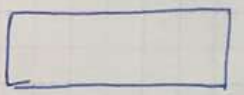
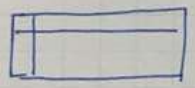
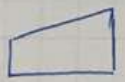
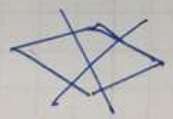
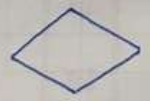
My Notes
Important Concepts worth keeping

Today : / /

流程圖 (flowchart)

流程圖 → 視覺化 的 形式

- 起止符號 (Terminal) 
- 流程符號 (Flowline) → →→→
- 程序 (Process) 
- 宣告符號 (Declare) 
- 人工輸入 (Manual input)  
- 顯示符號 (輸出) (Display) 
- 決策符號 (if) (Decision)  
  - 依決策結果擇一進行

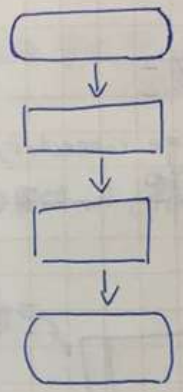
"Punctuality:Showing esteem for others by doing the right things at the right time." (IBLP)

My Questions
Problems & Difficulties needing exploration

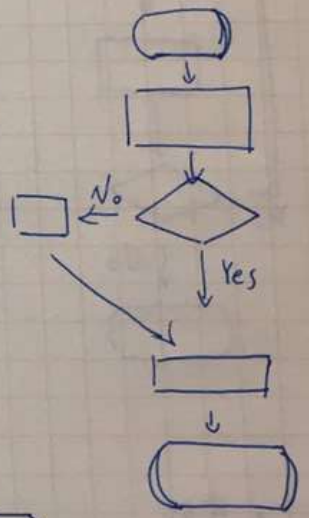
My learning weather report

- 主要流程架構

1. 循序 (sequential) 架構.

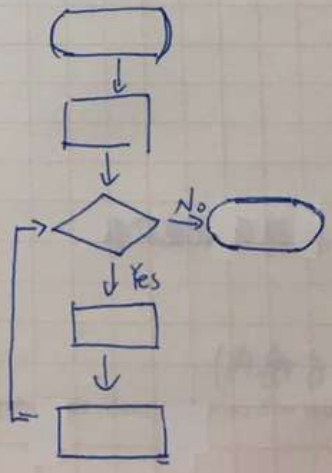


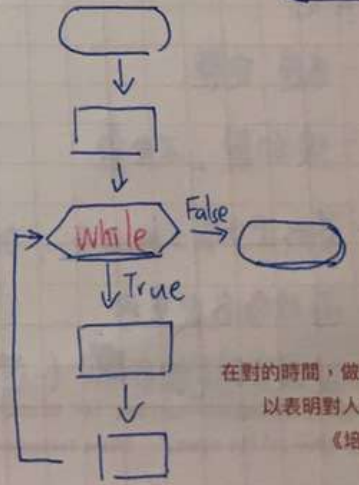
2. 選擇 (selective) 架構.



My Opinions
Thoughts, inspirations, and suggestions

3. 重覆(Iterative) 架構.
依檢測結果決定是否重覆進行





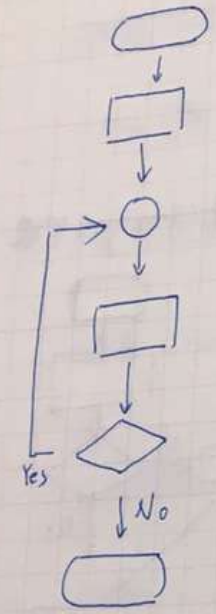
密碼 cipher key

在對的時間，做對的事，
以表明對人的重視。
《培基文教》

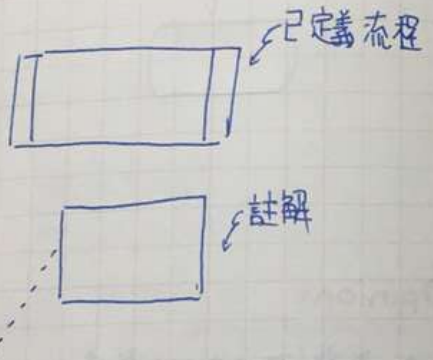
My Notes
Important Concepts worth keeping

Today: / /

do-while loop



○ 同頁參考.
(on-page connector)
沒有步驟, for做圖好看



- 須知
  - 清楚,完整.
  - 線和圖,不重疊
  - 菱形至少有2個分支,加注箭頭對應的值
  - 函數命名很重要
  - 避免寫入多個步驟 (一個方塊內)

"Do not worry about tomorrow; for tomorrow will care for itself. Each day has enough trouble of its own." (New Testament)

My Questions
Problems & Difficulties needing exploration

- 以虛擬碼 (Pseudo Code) 代替程式碼
- 詳細描述,無錯字
- 使用變數前,簡述定義,用途

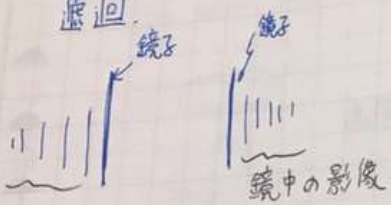
My Opinions
Thoughts, inspirations, and suggestions

不要為明天憂慮,
因為明天自有明天的憂慮,
一天的難處一天當就夠了。
《新約聖經》

My Notes
Important Concepts worth keeping

Today: / /

1-01
遞迴.



≠ (遞迴. → 要把大問題 縮小成小問題
迴圈 → 重覆執行.

binary search ——→策略: divide and conquer

小問題解決, 大問題就解決了 ——宜鴻

1-02
以遞迴將 字串倒過來印. ——→ 字串長度減一
(大) (小)

base case 遞迴中止的 case

So in everything, do to others what you would have them do to you...
〈New Testament〉

My Questions
Problems & Difficulties needing exploration

My learning weather report

1-05 載好的程式 ⇒ 遞回 ~~次數少~~
呼叫

1-06 binary search, 找最大. ⇒ 效率低
未排序的數列 ⇒ 分成一半 ⇒ 左邊陣列找最大
右邊陣列找最大

1-07 找第 k小 & 排序. ⇒ 適合用遞回.
陣列中找一個數為 pivot.
> pivot 放右邊.
< pivot 放左邊

My Opinions
Thoughts, inspirations, and suggestions

linear Recursion
- Test base case
- 條件可以有很多條遞回 但只走其中一條

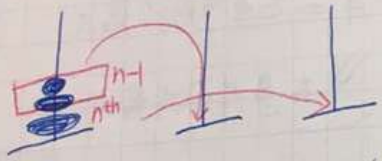
所以無論何事，
你們想要人怎樣待你們，
你們也要怎樣待人。
《新約聖經》

My Notes
Important Concepts worth keeping

Today: / /

1-08 Hanoi 河内塔

解3個盤子の問題 $\xrightarrow{\text{recurrsion}}$ 第一步上2個盤子如何 移動



n個盤子 ⇒ 2^n-1 步可全部移動.

My Questions
Problems & Difficulties needing exploration

My learning weather report

1-10

1-11 是 效率の高低 不又有 呼叫次数为依據

or 加法次数 or .....

1-12 费氏数列

(0), 1, 1, 2, 3, 5, 8, 11 .....

1-13 用相對宿舍の方法描述 呼叫次数.

描述≠实際

呼叫次数以指数成長 ⇒效率極低,

$n_k \geq 2^{\frac{k}{2}}$

*用空間換時間!!! or 其它遞回定義.

My Opinions
Thoughts, inspirations, and suggestions

呼叫次数以線性成長 ⇒較好の效率

1-17 |leaf nodes| - |internal nodes| = 1.

n個(總)盤 k個⇒有幾组.

C(n,k) = |leaf node|

My Notes
Important Concepts worth keeping

Today :

1-18

Tail recursion : 遞回後沒有其它執行動作

⇒ 易改成迴圈

Summary

1. Define the problem in terms of smaller problems
   遞回定義
2. See if a recursive call decreases the problem size
   問題簡化
3. Find a complete set of base cases
   終止條件
4. For every case it can eventually reach a base case
   保證終止

My Notes
Important Concepts worth keeping

Ch2 抽象化

Today / /

抽象化的
目的：節省維護，除錯時間

物件導向 review.

- 所有東西都是一個物件.
  - classes of objects (called instances)
  - Attributes ⇒ data member
  - behavior ⇒ method

程式所有要處理的東西都是物件.

- 一群物件！

- principles of Object-Oriented Programming

1 封裝 encapsulation
- object combine data & operation
- Hides inner detail

2. 繼承 inheritance (超卓结的概念)
- classes can inherit properties from other classes
- existing classes can be reused

3. 多型 polymorphism
- objects can determine appropriate operations at execution time

My Questions
Problems & Difficulties needing exploration

My learning weather report

operation contract : 設計程式時要先寫清楚的資料
- document the use and limitations of method
- specify data flow
- do not specify how module will perform its task
- specify pre- and post-condition
- unusual conditions 例外狀況.
  - assume they never happen
  - ignore invalid situations
  - return a value that ~~sign~~ signals a problem
  - throw an exception ⇒ 所有例外一個類別 → 集中管理

My Opinions
Thoughts, inspirations, and suggestions

- clearify before design the program
  - Purpose → 程式要解決的問題
  - Assumption → 假設情況 (What does the module assume)
  - Input → What data is available)
  - Output → effect

通常放在 header file

密碼 cipher key

凡是值得做的事，都是上坡路。
—約翰·麥斯威爾

My Notes
Important Concepts worth keeping

Today / /

Key Issue in Programming

1. Modularity
2. Style
3. Modifiability
4. Ease of Use
5. Fail-safe programming
6. Debugging
7. Testing

抽象化資料型態 ADT.

- 模組化 ⇒ 使程式可切割. (modularity)

好的程式指標 ⇒ 高內聚 cohesion - modules perform single ~~well-define~~ well-defined tasks

highly cohesive modules desired.

(每個函式只做一件事.)

⇒ 低耦合 loosely coupling

↳ measure of dependence among module

Habit is habit, and not to be flung out of the window by any man, but coaxed downstairs a step a time. —Mark Twain

My Questions
Problems & Difficulties needing exploration

My learning weather report

- functional abstraction. 模組化.

① specification 描述 (spec)

② implementation 實作.

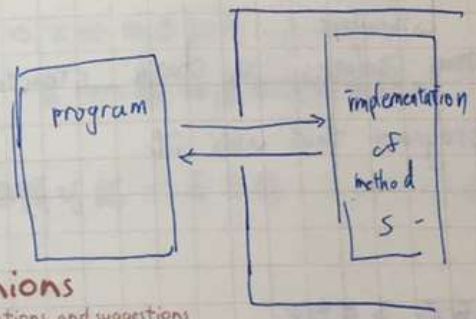
描述&實作分開

實作可改, 但描述之目的

目的一改, 其它就會有牽連

- information hiding 資訊隱藏.

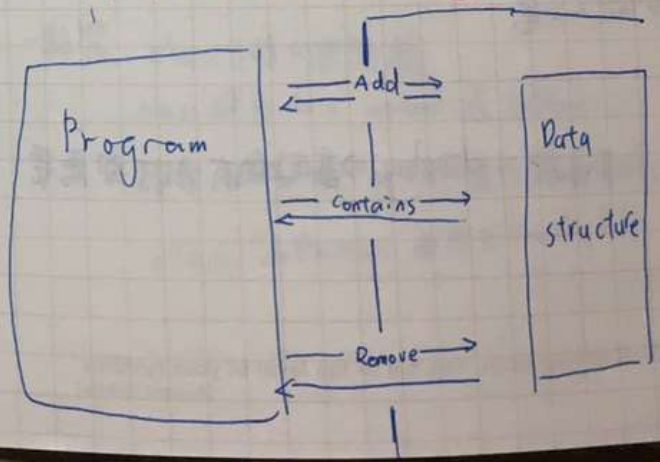
hide certain implementation details.



Isolation of module is not total

My Opinions
Thoughts, inspirations, and suggestions

- 資料抽象化 → 不須知道如何達成, 而是要達到什麼!



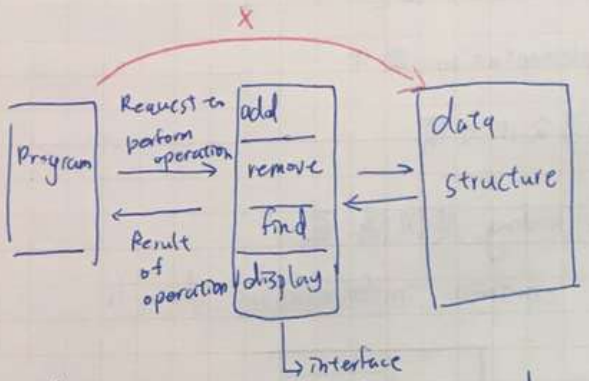
習慣就是習慣,
誰也不能將其扔出窗外,
只能一步一步地引下樓.
—馬克·吐溫

密碼 cipher key

My Notes
Important Concepts worth keeping

Today / /

ADT = data's + operation.



ADT operations isolates a data structure from the program that uses it.

predecessor → 順序靠前 → 先行者

successor → 順序靠後 → 後繼者.

head → 頭 ↔ Tail → 尾.

add, remove, find, display ⇒基本上所有 ADT都需要

You do not rise to the level of your goals. You fall to the level of your systems.
《Atomic Habits》

My Questions
Problems & Difficulties needing exploration

My learning weather report

設計

① ADT List Operation → 目的. 有什麼.

② Operation Contract for the Data list.

→ 以函數宣告., 每個函數需要什麼給我,
會有那些 函數, 會提供の interface 有什麼

③ 根據 Spec去~~implement~~ 實作

※ ADT ⇒ 不必知道如何implement.
看 spec就知道如何使用就ok

分析問題時就一'清目的!

My Opinions
Thoughts, inspirations, and suggestions

如何解一個問題 implementation
避免 violating the wall

-封裝 class可將內容隱藏
class裡有 data member & method.
設定 public or private 達到隱藏資料
class definition 通常在 header file

密碼 cipher key

決定你成功或失敗的，
不是你的目標，
而是你的系統。
《原子習慣》

My Notes
Important Concepts worth keeping

Today / /

⊖ 每一個 class 都有一個 constructor

constructor ≃ 型別構造,找到一塊記憶体,儲存該型別資料

constructor ≥ 1

↳ 不會有 return 值

→ constructor

Sphere ::Sphere(): theRadius (1.0)

屬於 sphere class 的 Sphere() function

destructor · 還回記憶体.

class spec 在 .h 檔裡

宣告.

在 .cpp 使用,呼叫 function 時都要有 class 名+::

Sphere :: Sphere() : theRadius .

My Questions
Problems & Difficulties needing exploration

My learning weather report

↱ .h

宣告: double getRadius() const;

↳ can't change data members.

.cpp 寫 function 的 definition

.h 也可寫 definition.

.h

.cpp

definition 定義 function. ≃ implement.

繼承

class ColoredSphere : public Sphere
{

繼承 被繼承

若 private 的話 繼承

inheritance ⇒ 不需重新定義就可直接使用父類別的 method & datamember

父: base class

子: derived class

My Opinions
Thoughts, inspirations, and suggestions

overload 多載 → 參數不同 ~~名字~~ 函數名字相同 → 執行不同程式碼

My Notes
Important Concepts worth keeping

Today / /

2-14

抽象化 ⇒ 好處：曾經產生過的資料型態 可在 reuse

2-16

scope resolution operator (::)
↳範圍 ↳指定那一個 class

using ⇒ allows the names of the elements to be used directly

using 內的有宣告的 class 都不需再 (class::)

- create

```
namespace smallNamespace
{
  int count=0;
  void abc();
} // end smallNamespace.
```

- use.

```
using namespace SmallNamespace;
count+=1;
abc();
```

暫停吸口氣，靜思問自己，關鍵 10 秒鐘，Ho 住十年功，【且慢功】

My Questions
Problems & Difficulties needing exploration

My learning weather report

try{ throw (type); ... } ≈ python try&except).
(故意type型態判斷)

```
catch (type1) {
}
catch (type2) {
}
.....
```

```
void myMethod (int x) throw(MyException)
{
}
```

↳ myMethod 只能拋出 MyException 類型的錯

throw() → 不會拋出異常

throw(...) → 可拋出所有異常.

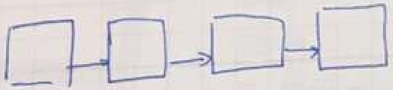
My Opinions
Thoughts, inspirations, and suggestions

慢比快還要快，《三采科說》

My Notes
Important Concepts worth keeping

Today: / /

3-01 鍊結串列



優：有彈性的擴充.

Insert ; delete

陣列 → 需搬動資料

Linked List → 不需搬動

3-02

指標 = 門牌

(型態)

(int *)p → p只是門牌

↳ 決定分配多少記憶体给 p 儲存的記憶体位置

p = new int → 有房子の門牌

↳ std :: bad_alloc 要不到記憶体

'p = &x

↳ 取 x の記憶体號碼

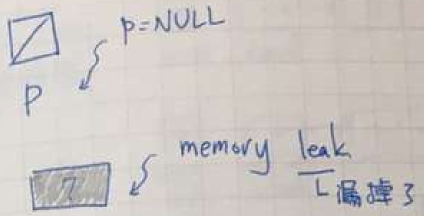
delete p ↔ p = new int.

p = NULL ↘

避免誤用已歸還の記憶体 (Avoid: dangling reference)

(非法)

My Questions
Problems & Difficulties needing exploration



3-04

double *anArray = new double[20];

要 20 個 double pointer

• an array name is a pointer to its first element.

delete [ ] oldArray ; (Array 內の記憶体都歸還)

My Opinions
Thoughts, inspirations, and suggestions

~~save / copy file~~

• open file

#include <cstdio>

```
int main(void)
{
    File *outfile = Null;
    string fileName = " DSsample.dat";
    outfile = fopen(fileName.c_str(), "a"); //open
```

使用這個檔案的模式

(a: 可寫可讀, 繼續 append)

My Notes
Important Concepts worth keeping

Today: / /

new ↔ delete

fopen ↔ fclose

3-06 Implement linked list

每一個 Node 由 struct 製作

每一個 Node 必包含2個部份

① 內容 (資料)

② 指向下一個節点 の pointer

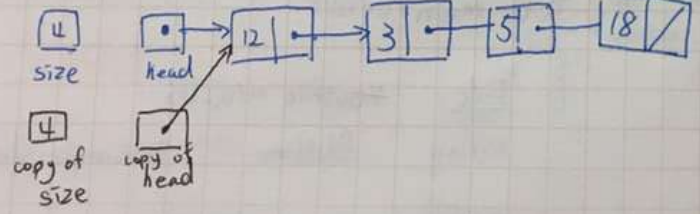
tail → Null

head → Null ⇒ linked list is empty.

動態配置 ⇒ destructor 要自己寫

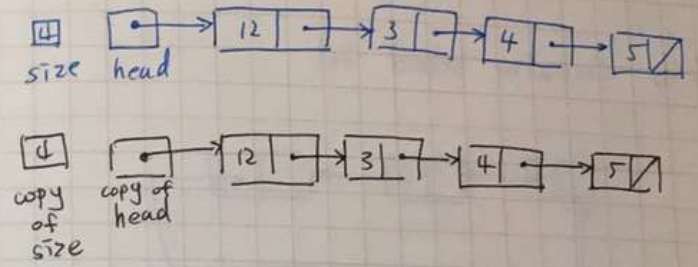
3-09 copy constructor

Shallow Copy 淺層複製



My Questions
Problems & Difficulties needing exploration

Deep Copy 深層複製



3-12

array	pointer
省空間	花空間
長度固定	長度可動.
retrieve → 快 (∵用index)	慢.
insert / delete 各有優缺	

My Opinions
Thoughts, inspirations, and suggestions

3-15 pointer 放在參數傳置時

是複製門牌號碼 ⇒ 修改會動到原本の值.

刪除、最後一個節点 (難)

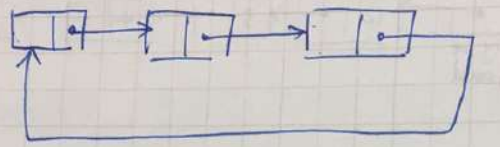
第一個 (易)

My Notes

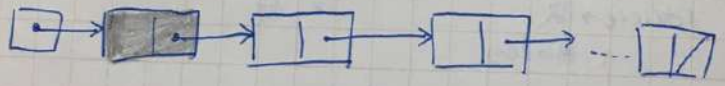
Important Concepts worth keeping

Today :

3-16 circular linked list



Dummy Head Node



⇒ 為了 eliminate the special case

The meaning is not in the words, but in the h

My Notes
Important Concepts worth keeping

CH 4.

Today: / /

x | y ⇒ x or y

xy or x·y ⇒ x 在 y 前.

<addition> : <digit> + <addition> | <digit>

C++ identifier 的第1個字必須為字母

<identifier> = <letter> | <identifier><letter> | <identifier><digit>

<letter> = a|b| ··· |z|A|B| ··· |Z|_

<digit> = 0|1| ··· |9

ex: A2C.

用遞迴, A2 判斷為合法的identifier 再遞回進 <identifier><letter>

可進到 base case 則整個 string 為合法 identifier

Source code 拆成 Token1, Token2 ··· ⇒ syntax 是否錯誤

A word fitly spoken is like apples of gold in baskets of silver.
《Proverbs》

My Questions
Problems & Difficulties needing exploration

Algebraic Expression

- Infix = <identifier> | (<infix><operator><infix>)
  An operator appears between its operands.
  ex: a + b
- Prefix = <identifier> | (<operator><prefix><prefix>)
  An operator appears before its operands.
  ex: +ab
- Postfix = <identifier> | (<postfix><postfix><operator>)
  An operator appears after its operands.
  ex: ab+

My Opinions
Thoughts, inspirations, and suggestions

一句話說得合宜，就如金蘋果在銀網子裡
《箴