**107\_1高中職程式設計基礎課程**

9/15 Topic0

•標準函式庫: 一些常用資料結構, 演算法 eg., Containers, iterators, …

•邏輯學(Logics):

1.命題，或稱為主張(Proposition) :是一種直接的敘述句(非疑問句)，不論該句的理念是對(True)的或是錯(False)的，但該句一定不會同時包含對與錯的兩種情形。

2.(1)E.g., 天空是藍的(為proposition)；1+1=2 (為proposition)

(2)E.g., y=x+1(不為proposition，因為y值和x在句中兩者無確切值無法確定，導致該句可能產生True或是False的可能性) 。

3.可以用小寫字母作為代表(p或q)

4.運算子(operator) :

⸣： 否定符號 • ⸣p, ⸣q

˄：和、且(AND 的意思) • p ˄ q：代表p和q

˅：或(OR的意思) • p ˅ q：代表p或q

⨁：不能相同(XOR的意思) • p ⨁ q：代表p和q不能相同

→：若…則… (if…then…) • E.g., p →q

↔：若且唯若(if and only if; iff) • E.g., p →q 且 q →p

5.用於分析規律結果的方法：真值表(truth table)

⸣： 否定符號

E.g., 命題：我是人類。 令p為我是人類，則⸣p為我不是人類。

˄：和、且(and 的意思)

E.g., 命題：青的蘋果是青蘋果。 令p為青色、q為蘋果，因此青蘋果只 在兩條件下都為真時成立。

˅：或(or的意思)

⨁：不能相同(XOR的意思)

E.g., 命題：明天我或弟弟要去掃廁所。 令p為我打掃廁所、q為弟弟打掃廁所， 因此打掃動作必然發生於至少有一個 人去打掃的情況下，才能合乎邏輯。

E.g., 命題：要嘛我吃魚，要嘛我吃牛 肉。令p為我吃魚、q為我吃牛肉，因 此，因為語句中我只能選擇之一樣， 因此只要有一者為真則結果為真，若 都要或是不吃則為假。

→：若…則… (if…then…)

E.g., 命題：如果下雨，天空一定有雲。令p為天氣是下雨、q為天空有雲，因 此，會下雨天空必然是有雲的，沒有下雨其天空也能會有雲，但也可能會沒雲；但是下雨天空卻沒有雲是不合邏輯的。

↔：若且唯若(if and only if; iff)

E.g., 命題：若且唯若冰淇淋是香草口味，小王會吃這個冰淇淋。令p為冰淇淋是香草口味、q為小王會吃這個 冰淇淋，則冰淇淋是香草口味的，那麼小王會吃這個冰淇淋；並且，如果小王吃冰淇淋，那麼這個冰淇淋就是香草口味的。

9/15 Topic1

•C++在高階語言中屬：

1.編譯式語言(compiled language):

(1)即會將原始程式碼轉換成中間碼至機碼

(2)與其相對為直譯式語言(interpreted language)，如Ruby

2.型態語言(typed language) :

(1)代表其變數需要定義型態

(2)與其相對為非型態語言(untyped language)

•註解:

1.// 雙斜線，此種方法只在該行後面才是註解，換行無效

2./\* 只要在這裡面的文字內容都算註解，包含換行 \*/

3.大小寫是不同(Case sensitive) :即代表大小寫是有區分的

4.命名規則

(1)不得是系統內的保留字或是識別符

(2)變數名稱必須要由字母、底線與數字構成

(3)變數名稱第一個字元必須為字母或底線開始

(4)最重要!! 必須賦予有意義的命名，千萬不要命名為x, y, z, a, b, c

•命名法:

1. 巴斯卡命名法 (Pascal Case) :單字之間不以空格斷開，第一個單字首字母採用大寫字母；後續單字的首字母亦用 大寫字母 • E.g., FirstName

2.駝峰命名法 (Camel Case) :第一個單字以小寫字母開始；第二個單字的首字母大寫 • E.g., firstName

3.匈牙利命名法 (Hungarian Notation) :一個變數名由一個或多個小寫字母開始，這些字母有助於記憶變數的類型和用途， 緊跟著的就是程式設計師選擇的任何名稱 • E.g., str\_FirstName

•一個專案中:

1.必有一個程式主體 main，並且以大括弧 {}包住，代表欲執行的指令

2.指令中涵蓋許多部分，但敘述 (statement)一定以分號 ; 結束

•函式庫(Library) :

1. C++標準函式庫

#include <函式庫名稱>

#include 將Library的內容加進程式內

稱為“預處理指令” (preprocessor directive)在編譯前，會先複製函式庫到程式中

1. C++ 有許多標準的內建函式庫

Input/output, math, strings, etc.予實作程式。

•命名空間(namespace):

1.可以區隔不同用途或範圍的變數常數

2.“std” 就是一個命名空間 – 裡面包含著標準函式庫的(識別符，identifiers) 定義

• E.g., 1: #include <iostream>

using namespace std; //包含整個std的命名定義

• E.g., 2: #include <iostream>

using std::cin;//可以指定命名空間內的變數作為使用

using std::cout; //可以指定命名空間內的變數作為使用

3.使用標準(standard)命名空間，以提供之後該些方法的定義

•識別符 Identifiers :

1.屬於變數(系統內)

2.依程式需求自行定義的名稱

3.程式中所用的各種名稱都屬於識別符

4.大小寫有別 (case-sensitive)

※若不在程式內使用namespace，則需要使用功能時，於原本有使用 std內識別符加入std 呼叫方法為：std:: 識別符名稱

※在C++語言裡面，不管檔案在多，其main只能有一個。

1.cout(念c out)，為輸出變數，其意義代表將由標準備於畫面中進行輸出

2.”<<”符號，為左移符號，會將橫面字串(雙引號所夾內容物)一個一個左移並交予輸出變數

3.endl為end line之意，代表之前動作結束後，將游標換行

•控制台輸入輸出 (Console I/O)：輸入輸出物件 cin, cout, cerr

1.定義在C++的<iostream>函式庫內

2.當需要使用輸入輸出時，必須要在程式最前面加上這幾行，告知C++需要使用這些物件 #include using namespace std; 或不使用上面方法，直接對識別符前加上std::

•cout：輸出

1.任何資料都可以輸出至螢幕上，包含: 變數 (Variables) 、常數 (Constants, Literals) 、運算式 (Expressions)

cout<< int\_numberOfGames<< “ games played.”; 則兩個數值會顯示在螢幕上: numberOfGames 變數的數值, “ games played.” 字串內容

2.可以有多個數值輸出在一個cout中

•cin：輸入(念c in)

1.不同之處: 在於後面的“>>” 即右移符號與cout 位移符號方向相反，把符號想成 >> 資料指向的方向 cin >> variable

2.將名稱 cout 改為cin – cin不能賦值給常數，因此 必須給予變數

cin >> int\_Num; //等待鍵盤上輸入內容，並將鍵盤輸入值存入int\_Num中

1. 通常在輸入資料前，最好先在銀幕輸出提示: cout << "Enter number of dragons: "; cin >> numOfDragons;

4.注意在cout內沒有換行符號’\n’，因為終端機需要在同一行等待鍵盤的輸入內容:

Enter number of dragons: \_\_\_\_ 底線為鍵盤輸入之內容

5.每個cin的內容應該需要有cout的輸出提示以讓使用者有足夠訊息操作

•cerr：錯誤輸出(念c error)

1.使用方法與cout相同，主要於除錯時使用。

2.cout可以重新轉向到其他設備，cerr卻只會於銀幕輸出

3.cout會先將資料儲存到緩衝區（buffer）中，而cerr則是直接將資料輸出，雖然會增加I/O的操作次數，減低程式的效率，但也使得程式設計師在程式開發除錯階段， 能更即時的接收到適當的輸出結果。

※int為整數型態，因此回傳整數(使用者可自行定義，其用意在於除錯用)

9/22 Topic2