My Notes
Important Concepts worth keeping

第九章

Today: , ,

遞迴：

用處：把問題變得越來越小。

好處：一目了然、簡化、易解釋（不見得速度快！）

應用：階乘、負幻、河內塔、fractal（碎形）……

↓ 塔的個數

解：$2^{n}-1$

Q：反向輸出字串

```
void WriteBackwards ( string s, int size) {
    if (size > 0) {
        cout << s.substr(size-1, 1);
        WriteBackwards (s, size-1);
    } // if
```

→ 此時 size = 0，結束遞迴

```
} // WriteBackwards
```

My Notes
Important Concepts worth keeping

Today: / /

較有效率的寫法：呼叫次數以線性成長.

(3,2) (2,1) (1,1) (1,0)

linearFibonacci(k)

↳ eg: 5 → 4 → 3 → 2 → 1

```
if (k==1)   return(k,0)
else{(i,j) = linearFibonacci(k-1)
    return(i+j,i)
        ↳ ans.
```

河內塔：

```
            個數    起點     終點        輔助
solveTowers(count, source, destination, spare){
  if (count == 1)
    cout << "Move a disk directly from"
         << source << "to" << destination;
  else
    solveTowers(count-1, source, spare, destination); // 將count-1個放到spare
    solveTowers(1, source, destination, spare); // 將最大的(剩下的)放到終點
    solveTowers(count-1, spare, destination, source); // 將其他的移往終點
```


My Questions → 不適合用遞迴
Problems & Difficulties needing exploration

My learning weather report

eg: n選k個 $(C(n,k) = 0 \text{ if } k > n)$

base case：$C(k,k) = 1$, $C(n,0) = 1$

$C(n-1,k-1) + C(n-1,k)$ if $0 < k < n$

呼叫次數：$C(n,k) + C(n,k) - 1$ ⌈∵是2元樹的k出 能運算的次數

(base case) (base case −1 = 非 base case)

Tail Recursion：尾端就是 only "recursion 的呼叫.

⇒ 易轉換成迴圈

My Opinions
Thoughts, inspirations, and suggestions

密碼 cipher key

My Notes
Important Concepts worth keeping

單元2

Today: , ,

class：含 ① data members
② methods

物件導向：封裝(encapsulation)、繼承(inheritance)、多型(polymorphism)

ADT list 's operation

createList() → 建構

destroyList() → 解構

isEmpty() → 是否為空 回傳 boolean

getLength() → 計算個數 回傳 int

insert( ) → 插入(將原位置資料往後移)

remove( ) → 移除(後面的資料將往前移)

retrieve( ) → 檢索 回傳 int

(位置, 元素, success)

↳ 傳回 boolean 確認對錯.

Begin from the end in mind. —Steven Covey

My Questions
Problems & Difficulties needing exploration

My learning weather report

結用 ADT 's operation：

eg: reverse

```
for (i=1 to aList.getLength()-1){
    aList.retrieve(i, item, success);
    aList.remove(i, success); // 先刪除
    aList.insert(aList.getLength()-i+2,
                 item, success); // 後插入
} //for
```

```
try{... throw(type); ....}
catch(type1)
    statement(s);
catch(type2)
    statement(s);
    ⋮
```

My Opinions
Thoughts, inspirations, and suggestions

namespace：命名空間

↳ 最常用：using namespace std; // 標準函式庫

exception：例外處理 (try, catch blocks)

{ try：用在拍出問題的地方
  catch：用在例外的狀況

類似 switch、case！一次做其中一種狀況！

My Notes 單元3

Important Concepts worth keeping

Today: / /

int *p; int x;

eg: ① p = &x → & = adress of P指向x.

② *p = 6 → 表示 *p 內容 = 6 → 加0* 代表內容!

先做①才能做②

delete !

* new完後 delete，不能設為 null!

eg: int *p;
int *q;

p = new int;

*p = 8;

q = new int;

*q = 4; ⇝ memory leak → 易 RTE (run time error)

p = NULL; // 不對! 應先用 delete 釋放記憶體位址

delete q; // 對的! 不會流覽空間.

q = NULL;

2個應緊接在一起! 中間不要再放其他程式!

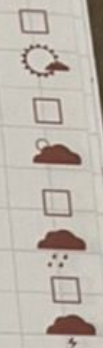
如中間有再用到q → 發生 dangling reference (illegal access)

"We recognize individual differences with respect to talents, character, capability, and background. We believe that full development of one's potential signifies success"

(CYCU's Education Philosophy)

My Questions
Problems & Difficulties needing exploration

My learning weather report



動態陣列宣告：

```
int arraySize = 50;
double *anArray = new double[arraysize];
```

這點！

當 oldArray 內的資料太多，想將之移到一新的動態陣列時

```
eg: double old = newA; //old中有資料
    newA= new double[3 * arraySize]; //newA可存資料量
                                     //滿old的3倍
    for(int i = 0; i < arraysize ; i++) //要重新將資料
        newA[i] = old[i];               //放入newA
    delete [] old; //刪掉舊的array!
```

↑ 讓電腦知道要刪除的是array.

My Opinions
Thoughts, inspirations, and suggestions

觀念：

有開檔 → 有關檔

有 new → 有 delete

有可能配置記憶體失敗：放在 try 中，去 catch 失敗狀況

My Notes 單元4

Important Concepts worth keeping

Today: / /

運算式：{ Infix：中序 eg：a+b ｜ Prefix：前序 eg：+ab ｜ Postfix：後序 eg：ab+ } ← 指運算子位置

→ 電腦較好處理！

中序 ① → 前序：看"("

② → 後序：看")"

↓

好處：不必看優先順序、結合律、括號

① eg：((a+b)×c) → xy+abc

② eg：((a+b)×c) → abcx+

My Questions

Problems & Difficulties needing exploration

backtraking

- 常和 recursion 一起出現

→ 解 8 皇后問題

My Opinions

Thoughts, inspirations, and suggestions