單元1. 遞迴

Date :

	迴圈	VS	遞迴
原理	透過條件式判斷真假以決定程式碼重複的次數		將大問題逐漸變小最後解掉問題
優點	效率高		因為是同一種解法重複，故程式精簡易解

設計遞迴程式碼要素

① 如何讓問題變小

② Base Case 終止條件、保證終止

Ex. 反向印出字串

遞迴函式開始 → 判斷終止條件（字串還有沒有字元）

↓

遞迴呼叫（少一個字元的字串） ← 輸出最後一個字元

Fibonacci Sequence 費氏數列：

$F_0 = 0$

$F_1 = 1$

$F_n = F_{(n-1)} + F_{(n-2)}$ 、$(n \geq 2)$

若一個遞迴呼叫次數為費氏數列

⇒呼叫次數以指數成長

⇒效率差

可如何改善：

① Box trace：

儲存呼叫過的遞迴函式結果，下次要用的時候可直接叫出結果

② 改變寫法：

一次回傳 (F_n, F_{n-1}) 的值，可使呼叫次數變成線性成長

Tail recursion 尾端遞迴：

在最後才再次呼叫遞迴的函式，可容易的改成迴圈，有些編譯器會自動轉化尾端遞迴為迴圈，讓程式更有效率

單元2、抽象化

class 類別：

屬性 Attributes：data members

運算 Behaviors：methods / member functions

物件導向的原則：

1. Encapsulation 封裝：隱藏內部資料
2. Inheritance 繼承：重複使用
3. Polymorphism 多型：為不同資料類型的實體提供統一的介面

Abstract Data Type (ADT) 描述和實作

Modularity (模組化)：

highly cohesive modules desired (高內聚)

Loosely coupled modules desired (低耦合)

Functional abstraction (功能性的抽象化)

Information hiding (資訊隱藏)

描述：解釋這個 function 的功能

實作：用程式碼把描述的功能做出來

好處：程式只要用這些寫好的 function 就能完成，且之後的維護也比較容易

Constructors 建構函式：要求記憶體
建立 class 的初始值
與 class 有相同的名字
不會有回傳值
若沒寫 compiler 也會生成默認的 Constructor

Destructors 解構函式：釋放記憶體
與 Constructors 相反，程式結束時用 Destructors 來釋放記憶體

若 Constructor 寫在 class 外部
要寫成 [class名稱]::[Constructor]
其它 function 也同理

Inheritance 繼承（可同時繼承多個 base class）：
父類別：base class
子類別：derived class（is）
子類別可使用父類別已宣告的東西

Overloading 多載：
名稱相同、但參數不同
系統會判斷傳進的參數執行不同的函式

Private：only class instances
Protected：subclass instances
Public：any class instances

Overriding 覆載：
兩個有繼承關係的 class 可允許有完全相同的函式（名稱、參數列都相同），使用時會以自己 class 裡的為準

Namespaces 命名空間
using namespace [名稱]
可使用 [名稱] 內所有已定義的 class

Exception 例外
try block 設定保護範圍
catch block 捕捉例外狀況

單元3、鏈結串列

陣列：需要移動資料
鏈結串列：不需移動資料

指標＝門牌

```
int *p;   // 沒有值，但不是NULL，沒有房子
p = &x;   // &x = 房子x的門牌
p = new int;   // 申請一棟新房子
delete p;   // 歸還房子
p = NULL;   // 徹底清除門牌
```

動態陣列

陣列名稱就是指標

anArray[2] ≡ *(anArray + 2)

```
double *anArray = new double[50];   // 配置50空間
double *oldArray = anArray;   // 紀錄現在的陣列
anArray = new double[150];   // 配置更大空間
for (int index = 0; index < 50; index++)   // 一個一個複製
    anArray[index] = oldArray[index];
delete [] oldArray;   // 歸還暫存的陣列
```

開檔	關檔	讀檔	寫檔
fopen	fclose	fscanf	fprintf
		fread	furite
		ifstream	ofstream

Linked Lists 鏈結串列

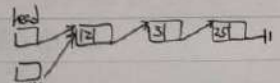
```
struct Node
{ int item;   // 要存的資料
  Node *next;   // 指向下個節點
};
```

必要注意改變指標時不可以讓節點變成沒人指的狀態，不然會找不到

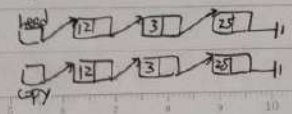
走訪：for (Node *cur = head; cur != NULL; cur = cur->next)
刪除：把刪除對象的上一個接到下一個，再刪掉刪除對象
新增：要新增位置的上一個節點指向新增對象，新增對象再指向下一個節點

複製 Linked List

shallow copy 淺層複製：

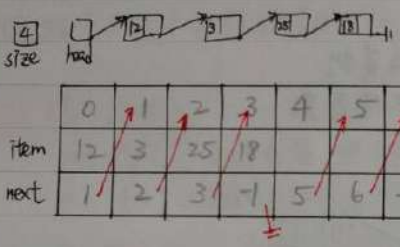


deep copy 深層複製：



Date:

以陣列做鏈結串列



Note:

傳址&：更改指標內容

節點內的資料也可是一個物件

技巧：

① 最後一項指回第一個

② 在第一項新增一個沒有資料的節點

③ 雙向指標：用兩個指標指向前一個和後一個

Date:

單元4. 以遞迴解題

x|y ⇒ x或y

x·y or xy ⇒ x緊接著y

迴文：字串前半顛倒等於字串後半

例：Anna. bob. 4884

運算式： a.b運算元 +運算子

Infix：中序 Ex. a+b

Prefix：前序 Ex. +ab

Postfix：後序 Ex. ab+

中序轉前序：看左括弧

中序轉後序：看右括弧

轉前/後序優點：不用優先權、結合律、括弧

☆一個前序式後面再接上非空字串一定不是前序式

Backtracking：遇到死路就退回上一個點

常與遞迴一起出現

Ex. 八皇后、迷宮

☆遞迴可用數學歸納法證明，以及測式效率