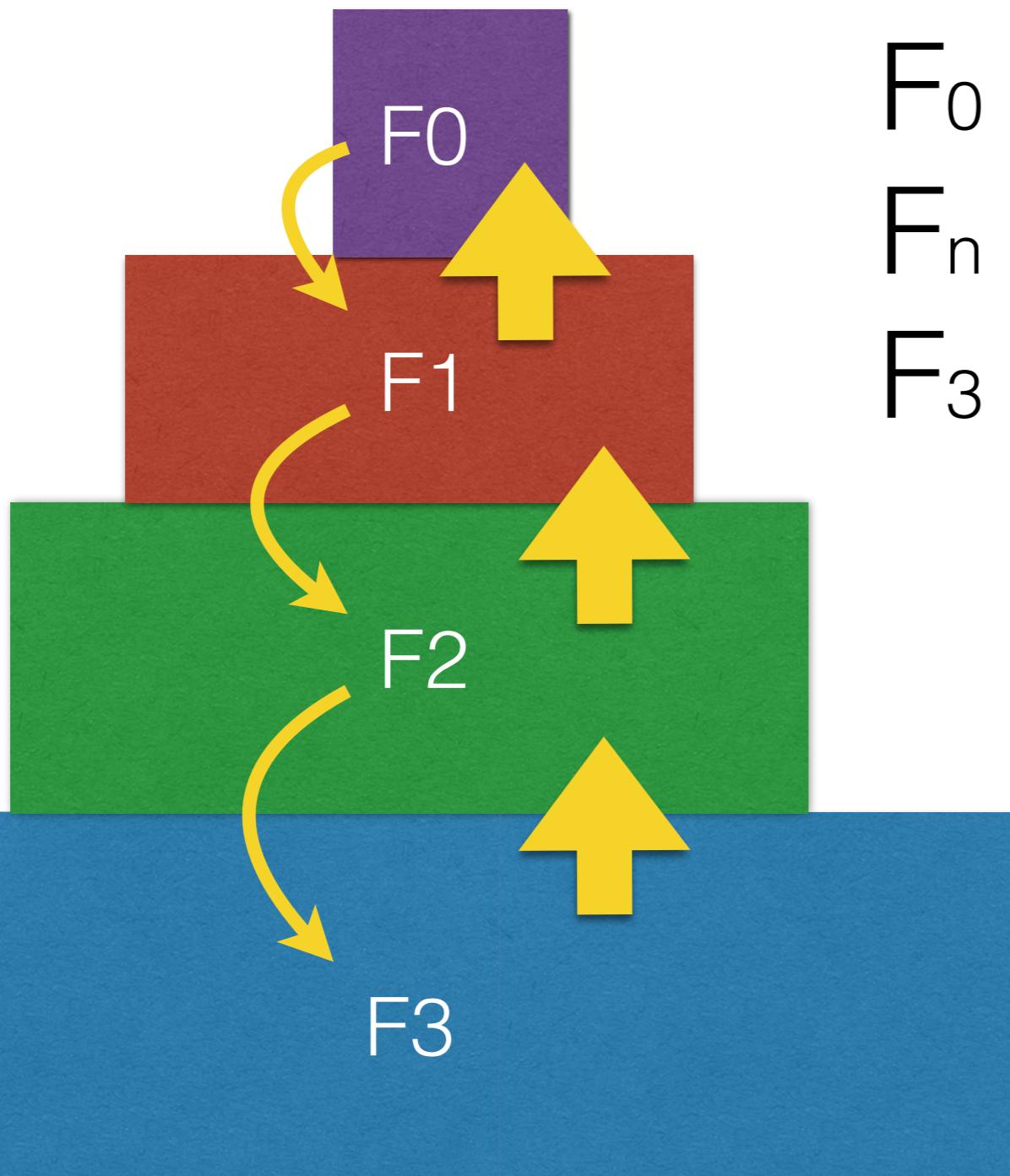
$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = 2 + 1$$

$$\underline{F_2 = 1} + 1$$

# 讓遞回變成一座塔：D



$$F_0 = 0$$

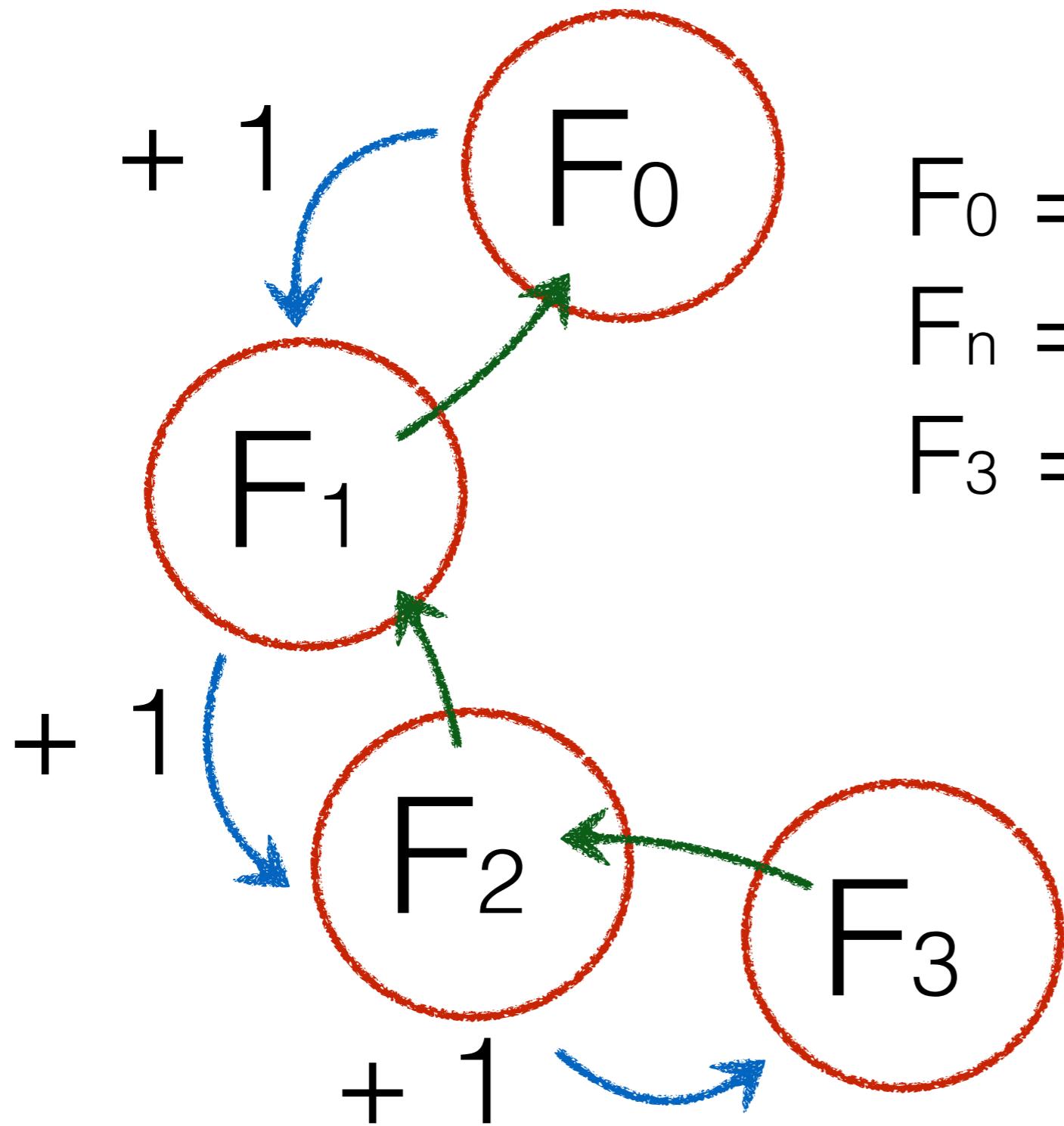
$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = \underbrace{F_2}_{\text{red}} + 1$$

$$\underbrace{F_2}_{\text{red}} = \frac{1}{\text{green}} + 1$$

$$\underline{F_1} = \underline{0} + 1$$

當然你也可以想成一串貢丸...



$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

$$F_3 = \underbrace{F_2}_{\text{red}} + 1$$

$$\underbrace{F_2}_{\text{red}} = \underbrace{1}_{\text{green}} + 1$$

$$\underbrace{F_1}_{\text{green}} = \underbrace{0}_{\text{green}} + 1$$

說了這麼多... 然後咧 X D

怎麼生出程式來：D？

還記得上禮拜學的...

Function ~

# Review - 兩數最大值

```
1 #include <iostream>
2
3 int MAX(int x, int y){
4     if( x > y )
5         return x;
6     return y;
7 }
8
9 int main(){
10    int x, y;
11    std::cin >> x >> y;
12    std::cout << MAX(x, y) << std::endl;
13    return 0;
14 }
```

把遞回式寫成函數

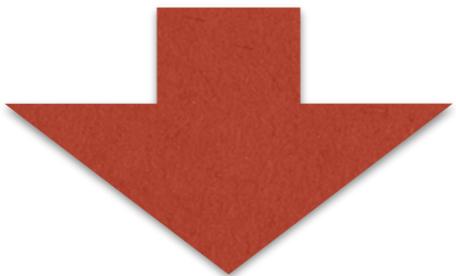
$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$

# 把遞回式寫成函數

$$F_0 = 0$$

$$F_n = F_{n-1} + 1 \quad (n \geq 1)$$



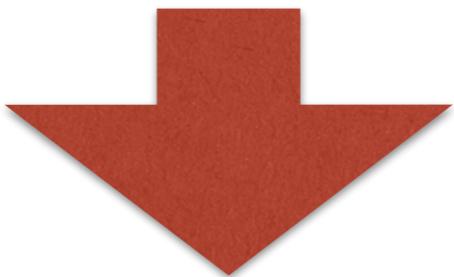
$$F(0) = 0$$

$$F(n) = F(n-1) + 1 \quad (n \geq 1)$$

# 把遞回條件定出來

$$F(0) = 0$$

$$F(n) = F(n-1) + 1 \quad (n \geq 1)$$



```
if( n == 0 ) 回傳 0;
```

其他狀況：

```
return F(n-1) + 1;
```

# 來來來～Code抵家～

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + 1;
7 }
8 ━━━━━━━━
9 int main(){
10    int n;
11    std::cin >> n;
12    std::cout << F(n) << std::endl;
13    return 0;
14 }
```

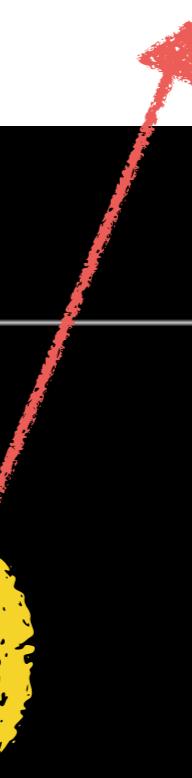
如果要從 $1+2+3+\dots+n$

要怎麼改剛剛的code呢？

(hint: 只要改一個地方)

# 改這裡嘍～

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
9 int main(){
10    int n;
11    std::cin >> n;
12    std::cout << F(n) << std::endl;
13    return 0;
14 }
```



如果還不太理解的話...

列式子：

$$F(0) = 0$$

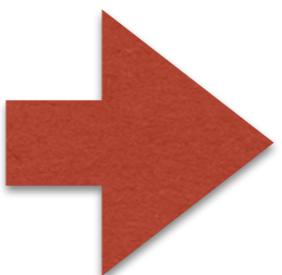
$$F(n) = F(n-1) + n$$

# 如果還不太理解的話...

列式子：

$$F(0) = 0$$

$$F(n) = F(n-1) + n$$



條件定出來：

```
if( n == 0 ) return 0
```

```
else
```

```
    return F(n-1) + n
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
9 int main(){
10    int n;
11    std::cin >> n;
12    std::cout << F(n) << std::endl;
13    return 0;
14 }
```



3

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){ → 3
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
9 int main(){
10    int n;
11    std::cin >> n; → 3
12    std::cout << F(n) << std::endl;
13    return 0;
14 }
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
9 int main(){
10    int n;
11    std::cin >> n;
12    std::cout << F(n) << std::endl;
13    return 0;
14 }
```

The diagram shows a yellow arrow pointing from the highlighted part of the code 'F(n-1)' to the mathematical expression '3 - 1 = 2'. Another yellow arrow points from the highlighted 'n' in the line 'std::cin >> n;' to the value '3'.

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
9 int main(){
10    int n;
11    std::cin >> n;
12    std::cout << F(n) << std::endl;
13    return 0;
14 }
```

The diagram shows a yellow arrow pointing from the highlighted portion of the code `F(n-1)` to the number `3`, indicating the value being returned by that recursive call.

# 直接用Code來思考...

```
2  
3 int F(int n){  
4     if(n == 0)  
5         return 0;  
6     return F(n-1) + n;  
7 }
```

```
1 #include <iostream>  
2  
3 int F(int n){  
4     if(n == 0)  
5         return 0;  
6     return F(n-1) + n;  
7 }  
8
```



# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){ → 2
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
```

The diagram illustrates a step in the recursive calculation of F(2). A yellow arrow points from the highlighted term  $F(n-1)$  in the code to the term  $2 - 1$  in the equation  $2 - 1 = 1$ , which is then simplified to  $= 1$ .

$$2 - 1 = 1$$

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
```

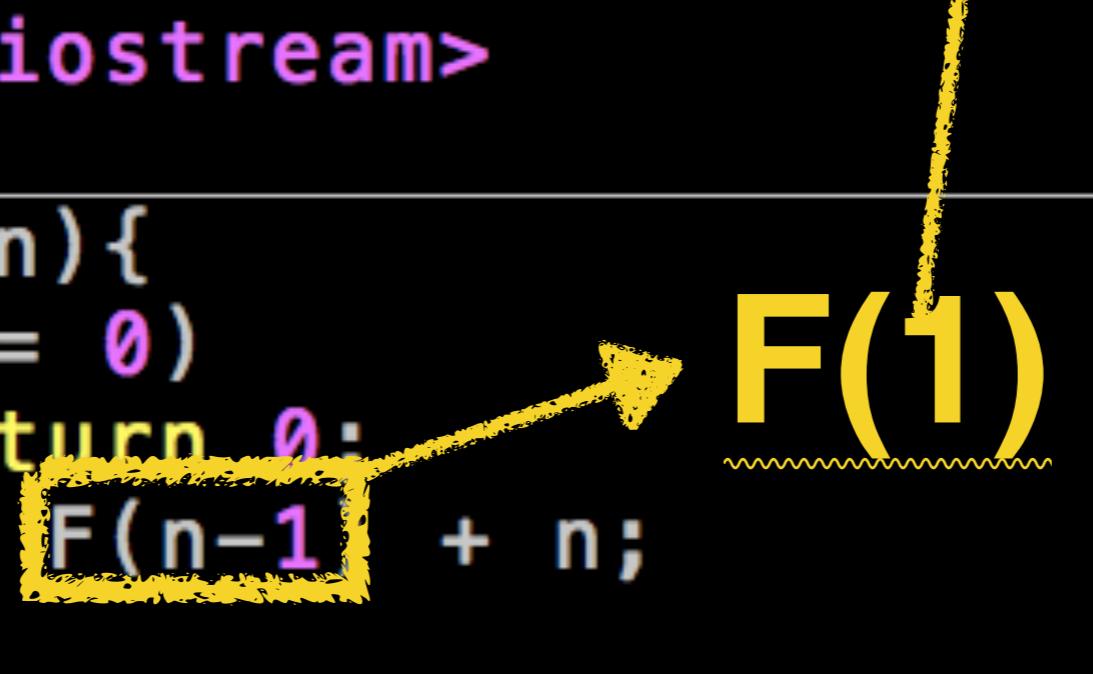
F(1)

F(n-1)

# 直接用Code來思考...

```
2  
3 int F(int n){  
4     if(n == 0)  
5         return 0;  
6     return F(n-1) + n;  
7 }
```

```
1 #include <iostream>  
2  
3 int F(int n){  
4     if(n == 0)  
5         return 0;  
6     return F(n-1) + n;  
7 }
```



# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){ → 1
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
```

A yellow arrow points from the highlighted 'n-1' in the code to the equation '1 - 1 = 0'. The equation is written in large yellow digits.

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
```

F(0)

F(n-1)

+ n;

F(0)

# 直接用Code來思考...

```
2  
3 int F(int n){  
4     if(n == 0)  
5         return 0;  
6     return F(n-1) + n;  
7 }
```

```
1 #include <iostream>
```

```
2  
3 int F(int n){  
4     if(n == 0)  
5         return 0;  
6     return F(n-1) + n;  
7 }
```

F(0)

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){ → 0
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n){ → 0
4     if(n == 0)
5         return 0; ←
6     return F(n-1) + n;
7 }
8
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n) {
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
8
```

```
1 #include <iostream>
2
3 int F(int n) {
4     if(n == 0)
5         return 0;
6     return F(n-1) + n;
7 }
```

$F(0) = 0$

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n) {
4     if(n == 0)      F(0) + 1
5         return 0;
6     return F(n-1) + n;
7 }
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n) {
4     if(n == 0)      F(0) + 1 = 1
5         return 0;
6     return F(n-1) + n;
7 }
```

```
1 #include <iostream>
2
3 int F(int n) {
4     if(n == 0)      2
5         return 0;    F(1) = 1
6     return F(n-1) + n;
7 }
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n) { → 2
4     if(n == 0)      F(1)+2 = 1+2
5         return 0;
6     return F(n-1) + n; → 3
7 }
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n) {  
4     if(n == 0)  
5         return 0;  
6     return F(n-1) + n;  
7 }
```

$F(1)+2 = 1+2$

```
1 #include <iostream>
2
3 int F(int n) {  
4     if(n == 0)  
5         return 0;  
6     return F(n-1) + n;  
7 }
```

$F(2)=3$

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n) {
4     if(n == 0)    F(2)+3=3+3
5         return 0;
6     return F(n-1) + n;
7 }
```

# 直接用Code來思考...

```
1 #include <iostream>
2
3 int F(int n) {
4     if(n == 0)    F(2)+3 = 3+3
5         return 0;
6     return F(n-1) + n;
7 }
```

```
8
9 int main(){
10    int n;          F(3) = 6
11    std::cin >> n;
12    std::cout << F(n) << std::endl;
13    return 0;
14 }
```

呼... 投影片終於做完了

遞回最終哲學 :  $f(f(x))$

遞回最終哲學： $f(f(x))$

自己呼叫自己：D

# 必須要注意的地方

- 遞回必須要設好底線~~~~~  
要不然會爛掉 X D D D D D D
- EX. if( n == 0) return 0;

# 範例：G C D

```
1 #include <iostream>
2
3 int gcd(int a, int b){
4     if(a % b == 0)
5         return b;      輾轉相除法
6     else
7         return gcd(b, a % b);
8 }
9
10 int main(){
11     int a, b;
12     std::cin >> a >> b;
13     std::cout << gcd(a, b) << std::endl;
14     return 0;
15 }
```

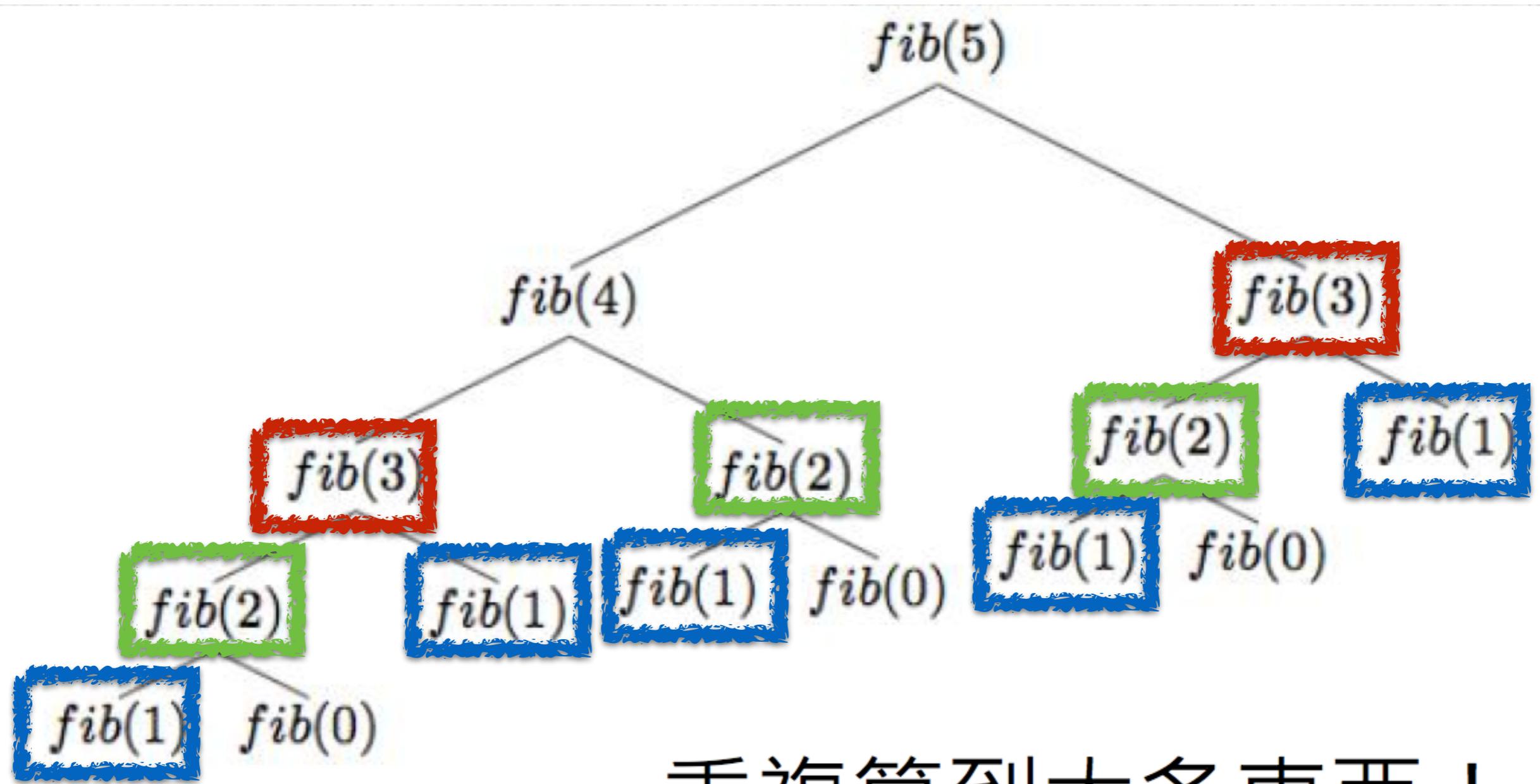
# 練習：TOJ #33

- 費波那契數列
- 提示：列出遞回關係式→改成函數表達式
  - 列出關係式後，注意起點**不止一個**
    - $f(0) = 1$
    - $f(1) = 1$
  - 寫成Code萬歲！！！！

# 練習：TOJ #33

- 請嘗試跑出第100個數字
- 你會發現它跑得非常慢 X D D D D
- 甚至在今天跑不出來 X D

# 練習：TOJ #33



# 練習：TOJ #33

- 你應該會發現到.....
- 輕鬆表達數學遞回
- 但是效率有點差

# 練習：TOJ #33

- 有時候用迴圈跑比較快 X D
- 想一下費式數列怎麼用迴圈跑