遞回 Recursion

by 多拉 A 夢

來看看小時候學的遞回...

小時候的我們...

$F_0 = 0$

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

小時候的我們...

$$F_0 = 0$$

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

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

小時候的我們...

$F_0 = 0$

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

$F_2 = F_1 + 1$

$= (F_0 + 1) + 1 = 2$

小時候的我們...

$$F_0 = 0$$

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

$$F_3 = F_2 + 1$$

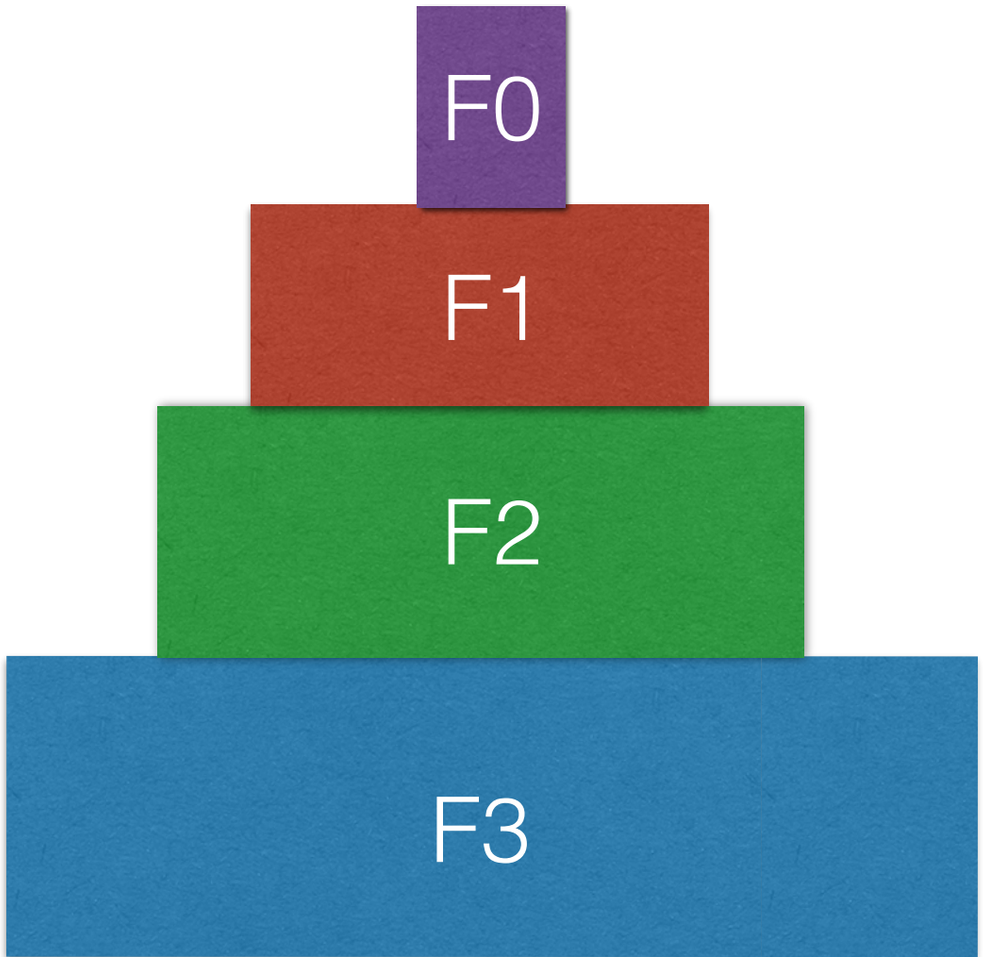
$$= (F_1 + 1) + 1$$

$$= ((F_0 + 1) + 1) + 1 = 3$$

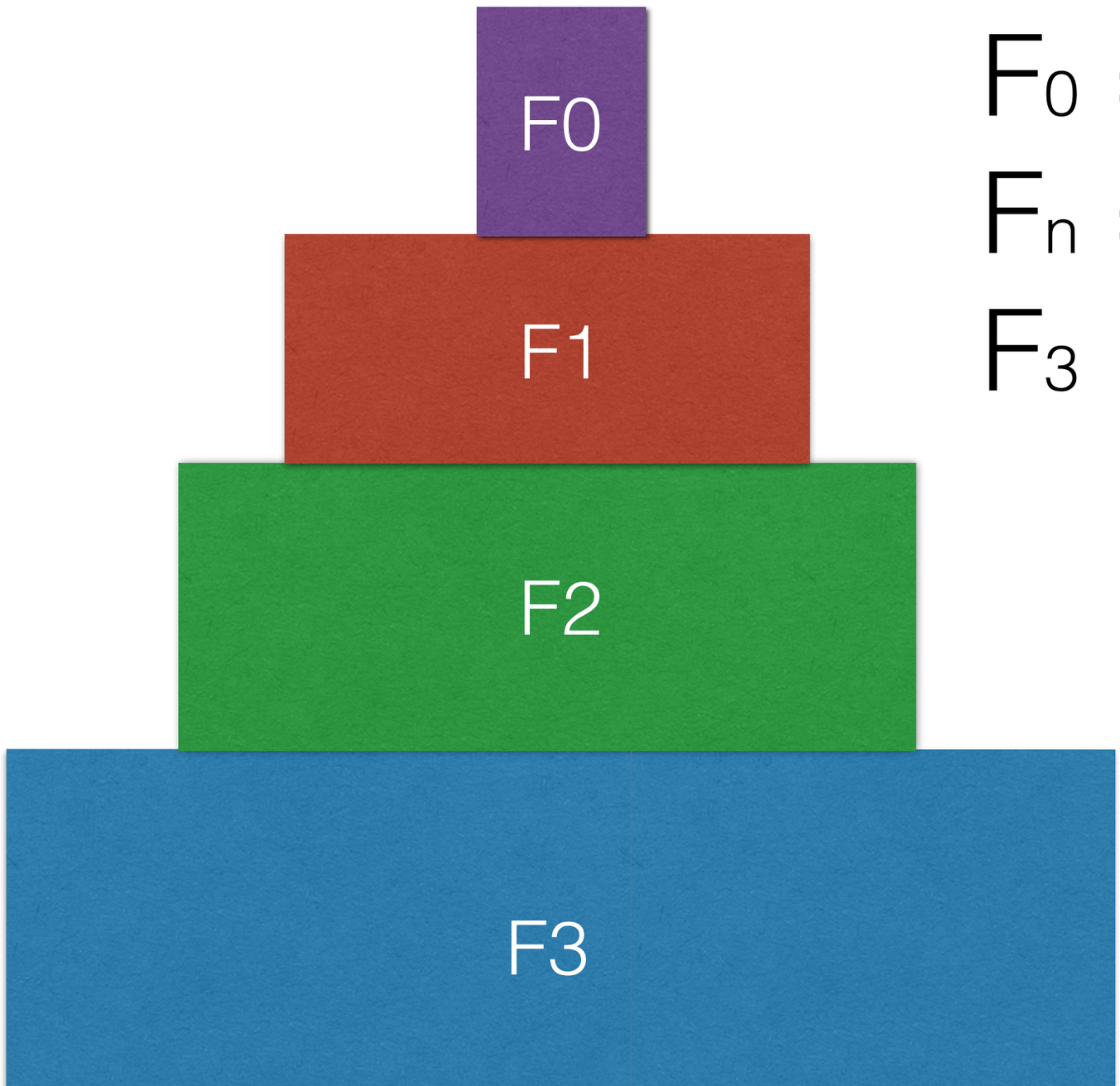
數字太難看？

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

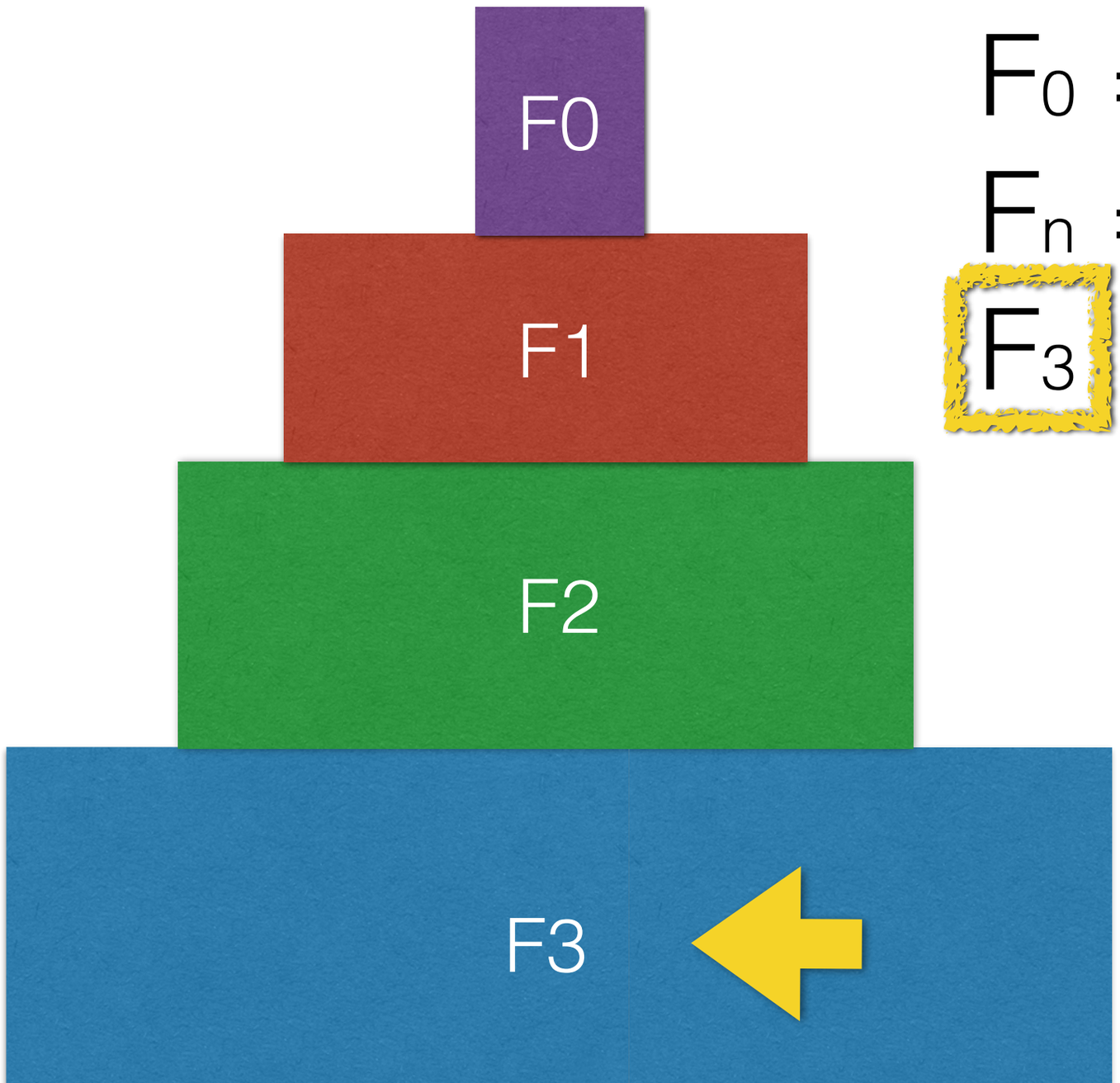


$F_0 = 0$

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

$F_3 = \ ?$

讓遞回變成一座塔：D

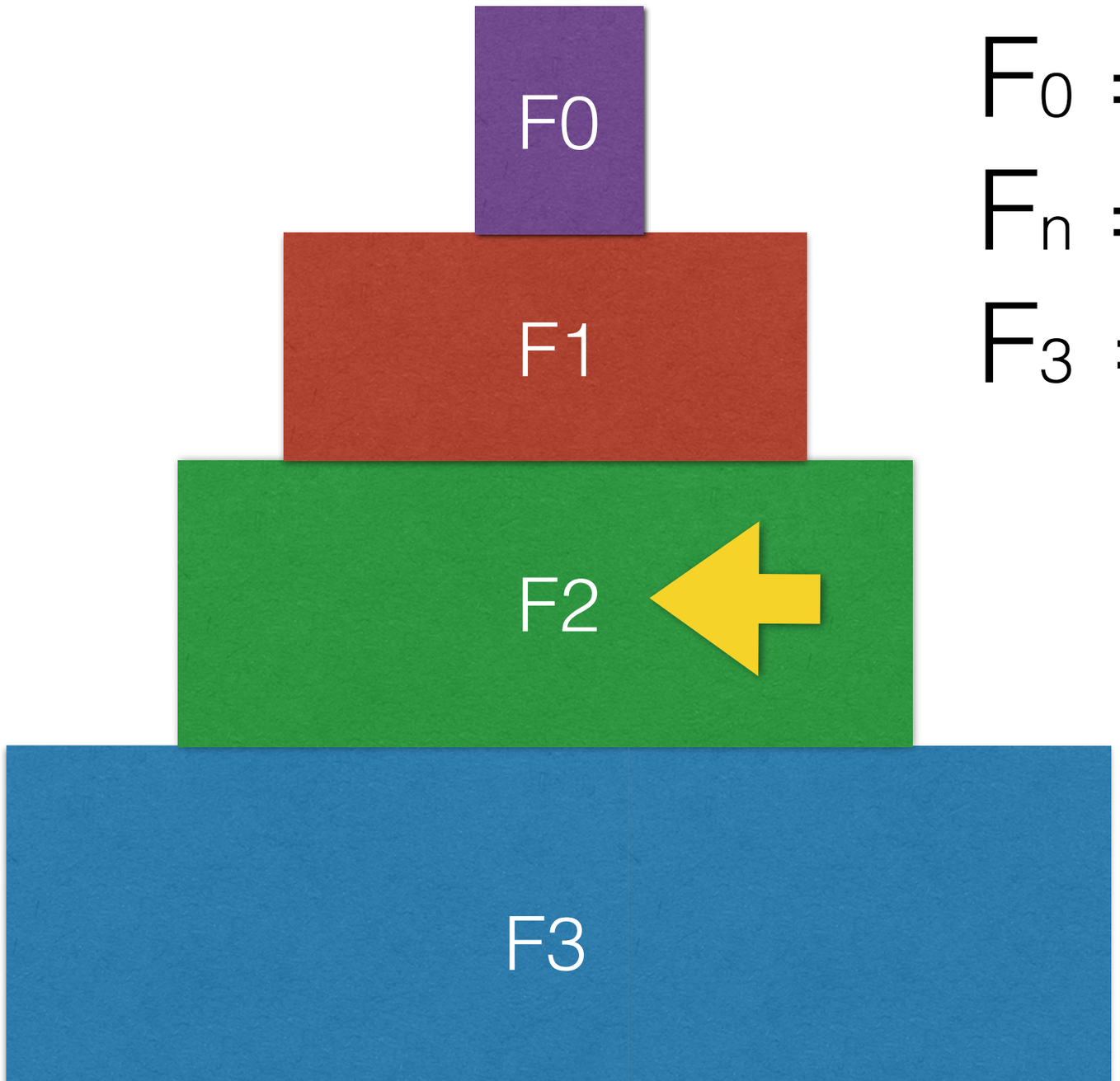


$F_0 = 0$

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

$F_3 = F_2 + 1$

讓遞回變成一座塔：D



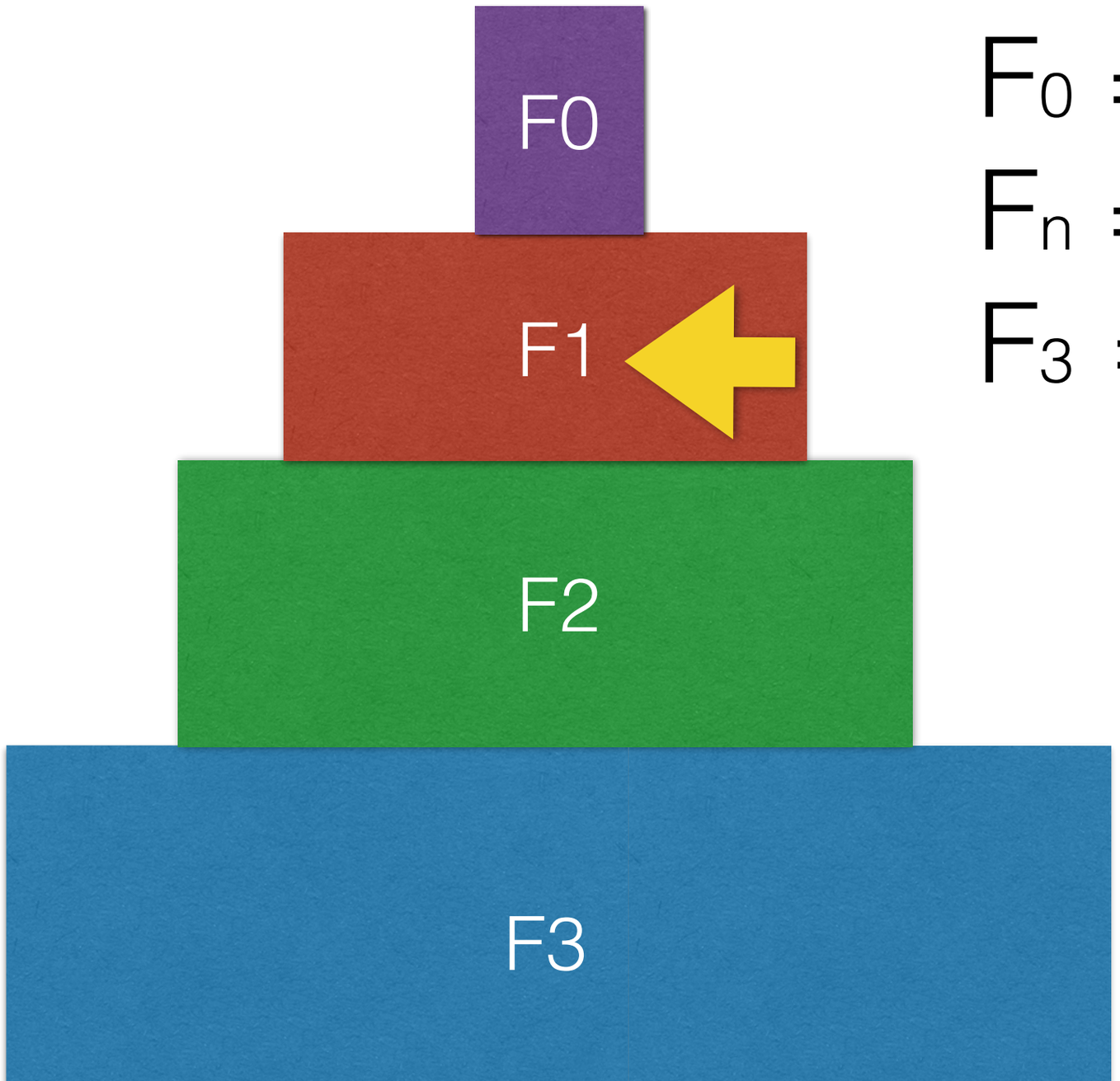
$F_0 = 0$

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

$F_3 = F_2 + 1$

$F_2 = F_1 + 1$

讓遞回變成一座塔：D



$$F_0 = 0$$

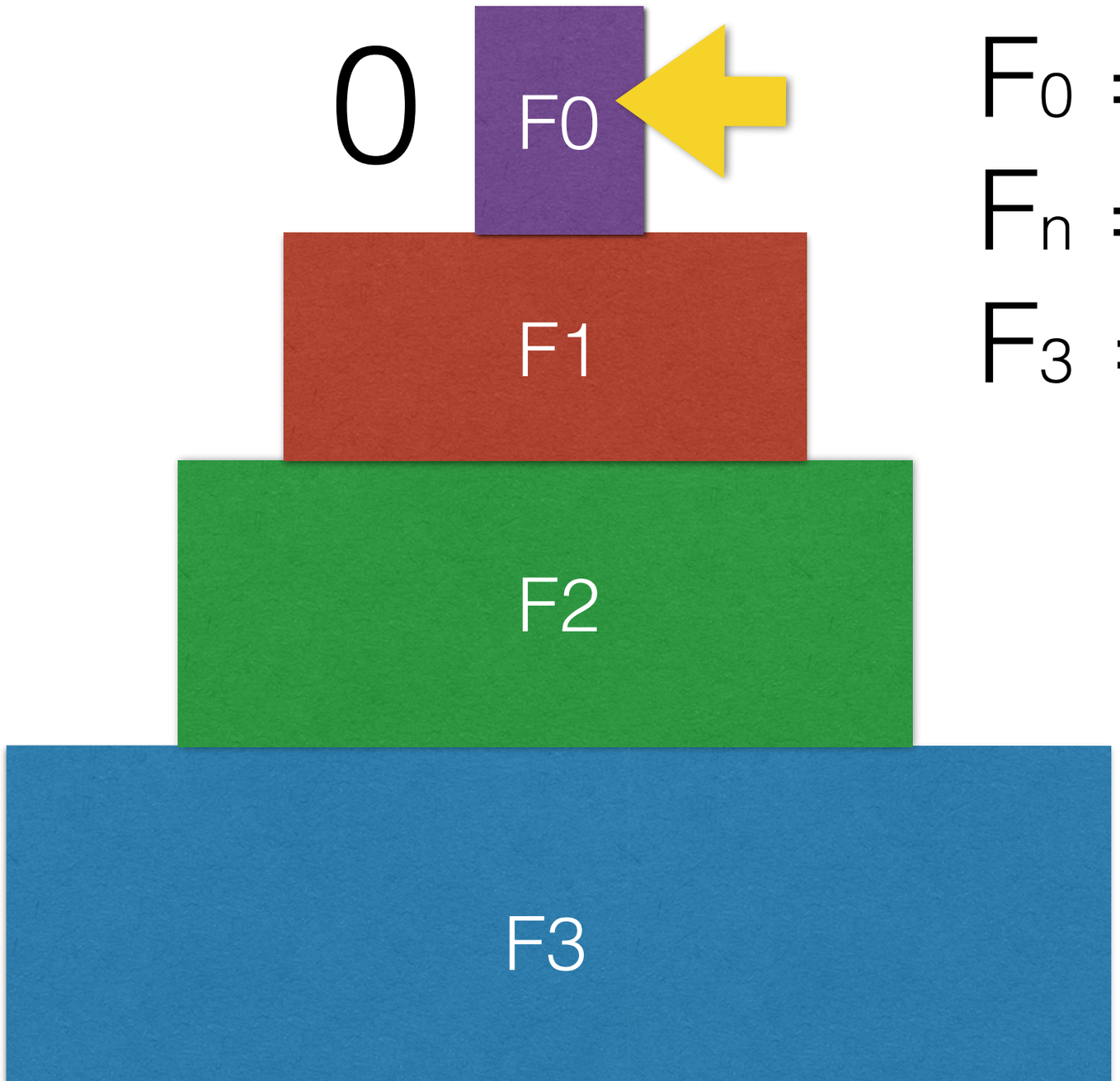
$$F_n = F_{n-1} + 1 \ (\ n >= 1)$$

$$F_3 = F_2 + 1$$

$$F_2 = F_1 + 1$$

$$F_1 = F_0 + 1$$

讓遞回變成一座塔：D



$F_0 = 0$

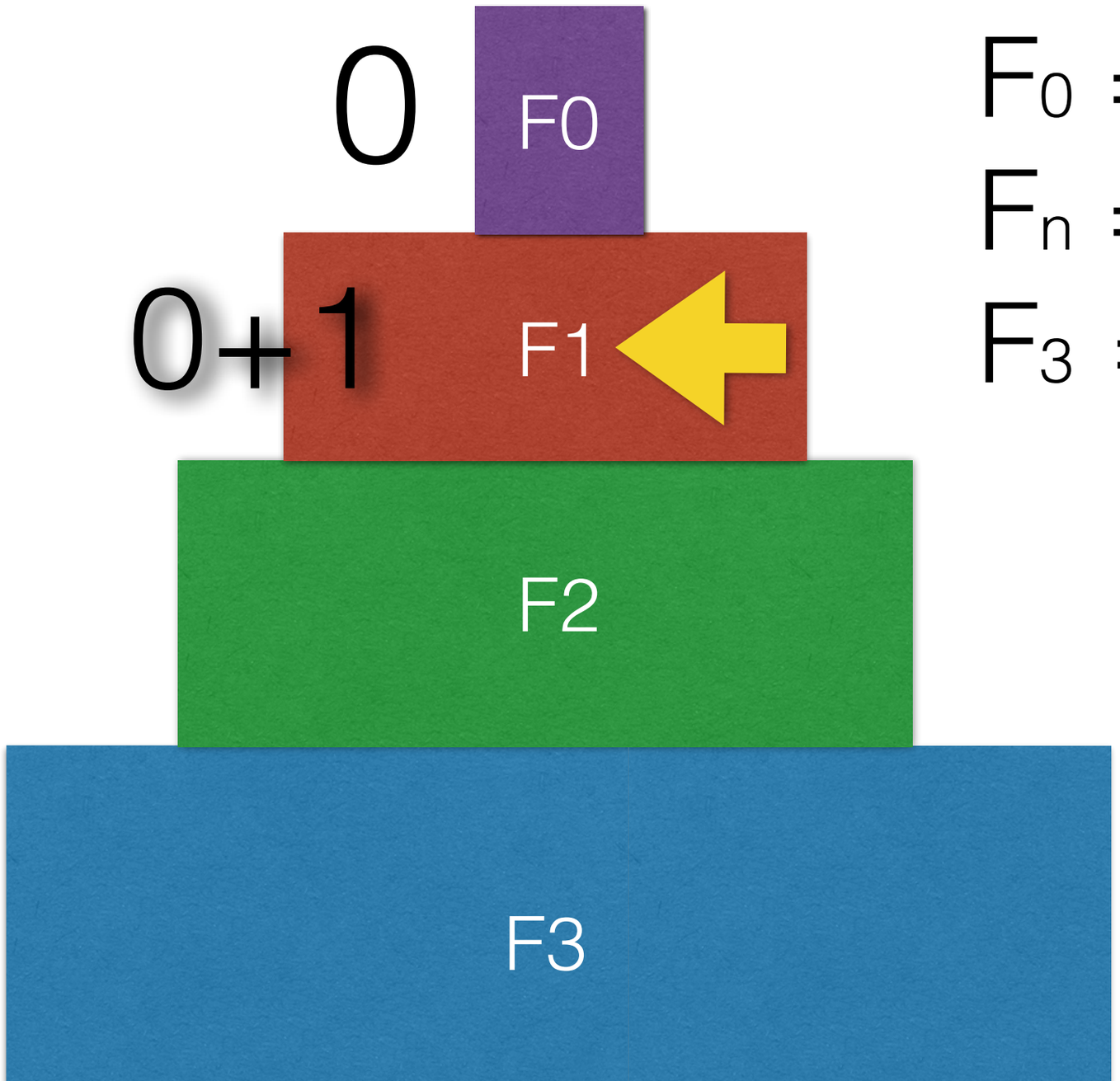
$F_n = F_{n-1} + 1 \ (n >= 1)$

$F_3 = F_2 + 1$

$F_2 = F_1 + 1$

$F_1 = F_0 + 1$

讓遞回變成一座塔：D



$F_0 = 0$

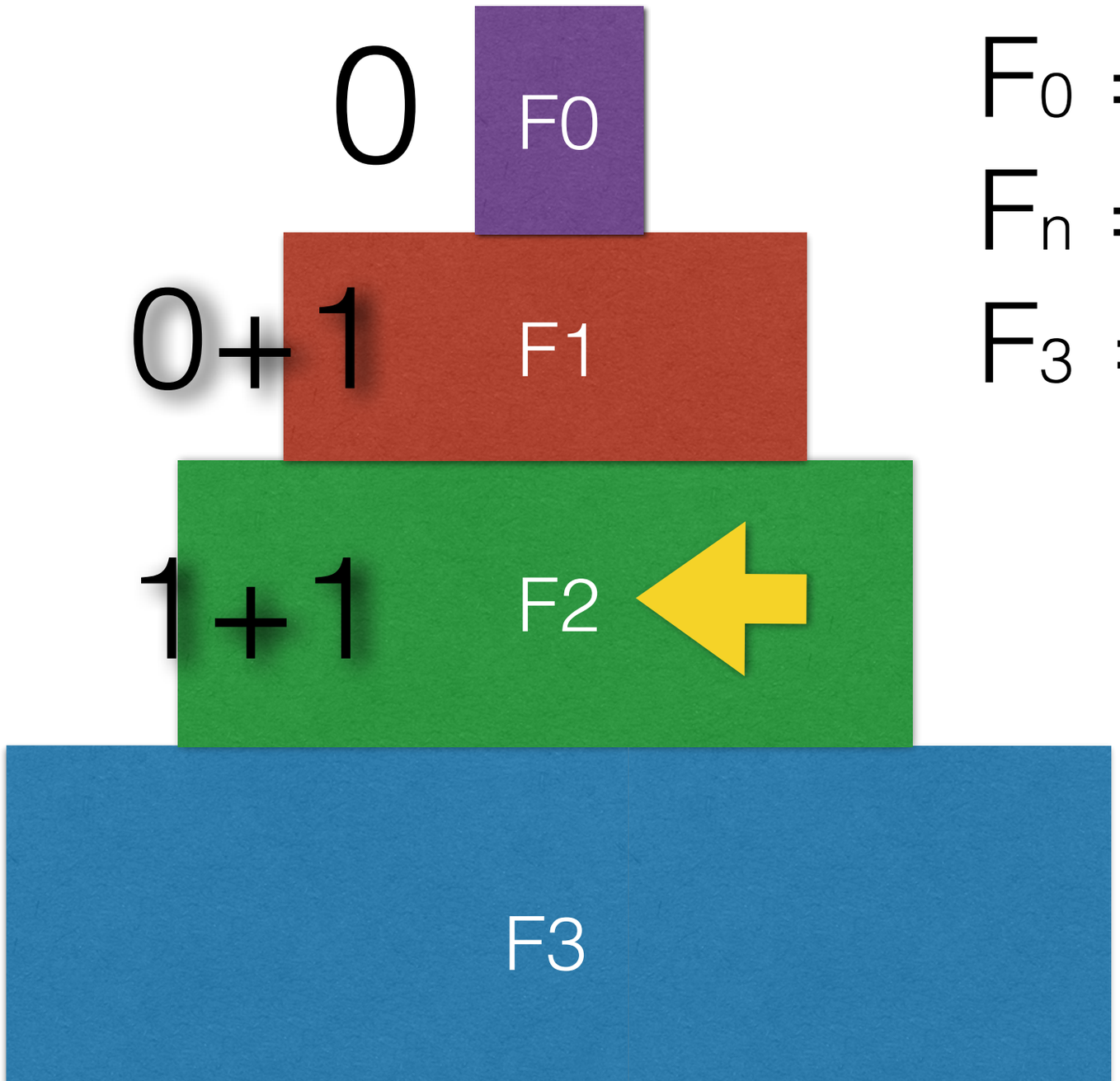
$F_n = F_{n-1} + 1 \ (n >= 1)$

$F_3 = F_2 + 1$

$F_2 = F_1 + 1$

$F_1 = 0 + 1$

讓遞回變成一座塔：D



$F_0 = 0$

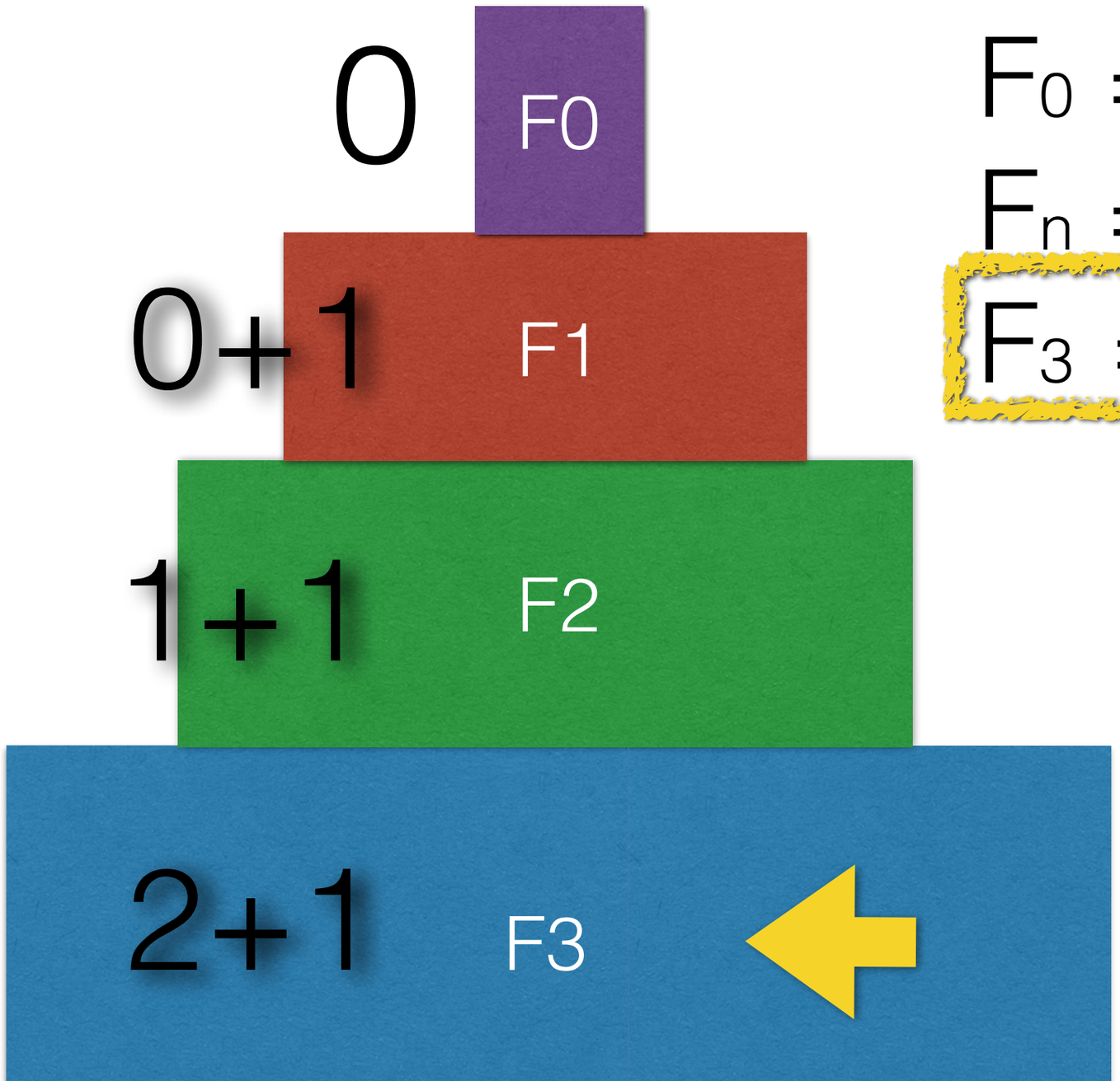
$F_n = F_{n-1} + 1 \ (n >= 1)$

$F_3 = F_2 + 1$

$F_2 = 1 + 1$

$F_1 = 0 + 1$

讓遞回變成一座塔：D



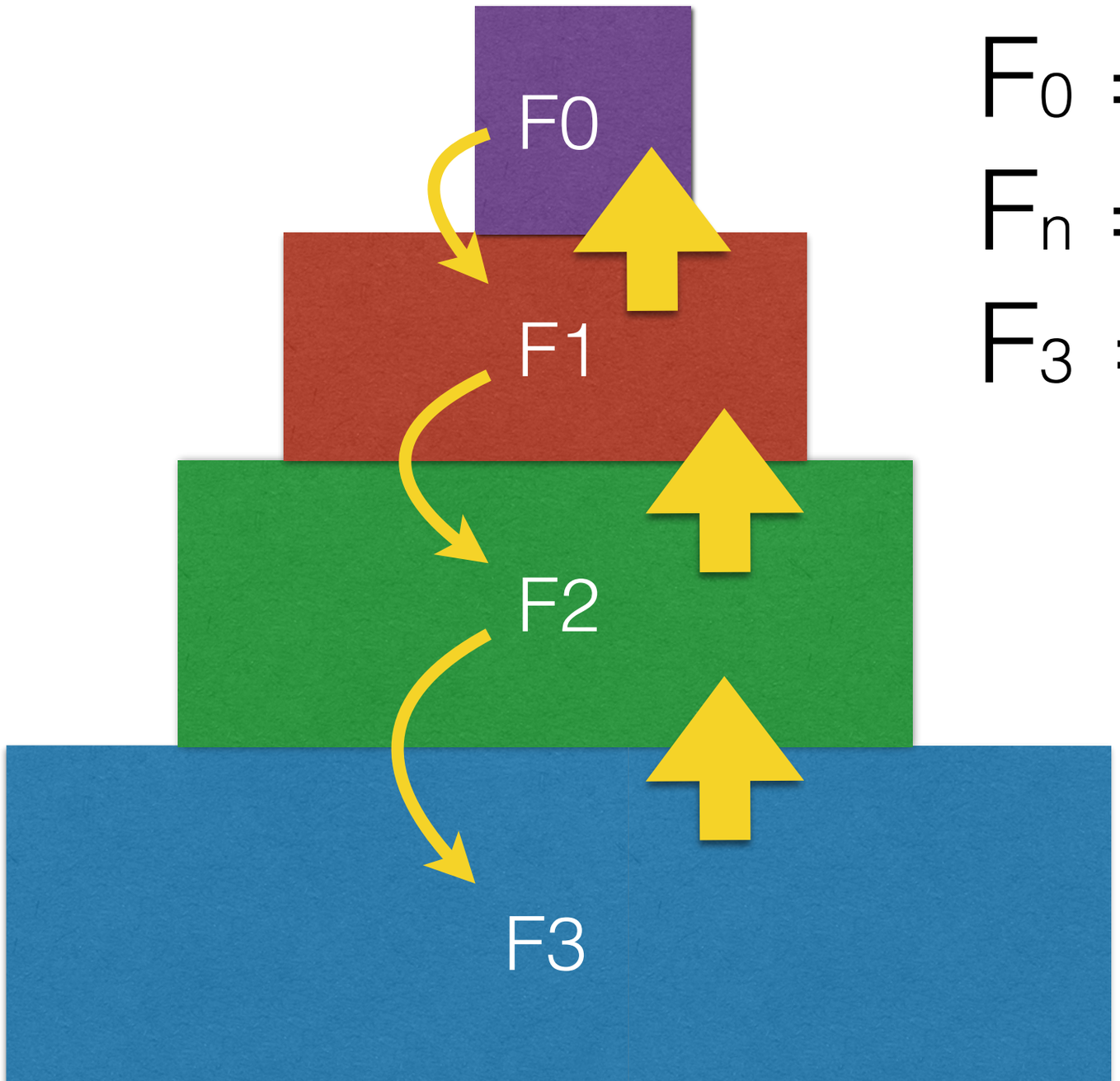
$F_0 = 0$

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

$F_3 = 2 + 1$

$F_2 = 1 + 1$

讓遞回變成一座塔：D



$$F_0 = 0$$

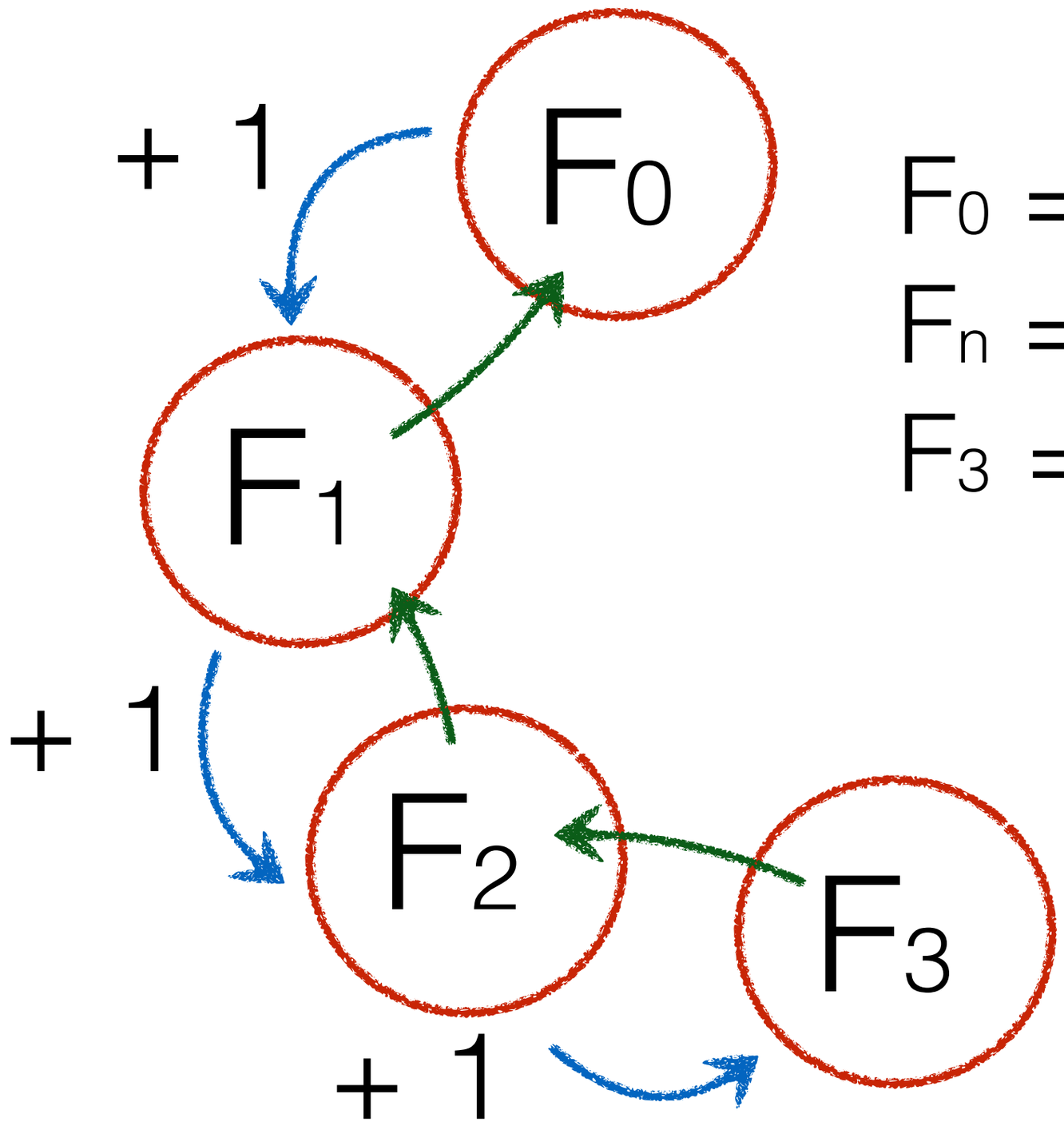
$$F_n = F_{n-1} + 1 \ (\ n >= 1)$$

$$F_3 = F_2 + 1$$

$$F_2 = 1 + 1$$

$$F_1 = 0 + 1$$

當然你也可以想成一串貢丸...



$F_0 = 0$

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

$F_3 = F_2 + 1$

$F_2 = 1 + 1$

$F_1 = 0 + 1$

說了這麼多... 然後咧ＸＤ

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

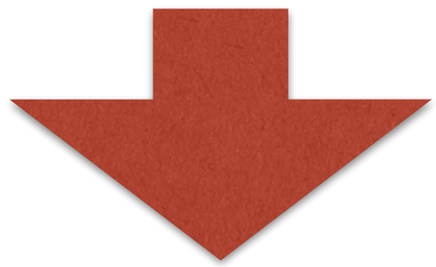
$F_0 = 0$

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

把遞回式寫成函數

$F_0 = 0$

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



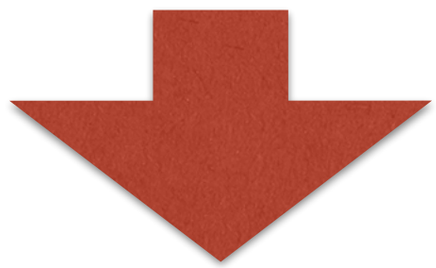
$F(0) = 0$

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

把遞回條件定出來

F(0) = 0

F(n) = F(n-1) + 1 (n >= 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+...+n

要怎麼改剛剛的code呢？

(hint:只要改一個地方)

改這裡噢～

```
 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 }
```

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 }
```

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 - 1 = 2

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 }
```

F(2)

3

直接用Code來思考...

```
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
```

F(2)

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
```

2

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 }
```

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)

直接用Code來思考...

```
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(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 }
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 }
```

1 - 1 = 0

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)

直接用Code來思考...

```
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){
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
```

0

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
```

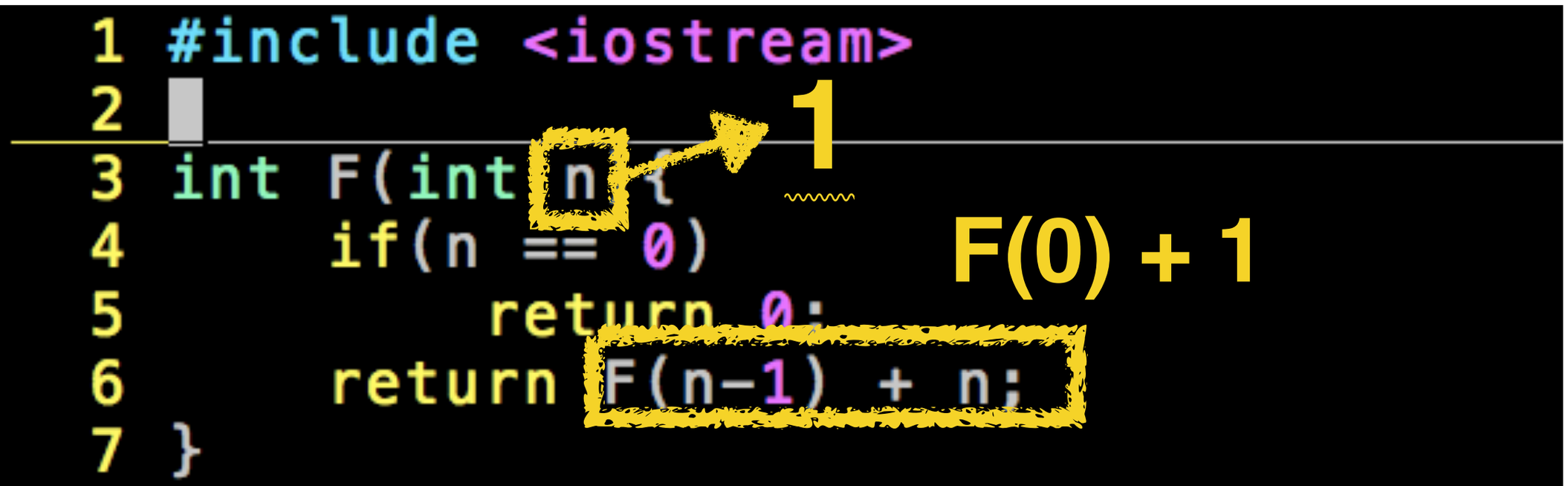
0

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

1

F(0) = 0

直接用Code來思考...



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 }
```

1

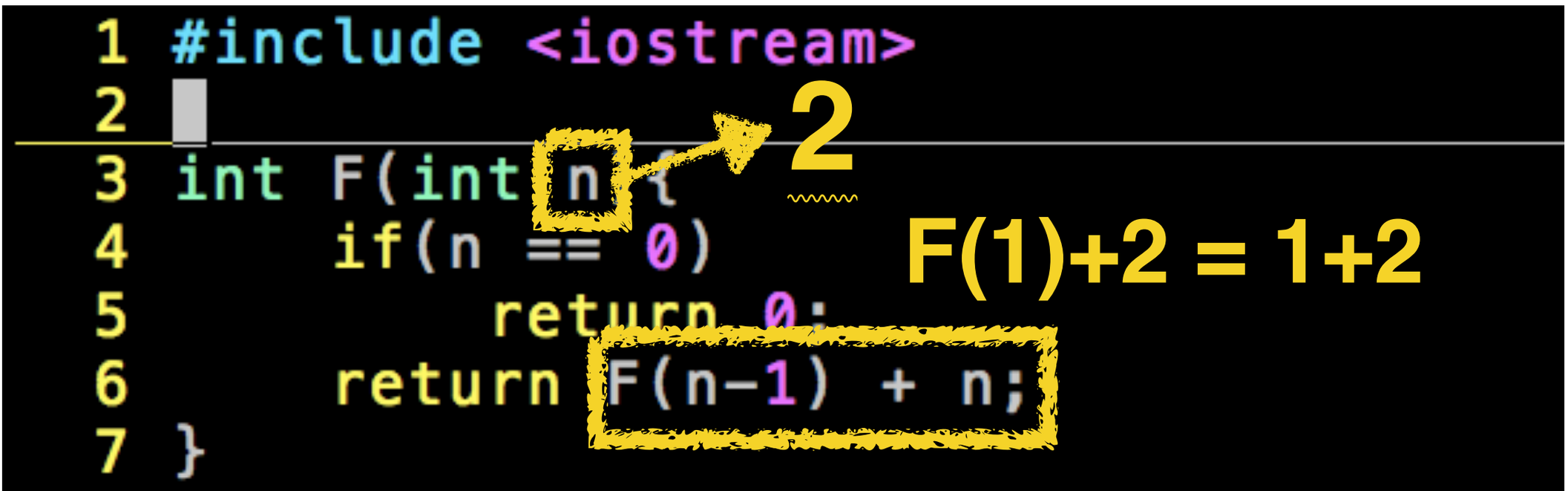
F(0) + 1=1

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

2

F(1) = 1

直接用Code來思考...


#include <iostream>
int F(int n){
if(n == 0)
return 0;
return F(n-1) + n;
}
2
F(1)+2 = 1+2

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 }
```

2

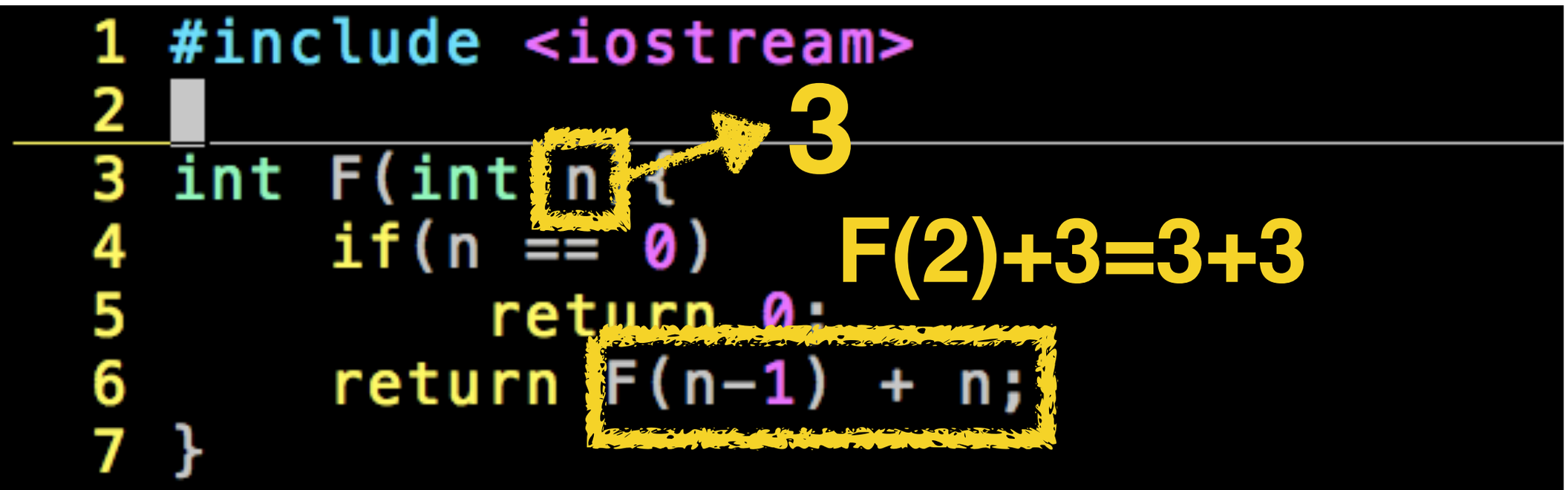
F(1)+2 = 1+2

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

3

F(2)=3

直接用Code來思考...


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

3

F(2)+3 = 3+3

F(3) = 6

~~呼… 投影片終於做完子~~

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ＸＤＤＤＤＤＤ

  - EX. if(n == 0) return 0;

範例：ＧＣＤ

```
 1 #include <iostream>
 2
 3 int gcd(int a, int b){
 4     if(a % b == 0)
 5         return b;
 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 }
```

輾轉相除法

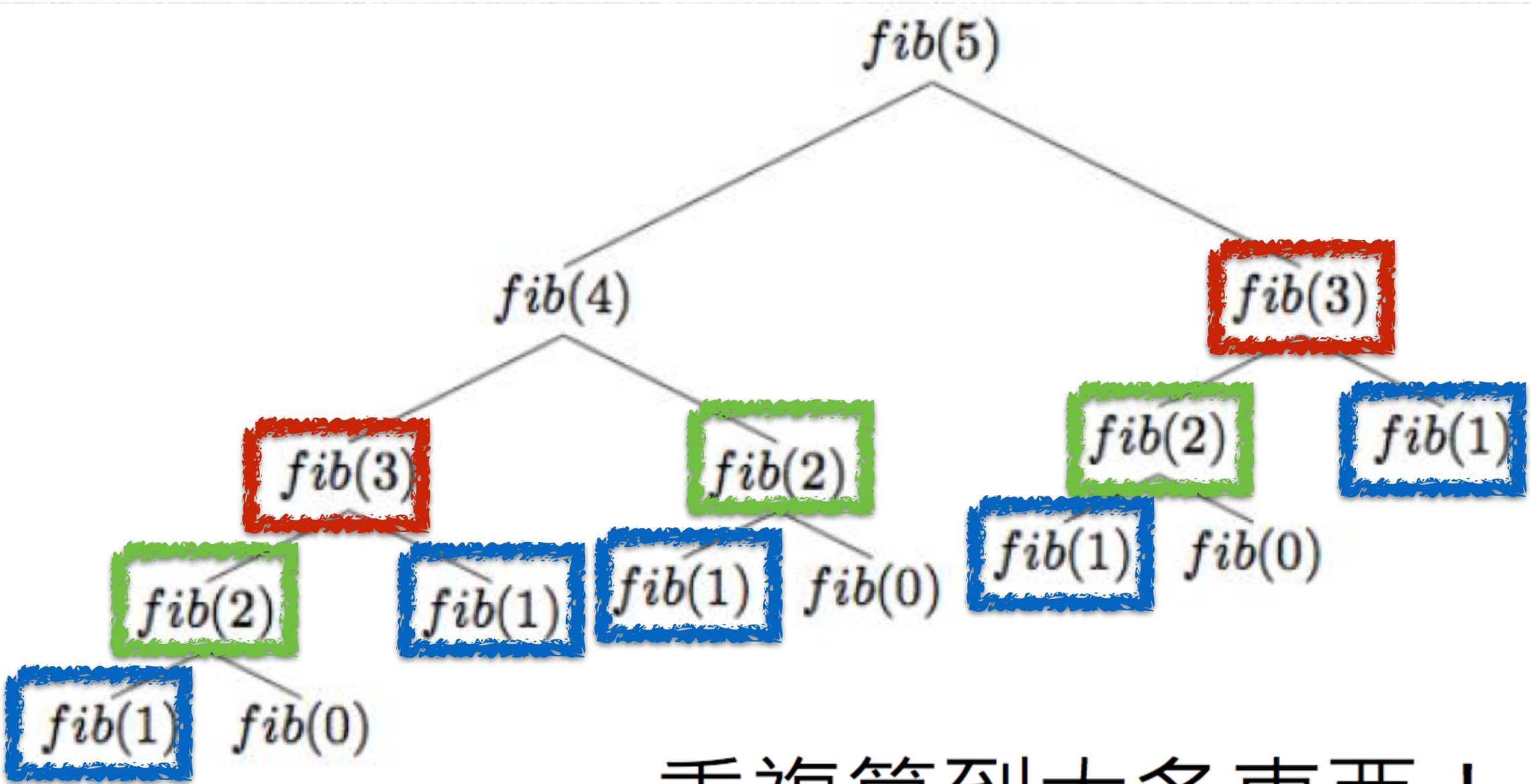
練習：TOJ #33

- 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ＸＤＤＤＤＤ
- 甚至在今天跑不出來ＸＤ

練習：TOJ #33



重複算到太多東西！

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ＸＤ
- 想一下費式數列怎麼用迴圈跑