Ch1. 遞迴

for 皆可改為<u>遞迴</u>
^易讀但有時會複雜化

注意控制範圍!! (要有終止條件)

Binary Search 可減半搜索

找出規律，方可解

費氏數列: (用空間換時間)

n_k at least doubles every other time

$n_k \geq 2^{\frac{k}{2}}$ (指數成長) 效率差

linear Fibonacci makes k recursive calls only (線性成長) 效率高

$n * (n-1) * (n-2) * \ldots * 1$

```
int fact (int n) {
    if (n==0) return 1;
    else return n·fact(n-1);
}
```

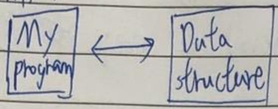
Date

Ch2. 物件導向 →tool → 想好需要的資料及目的
ex: 車子(大小車、輪胎、馬力…)

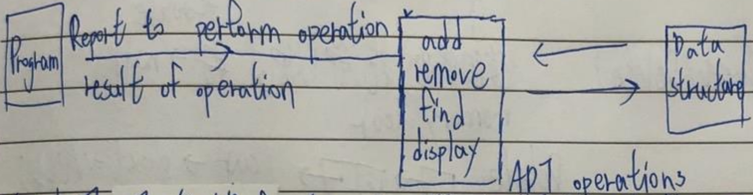
抽象化 → 模組化的延伸 { 內聚 - highly cohesive modules desired
耦合 - loosely coupled modules desired

分析問題
資訊隱藏(描述&實作分開)

Abstract Data Type (ADT)



ADT is composed of
→ a collection of data
→ a set of operations on that data



可利用現存的function去做更複雜的事，不用去看程式碼的寫法重新寫一個(積木組合)
要抓住可變參數該如何有效運用，事前盡量先想好方便使用

Ch3 Link list → pointer　　　int *p;　int x;

宣告卡有空間 (new)　　指向x → p=&x; (有空間)

程式沒結束謹慎使用null (可能使程式掛掉)

動態陣列 (較無彈性)

通常會有 cur (new) & prep 兩個紀錄 (不用到 head)

與 array 比較:　　　　struct　Struct Node{

1. size 較彈性　　　　　int x;

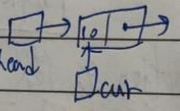
2. 較花空間　　　　　　node * next;

3. 檢索慢　　　　　　　}　　　　先delete

4. 無須搬動 (Insert/Delete)　　後面要刪乾淨以免　再 → null

memory leak

Delete



cur → next=NULL;
delete cur;
cur=NULL
} 不能反!

Ch4.

定義語言: 遞迴判斷文法 ex: 回文, A^nB^n
檢查輸入是否正確
可運用在前、中、後序式

Backtrack: Eight Queen (八皇后問題)、迷宮
走錯就退回來, 找起點終點路徑

數學歸納法: (利用遞迴定義)
可證遞迴
可評估效率

※ Algebraic Expressions

Φ Infix expressions　Ex: a+b 中序
Φ Prefix 〃　Ex: +ab 前序
Φ Postfix 〃　Ex: ab+ 後序

* 一個前序式後面再接上非空字串一定不是前序式