Chap1 遞迴 Recursion

* 迴圈 vs 遞迴

遞迴：程式簡潔明瞭，較省記憶體空間，存取較費時

迴圈：程式較長，浪費記憶體空間，較能省時間

* 遞迴好用的時候

- 階乘
- 最大公因數
- 費氏數列
- 組合數字（複作業一）
- 河內塔

* Example (階乘)

```
int factorial (int num){
    if (num == 1) return num;
    else return num * factorial (num - 1);
}
```

← 呼叫自己

Chap 2 資料抽象化 Data Abstraction

* simple ADT
  - 清單
  - 行事曆
  - 球

* OOP

```
class Name {
  Attributes: data members
  Behaviors: methods
}
```

* Example (貨物清單)

def predecessor 先行者 . successor 後繼者

List 可以有的 Operations

- 建構
- 解構
- 是否為空
- 計算個數
- 插入
- 刪除
- 檢索

Chap3 鏈結串列 Linked List

* pointer

指標 = 門牌

設一變數 int x;

p = &x; (p存放房子x的門牌)

p = new int; (找一個新地方蓋房子)

q = p (把p上的號碼也讓q記著)

p = NULL (遺忘門牌但房子還在)

delete p; (把房子拆掉也忘記門牌)
p = NULL;

*直接 p = NULL 會造成 memory leak

Chap 4 遞迴解題 Recursion for Problem Solving

- 前序 Prefix 運算子 運算元 運算元 ex + 1 2
- 中序 Infix 運算元 運算子 運算元 ex 1 + 2
- 後序 Postfix 運算元 運算元 運算子 ex 1 2 +

※人日常習慣中序