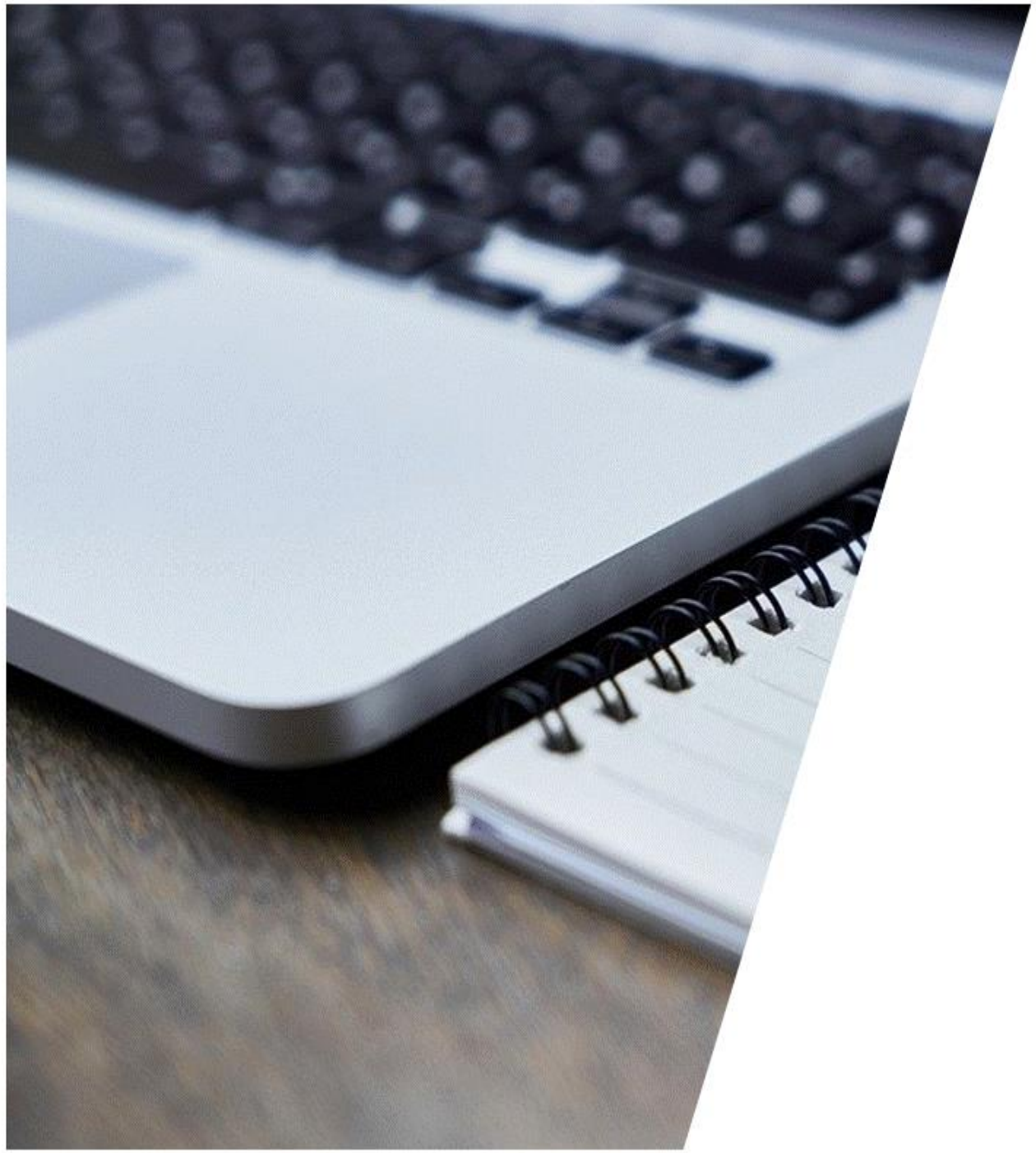


# C 語言

## 遞迴


一些複雜圖騰，
其實是有規則的

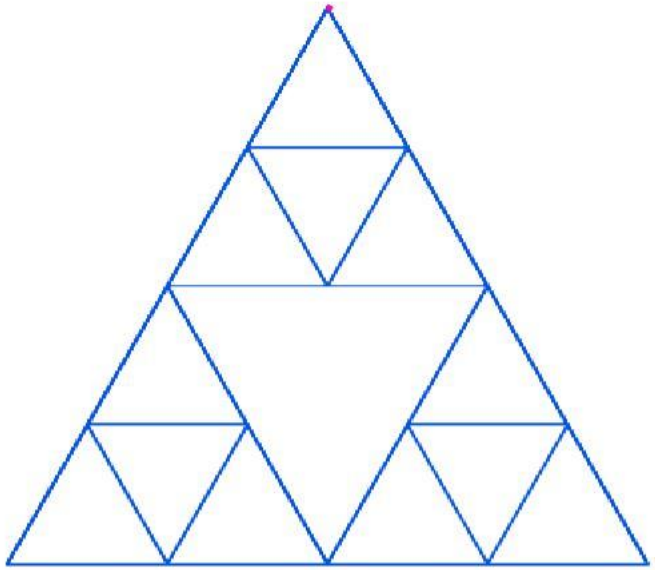
# 自然界的碎形

# 這些圖形，

- 不斷的重複同樣的東西
- 只不過每次重複的是比較小的東西。

# 這個現象，稱爲遞迴(recursion)

- 在電腦科學中，遞迴的本質
  - 重複地將問題分解爲相同子問題的方式
  - 在程式語言支援下，可透過函式中呼叫自身函式來實現遞迴



# 函式呼叫

- 目前我們討論過的函式呼叫，都是以階層式的方式呼叫其他含式
- 不過，對否些類型的問題來說，函式可以呼叫它自己

```c
#include <stdio.h>

float area3(float r,float h);
float area2(float r);

int main(){
    float p=3.14,r,h;
    int i;
    for(i=1;i<=10;i++){
        scanf("%f %f",&r,&h);
        printf("圓柱體體積%.2f\n" ,area3(r,h)); // 1
    }
    return 0;
}

// 2
float area3(float r,float h){
    return area2(r)*h;
}

// 3
float area2(float r){
    p=3.14;
    return 3.14*r*r;
}
```

# 遞迴呼叫

- 函式持續的將每一個問題分成兩個概念性的小塊
  - 每一次衍生出較簡單的問題，逐漸接近基本情況
  - 達到基本情況時，函式會將結果傳給上一分的函式，接著如骨牌效應似的回傳動作，最終可以將原始的函式將結果回傳給main
- 函式呼叫它自己來解決一個較小的問題，稱為遞迴呼叫，也稱為遞迴步驟。
- 遞迴步驟裡包含return這個保留字，因為它的結果會和知道問題該如何解決的部分，結合而形成最後的結果，並且傳回給最原始的呼叫者，可能是main()

```c
#include <stdio.h>
#include <stdlib.h>

int fac(int x){
    if (x>0)
        return x*fac(x-1);
    if(x==0)
        return 1;
}

int main( ){
    int x,y;
    scanf("%d",&x);
    y=fac(x);
    printf("%d\n",y);
    return 0;
}
```

# 練習

用自己呼叫自己的方法，計算階乘值。

給一個n值，計算非負整數的n!
1*2*3*....n的結果。

# n! 數學上的遞迴函式

**例 1: $f_1(n) = n!$**

由 n! 定義可知

$$n! = \begin{cases} 1, & \text{當 } n=0, \\ n \bullet (n-1) \bullet \ldots \bullet 1, & \text{當 } n > 0. \end{cases}$$

當 n > 0 時，

$$\begin{aligned} f_1(n) &= n \bullet (n-1) \bullet \ldots \bullet 1 \\ &= n \bullet [(n-1) \bullet \ldots \bullet 1] \\ &= n \bullet f_1(n-1) \end{aligned}$$

因此，$f_1(n)$ 可重新定義為

$$f_1(n) = \begin{cases} 1, & \text{當 } n = 0, \\ n \bullet f_1(n-1), & \text{當 } n > 0. \end{cases}$$

於例 1 中，``當 n = 0 時，$f_1(n) = 1$" 稱為起始條件。

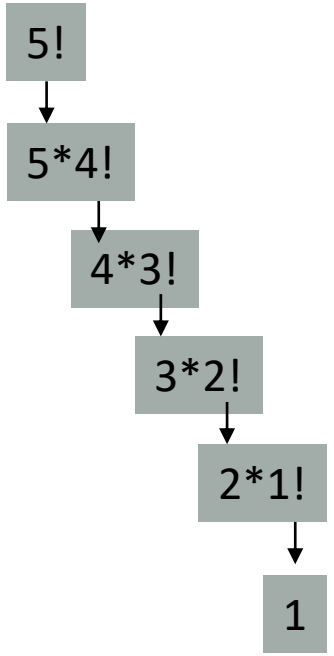
- 像是階梯一級一級的乘下，故以階乘爲名
  - n!=n*(n-1)*(n-2)*.....*1，其中n>0, 0!=1

# 階乘函式的遞迴定義

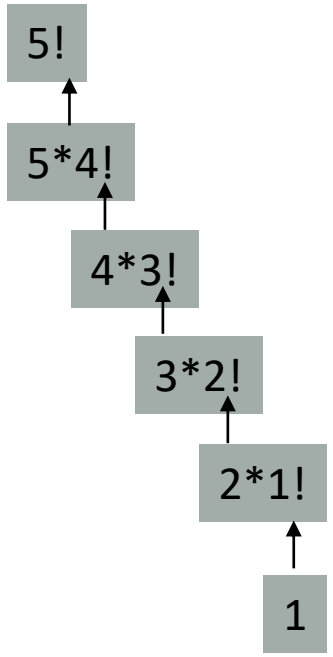
- 像是階梯一級一級的乘下，故以階乘爲名
  - n!=n*(n-1)*(n-2)*…..*1，其中n>0, 0!=1
  - n!=n*(n-1)!

# 以5!為例

- 5!=5*4*3*2*1
- 5!=5*(4*3*2*1)
- 5!=5*4!




遞迴呼叫的進行




傳回5*24=120

傳回4*6=24

傳回3*2=6

傳回2*1=2

傳回1

每一個遞迴呼叫的傳回值

# 將n!數學遞迴轉爲程式

```c
#include <stdio.h>
#include <stdlib.h>

int fac(int x){
    if (x>0)
        return x*fac(x-1);
    if(x==0)
        return 1;
}

int main( ){
    int x,y;
    scanf("%d",&x);
    y=fac(x);
    printf("%d\n",y);
    return 0;
}
```

# 練習

用遞迴解的名題

費氏數列

# 費氏數列 (1/2)

某人飼養一對新生兔子設兔子過了一個月後長大成熟，再過一個月即可下一對兔寶寶。考慮兔子不死亡的條件下試問過了 n 個月後，會有幾對兔子?

| 月數 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 |
|---|---|---|---|---|---|---|---|---|---|---|
| 新生兔 | 0 | 1 | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 |
| 成熟兔 | 0 | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 |
| 總數 | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 |

# 費氏數列 (2/2)

**費波那西數列**（Fibonacci Sequence），又譯**費波拿契數**、**斐波那契數列**、**費氏數列**、**黃金分割數列**。

在數學上，**費波那西數列**是以遞歸的方法來定義：

- $F_0 = 0$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$

# 電腦程式解費氏數列

```c
#include <stdio.h>
#include <stdlib.h>

int fibonacci(int num){
    if (num == 0) return 0;
    else if (num == 1) return 1;
    else
        return fibonacci(num-1)+fibonacci(num-2)
}

int main( ){
    int n,i;
    scanf("%d",&n);
    printf("%d ",0);
    for(i=0;i<=n;i++){
        printf("%d ",fibonacci(i))
    }
    return 0;
}
```

費氏函數

主程式