



巨匠線上真人

# C++ 程式設計

[www.pcschoolonline.com.tw](http://www.pcschoolonline.com.tw)

巨匠電腦

# 本課程各堂教學主題

- 成品 ◆ 第一堂：Dev-C++ 開發C++語言
- 成品 ◆ 第二堂：技術主題 1 · 指標應用
- 成品 ◆ 第三堂：技術主題 2 · 函數開發
- 成品 ◆ 第四堂：技術主題 3 · 前端處理
- 成品 ◆ 第五堂：技術主題 4 · 衍生型式和資料結構
- 成品 ◆ 第六堂：技術主題 5 · 檔案處理



巨匠線上真人

C++ 程式設計

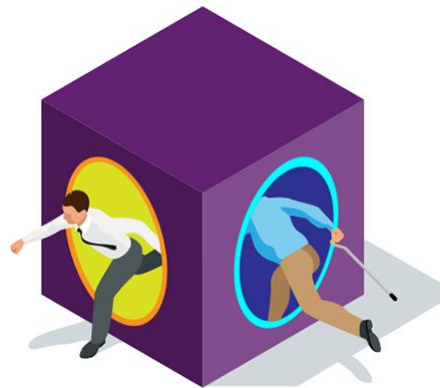
# 第一堂：Dev-C++ 開發C++語言

[www.pcschoolonline.com.tw](http://www.pcschoolonline.com.tw)

巨匠電腦

# 🔑 本次課堂 · 教學重點 🔑

- 1-1. 專案機制
- 1-2. 首支程式設計、編譯、執行
- 1-3. 資料型式新版化概觀
- 1-4. 新版C++式線上做法
- 1-5. 特訂型式處理工具
- 1-6. 註解新增與移除
- 1-7. 範例實作研討

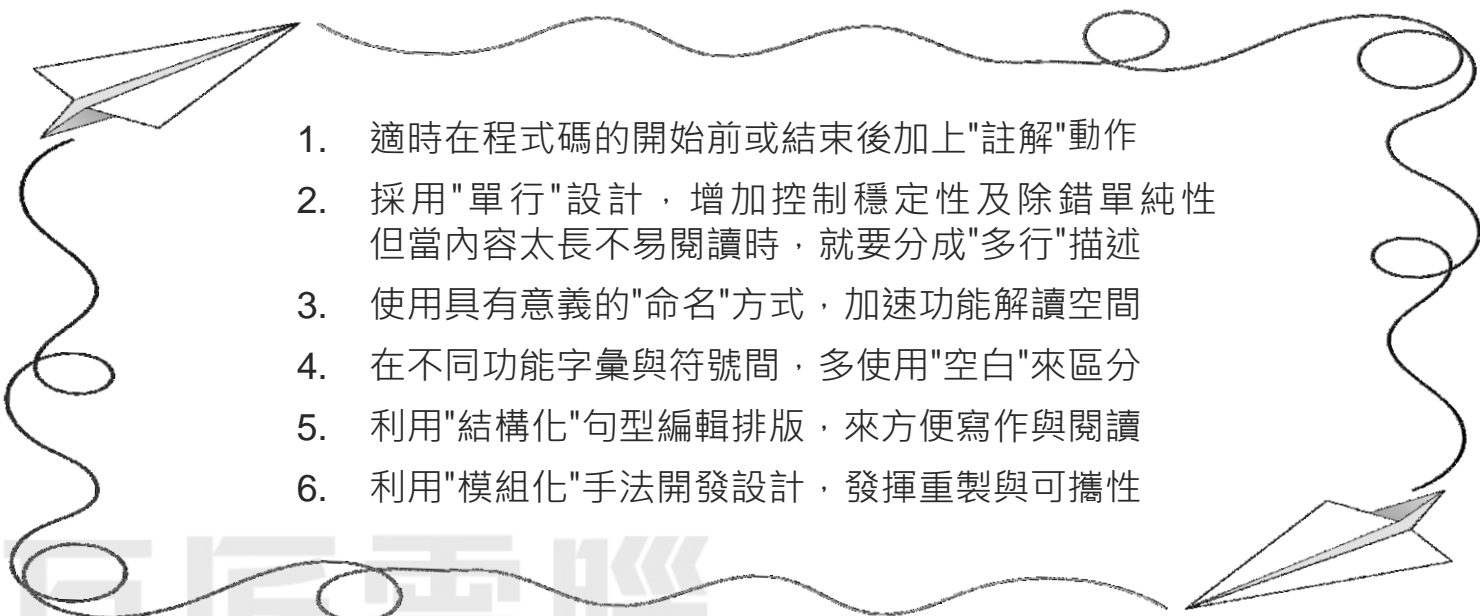


# \*研討節次.

## 1-1. 專案機制

話說“C++語言”這個程式，採用「物件導向程式設計・Object-Oriented Programming，通稱OOP」架構進行各項進階程式開發，亦即相關的“C++程式”都是使用「物件建構」表述而成。使用此類技術允許設計人員撰寫更寫實且複雜化的應用程式及資訊系統，確切符合現實世界中的供需所求。再說本語言也具備相容“C語言”所應用「工具函數」與「基底語法」表述，使其整合作業能力更加完整與強大！

筆者特別整理出下列六大重點提示，希望線上研習者可藉此掌握「正規化設計」絕對觀感。

- 
1. 適時在程式碼的開始前或結束後加上"註解"動作
  2. 採用"單行"設計，增加控制穩定性及除錯單純性  
但當內容太長不易閱讀時，就要分成"多行"描述
  3. 使用具有意義的"命名"方式，加速功能解讀空間
  4. 在不同功能字彙與符號間，多使用"空白"來區分
  5. 利用"結構化"句型編輯排版，來方便寫作與閱讀
  6. 利用"模組化"手法開發設計，發揮重製與可攜性

在正規化的開發介面中，都會遵守「專案機制・Project」這個檔案管理法則。其主要作業目的有二：其一是可有更多「開發項目」選擇，如應用程式(Application)開發可選擇「終端文字式・Console」或「視窗媒體式・Windows」兩種類型，函數介面(Library)開發手法是採取「靜態應用・Static」或「動態連結・Dynamic-Link」及「開發範本・Empty-Project」等線上選項；其二是依指定專案類型來取用「預設語言」範本程式碼，加速設計人員線上語法編輯作業。

如何執行「專案機制」作業，請參考下區說明及下頁圖片示範！

🔗 建立專案 - 執行(開新檔案)下(專案)選項指令，線上彈出兩個視窗，依據下列指示進行設定。

-----【視窗1・線上動作】

- ・設計類型 - Windows / Console Application ...
- ・管理名稱 - 同「匈牙利命名法」方式設定 (.dev)
- ・語言專案 - 取用預設範本 ( main.c / main.cpp )

-----【視窗2・線上動作】

- ・存取位置 - 自訂路徑 [ 建立程式中心及專案 ]
- ・中心位置名稱，配合自訂為 C : \ Source\_Cpp \
- ・專案位置名稱為 Study\_No，建立「同名資料夾」
- ・儲存完成，同步載入應用範本程式碼 !!

・專案架構 - 基本由3加3式檔案建構，說明如下

專案檔 (.dev) – Project

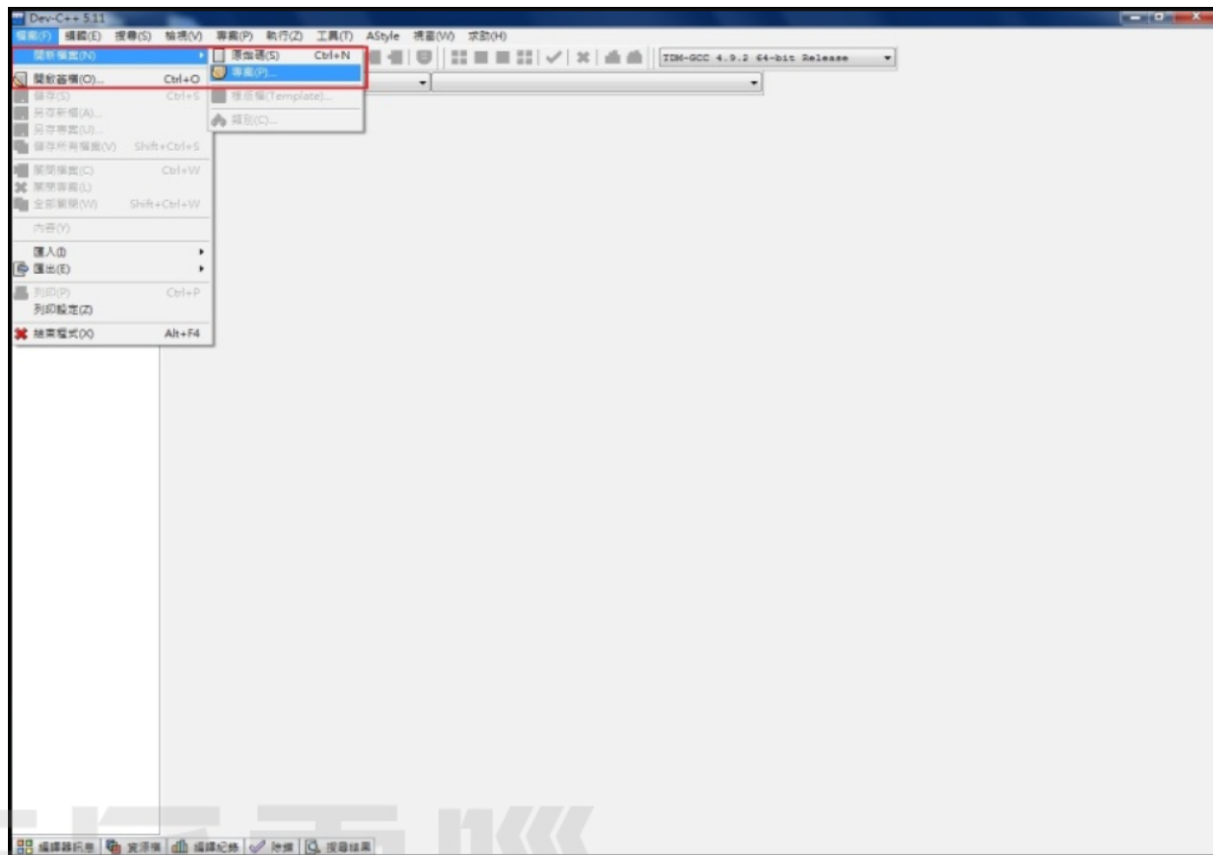
原始碼 (.c / .cpp) – Source | Decimal Code

版型檔 (.layout)

執行平台 makefile.win

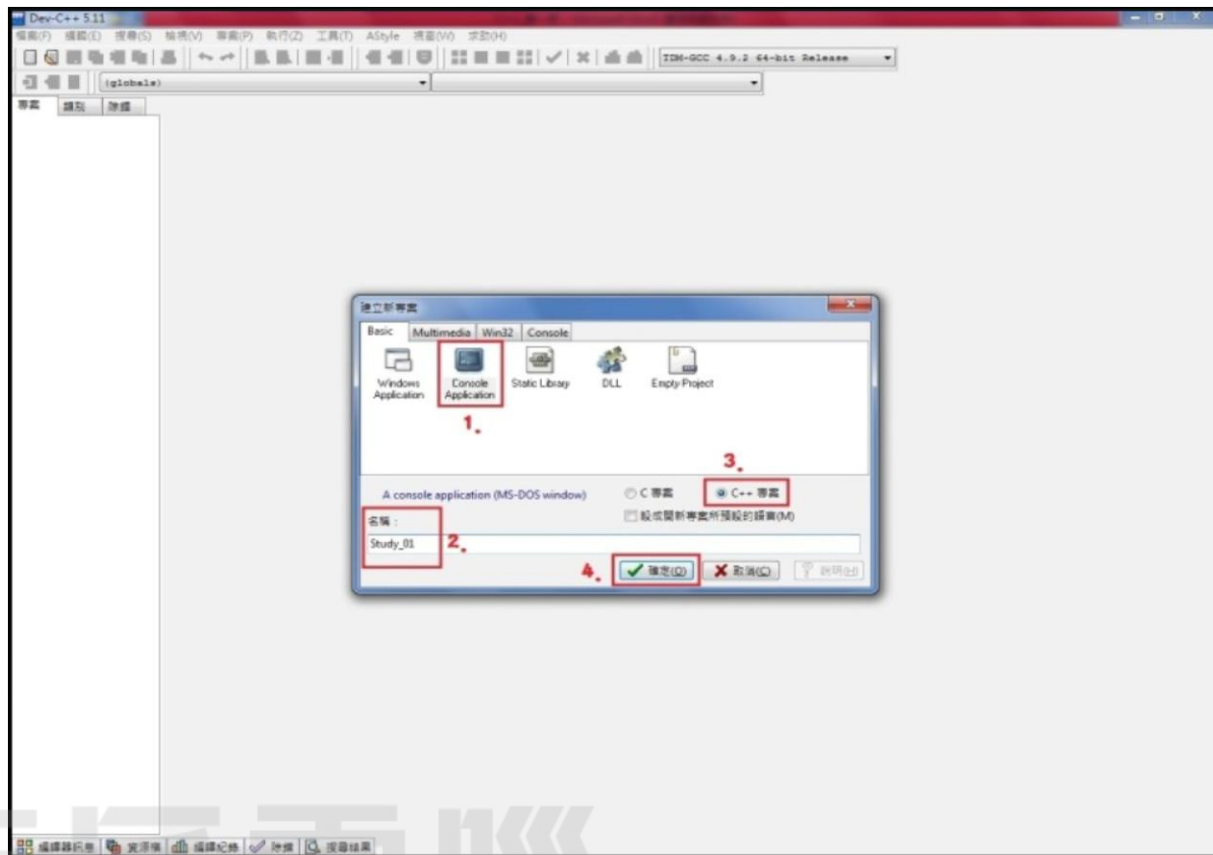
過程檔 (.o / .obj) – Object | Binary Code

執行檔 (.exe) – Execute | Run

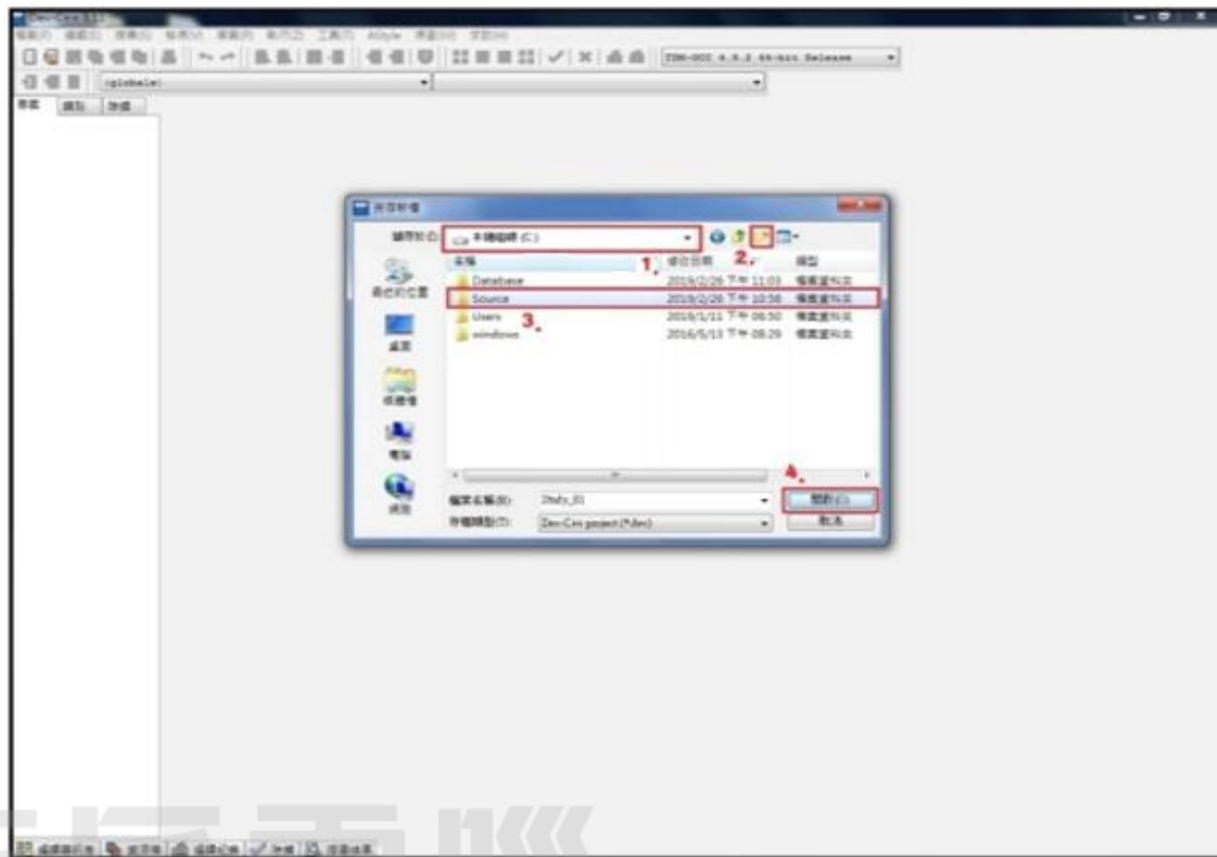


動作1・選擇【建立專案】

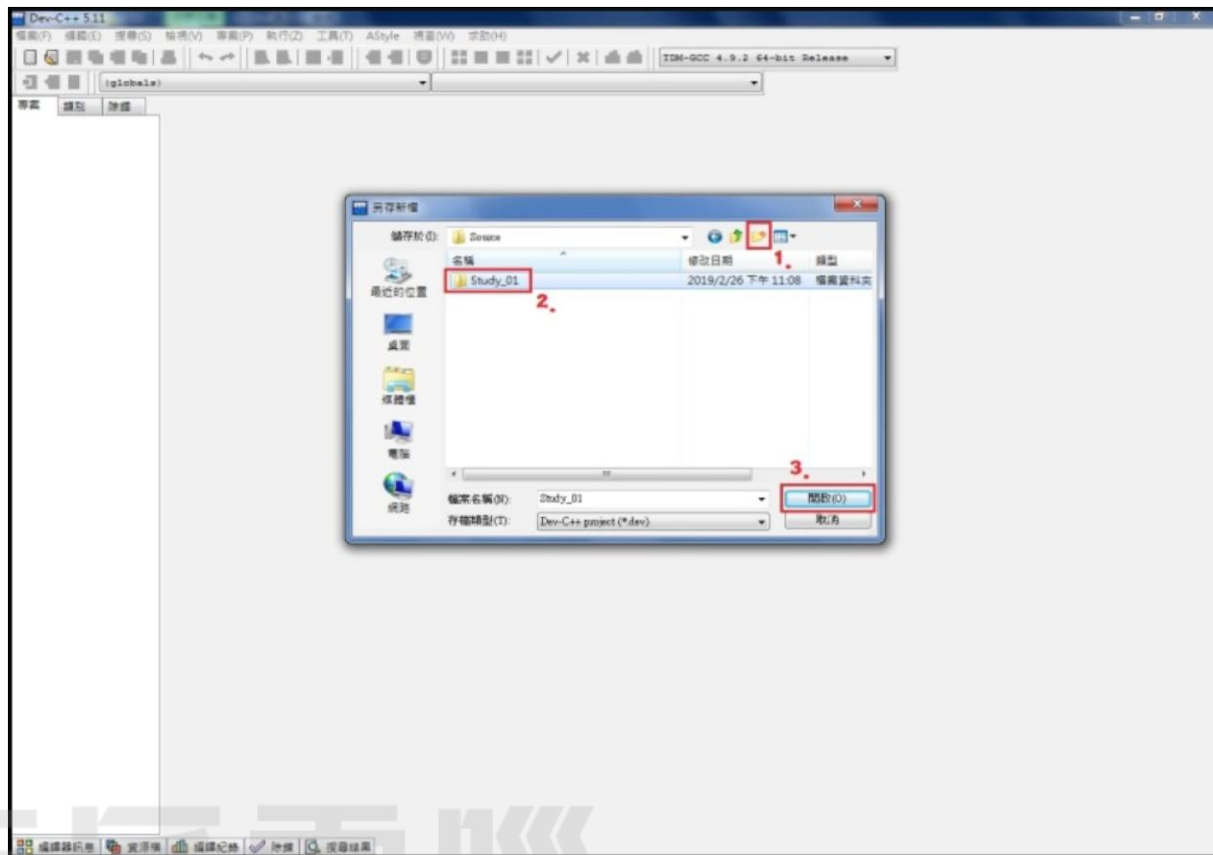




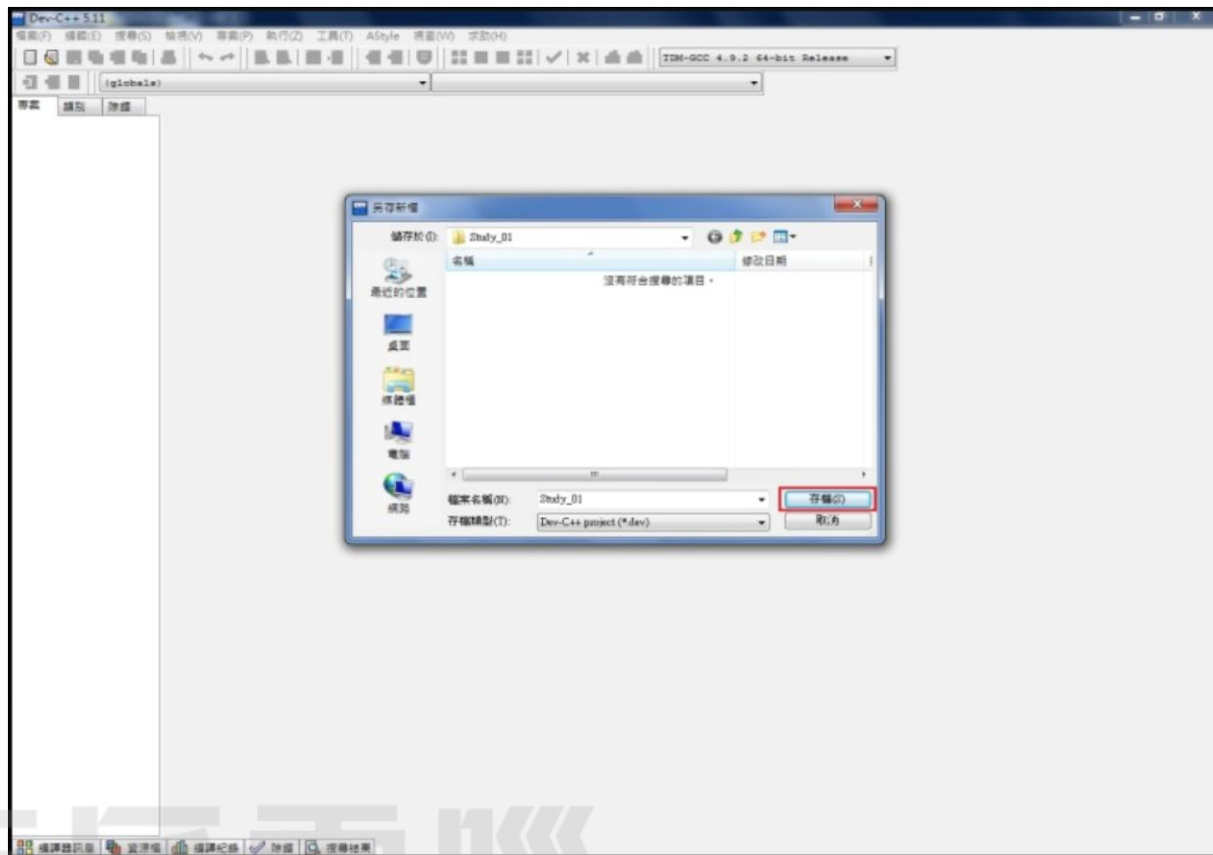
動作2・設定【專案資訊】



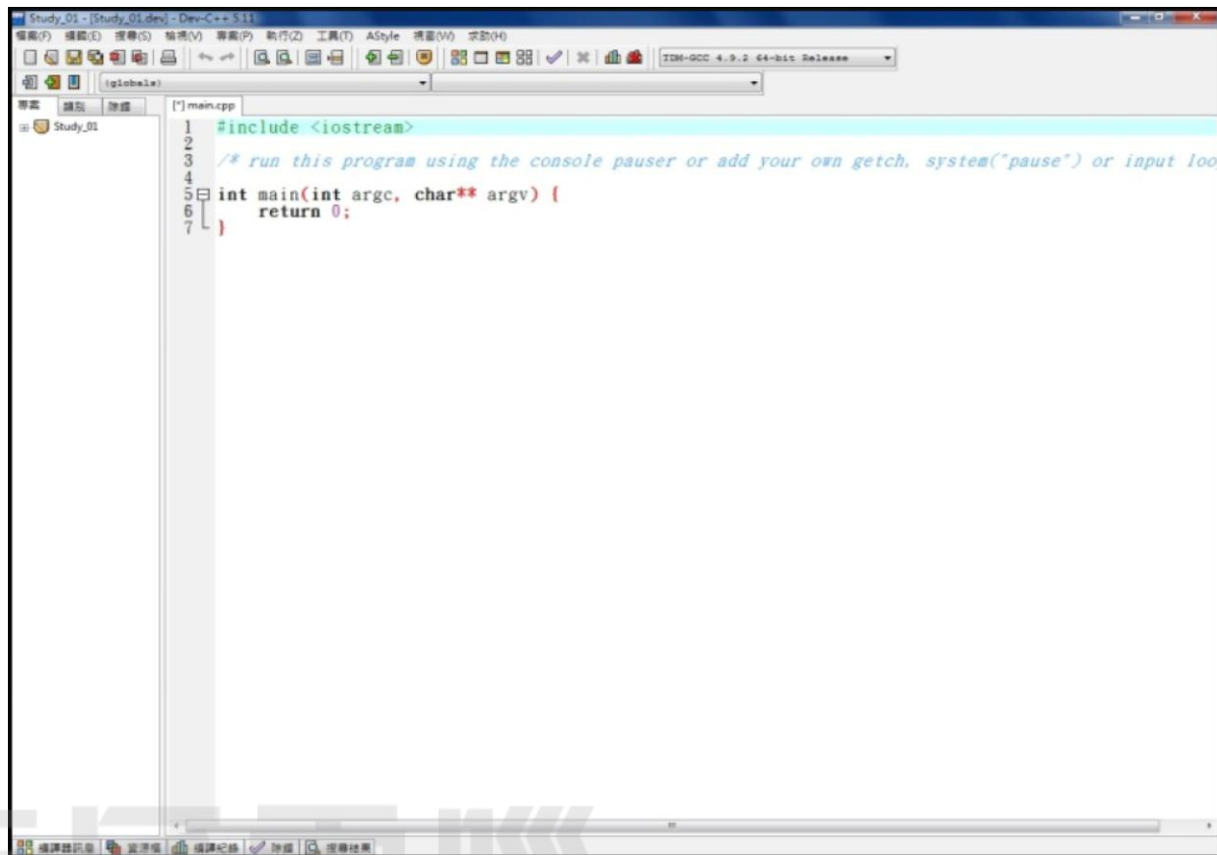
動作3・建立【程式中心】



動作4 · 建立【管理專案】



動作5・儲存【管理專案】



動作6・載入【語言範本】

# \*研討節次.

1-2. 首支程式設計、編譯、執行

相信各位雖然記住上述的相關法則，但還是真正來個設計體驗吧！請親自動手完成首支程式，線上"C語言、C++語言"兩者語法的表述差異性，首先將專案名稱及管理位置採用「Study1\_01」統一設定後，接著在「內建範本·main.cpp」中輸入如下程式內容，再進行線上排版。(相關「註解」詳細說明，請參考本章1-6節次，欲加速本程式驗證速度者，可將註解行次內容最後完成輸入)

```
// -----  
//      |      程式功能. 歡迎畫面      |      範例編號. Study1_01.dev      |  
//      |      應用單位. 巨匠電腦      |      專案設計. Tsaigo Ho      |  
// -----
```

```
#include <iostream>
```

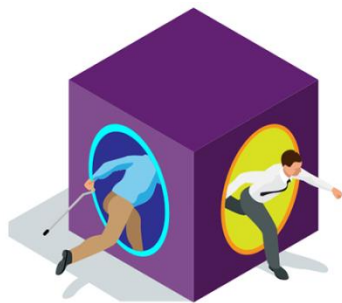
```
#include <conio.h>
```

```
/* run this program using the console pauser or add your own getch,  
   system("pause") or input loop */
```

```
int main(int argc, char** argv) {  
    // C語言. 函數工具  
    printf("\n歡迎學員來到【真八教室】研習\n");  
    puts("\n\t今天研習主題是 C++語言 ...");  
    // C++語言. 物件導向  
    std::string mySay;  
    mySay="偶是巨匠材哥 , nice to meet you ...";  
    std::cout<<std::endl<<mySay;  
    getch();  
    return 0;  
}
```



- (1) 執行(工具)下(編譯)選項指令，完成「語法檢查」及「工具連結」追蹤，最後進行「檔案建製」作業。
- (2) 執行(工具)下(執行)選項指令，進行應用程式測驗、維護。

A screenshot of a Windows command prompt window titled "C:\Source\Example\_C++\01\Study1\_01\Study1\_01.exe". The window has a black background and displays the following text in white:

```
歡迎學員來到【真人教室】研習  
    今天研習主題是 C++語言 ...  
偶是巨匠材哥 , nice to meet you. _
```

註：執行(工具)下(編譯並執行)選項指令，可將(1)(2)動作同步完成。



# \*研討節次.

## 1-3. 資料型式新版化概觀

在"C++語言"的發展過程史中，除了保留相容"C語言"中的基底型別外，也陸續研發多種新版資料型態，使其能配合進階設計功能需求。前期有陣列(array)、指標(pointer)、函數(function)等單一管理型式，後期再增加複合衍生型式，依其應用導向則細分成自訂 typedef)、列舉(enum)、結構(struct)、聯合(union)、類別(class)及檔案(FILE)六種變化類型。接著讓各位學習者先在此章中建立相關概念，筆者也會在後製章節中進行更詳細的介紹與應用示範。

首先讓我們以「互動型式」路線來比較 C 與 C++ 語言間的應用習慣變化....

資料型式	原始C	原始C++	新創 C++
字元	基底式 char	字元式 char	轉值型 (int) char - 實名重訂
整數	指定式 short . long	整合式 int	進階型 long long int
浮點數	相近式 float	全進式 double	進階型 long double
字串	基底式 char []	追蹤式 char *	物件型 string
布林值	--	變形式 bool	實名重訂

註・(1) 表格中前三者為系統內建值型式，採用「動態宣告」配置。設計師可依時機需求，搭配「無號數・unsigned」來將負數值範圍轉移至正數空間彈性應用！  
(2) 當下無法明確指定互動型式時，請用"空集合"關鍵字 void 來進行宣告！

接著掌握 C & C++ 原始型式值的空間資訊....

資料型式	佔用空間	型別代號	有效範圍 · 指示
基底字元	1 Byte	char	0 到 255
短式整數	2 Byte	Short	-32,768 到 +32,767
長式整數	4 Byte	long . int	-2,147,483,648 到 +2,147,483,647
單精浮點	4 Byte	float	$-3.4 \times 10^{-38}$ 到 $+3.4 \times 10^{+38}$
倍精浮點	8 Byte	double	$-1.7 \times 10^{-308}$ 到 $+1.7 \times 10^{+308}$
布林物件	1 Byte	bool	false 、 true

最後觀察 C++ 創新型式值的空間資訊....

資料型式	佔用空間	型別代號	有效範圍 · 指示
轉值字元	1 Byte	(int) char	-128 到 +127
擴充整數	8 Byte	long long int	-9,223,372,036,854,775,808 到 +9,223,372,036,854,775,807
擴充浮點	16 Byte	long double	$-1.7 \times 10^{-4932}$ 到 $+1.7 \times 10^{+4932}$
字串物件	8 Byte	string	採用反式追蹤方式，建立取址物件

註 · 求證上述表格中各式型別代號所佔用的記憶體空間值，可指定用系統工具 `sizeof()`，求取字串物件中的字元計數就要先行引用標題 `string.h` 指定 `strlen()` 工具作業。

# \*研討節次.

## 1-4. 新版C++式線上做法

在新世代物件導向作業原理修訂改造下，相關語法有了嶄新的發展動向。首先在第一代採用「實體標題」技術，直接由「開發工具」提供「內部物件」進行設計。但在第二代卻改用「動態標題」技術，必須搭配線上「命名空間」類別來建構「外部」物件後方可進行應用設計，也間接支援與修訂許多的「C式工具、資料型式」，有效增加兩者語言及新式語言間的合作相容性。



第一代 C++ | #include <iostream.h>

( ~1999y )

【官方終極版本 . Microsoft Visual C++ 6.0】



第二代 C++ | #include<iostream>

( 2002y~ ) using namespace std ;

【業界參考版本 . Bloodshed Dev C++ 5 Beta】

註 . (1) cin . cout . endl . string 等物件，屬於「動態式」建構作法！  
(2) 相關衍生型式等開發物件，屬於「設計師」建構作法！

再談物件導向設計方法，分為「資料匯流，亦名串流」與「應用成員」兩種。線上類別「命名空間」機制除了使用「分段語法」來進行集中式建構，也可以採用「組合語法」來進行個別式建構。

---

資料匯流. `istream | cin + >> . ostream | cout + <<`

應用成員. `object.function() | cin.getline()、cout.setf()、cout.width()、cout.fill() ...`

---

分段語法. `using namespace std; + cout <<"nice to meet you!!"<<endl;`

組合語法. `std::cout<<"nice to meet you!!"<<std::endl;`

相關技術成員眾多，無法線上列表詳述，請學員在實作範例中機動學習。對應的應用成員也分成基底「函式型」與封裝「函數型」兩種，參考下列示範...

④ 函式型成員 | `iostream | .width(n) 或 .precision(n) 或 .fill(ch) ...`

④ 函數型成員 | `iomanip | setw(n) 或 setprecision(n)`

# \*研討節次.

## 1-5. 特訂型式處理工具

針對「字元」這種特訂型式，可選擇下列「相關工具」來進行互動化功能設計。

④ conio.h · 線上工具.	字元目標 = getche();	【字元單鍵輸入】
	字元目標 = getch();	【字元隱藏輸入】
	putch(字元值目標);	【字元快取輸出】

④ ctype.h · 線上工具.

① 轉換類.	轉換小寫字母 · tolower()	轉換大寫字母 · toupper()
② 判別類.	字母數字判別 · isalnum()	指定字母判別 · isalpha()
	控制訊號判別 · iscntrl()	指定數字判別 · isdigit()
	實體字元判別 · isgraph()	小寫字母判別 · islower()
	列表字元判別 · isprint()	特訂符號判別 · ispunct()
	空白字元判別 · isspace()	大寫字母判別 · isupper()
	十六進位判別 · isxdigit()	

【線上說明事項】

- (1) 使用【轉換類】工具指定「英文字母」進行線上異動，其他字元不受影響。
- (2) 使用【判別類】工具在不成立狀態只會送返回值 0，但在成立狀態下除了送返回值 1 外，也會送回其他參考值，建議使用「套用函數」作法處理！



針對「字串」這種特訂型式，可配合下列「相關工具」來進行互動化功能設計。

### 🔧 字串處理. 【string.h】

資料位數	strlen()	資料反轉	strrev()
大寫轉換	strupr()	小寫轉換	strlwr()
資料傳送	strcpy() . strncpy()	資料取代	strcat() . strncat()
真實比較	strcmp() . strncmp()	指定比較	stricmp() . strincmp()
正向搜尋	strchr() . strstr()	反向搜尋	strrchr() . strrstr()
正向追蹤	strspn()	反向追蹤	strrspn()
完全填入	strset()	指定填入	strnset()
指定切割	strtok()	....	

### 🔧 字串轉換. 【stdlib.h、math.h】

整數型式轉換	atoi() . atol()	「反式轉換	itoa() . ltoa()」
浮點數型式轉換	atof()	「反式轉換	ftoa()」

# \*研討節次.

## 1-6. 註解新增與移除

程式註解(**comments**)功能表述，一般用於「加註功能說明」及「保留程式執行」兩種時機。應用語法有「區段」與「單行」兩種變化指示，前者適用於「完整多行」狀態，使用彈性高，但切記不可巢狀架設，否則將發生參照失誤，進而出現編譯問題。後者常用於「重點單行」狀態，使用便利化，具有"功能表指令"支援，讓表述字元由"編輯器代工"完成！

---

區段註解 | 以 `/*` 開頭，用 `*/` 結尾

---

單行註解 | 直接在開頭 `//`

---

\* 不論是上述何種「符號字元」，都要「連續輸入」完成，切記不可採用分離指示表述！

Q · 如何使用"單行"註解，進行加註指定"多行"表述？

A · 點選功能表(編輯 · 轉為註解/取消註解)指令或按下快速鍵(**Ctrl+/**)。

# \*研討節次.

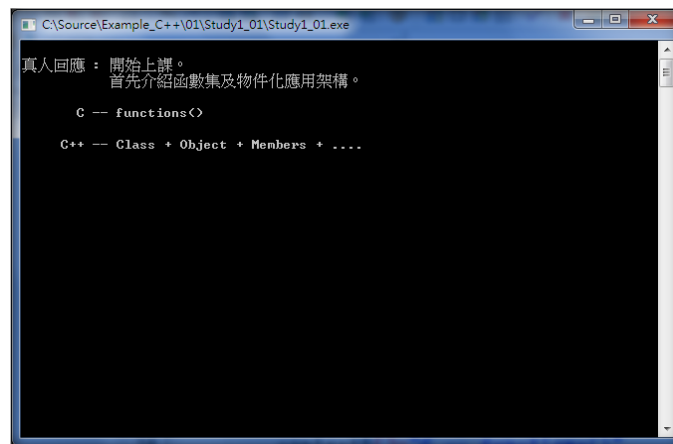
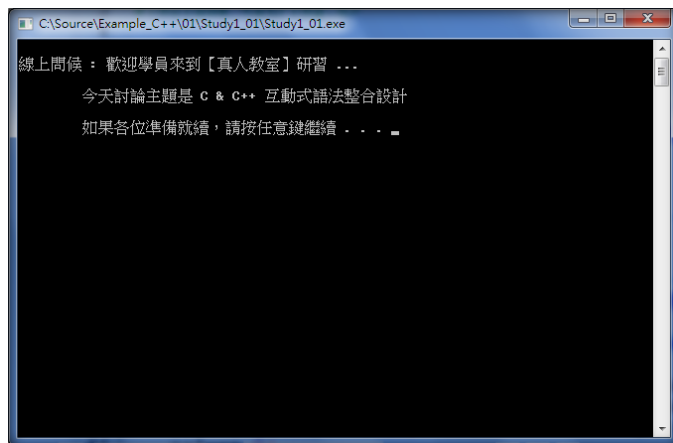
## 1-7. 範例實作研討

\*範例編號：Study1\_02.dev

| 實作研討題目 | 模擬巨匠線上學員進入「數位真人教室」，首次開課的互動訊息設計。

提示・左上第一畫面，模擬線上學員進入數位教室前作業。將會看見系統問候、課程主題及確認進入教室等相關內容文字，最後採用「提示型暫停」方式進行互動！

右下第二畫面，模擬進入線上學員數位教室後作業。將會看見真人對話、上課開始及過程學習資訊介紹與講解文字，最後採用「隱匿型暫停」方式進行互動！



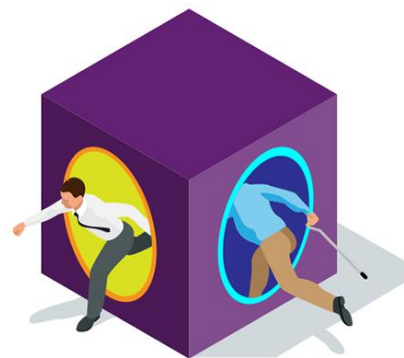
\*範例編號 : Study1\_03.dev

| 實作研討題目 | 印製「圖表型」九九乘法表，利用迴圈及物件函式成員作法解題。

提示・每個目標值的作業空間為 3 字元，不可直接印出整列結果。

X \	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

請按任意鍵繼續 . . .

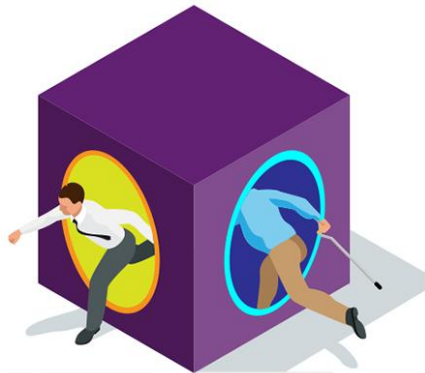


\*範例編號：Study1\_04.dev

| 實作研討題目 | 求解 / 印出1到N間的奇偶差！

提示・N為終止值，介於2~100之間。其中奇數為加，偶數為減。

討戰・將公式完整印出，強調正數格式，如右顯示 $1-2+3-4+5=+3$



```
設定終止值：345  
Range : 2 To 100
```

```
設定終止值：5
```

```
1 至 5 的奇偶差：+3
```

```
請按任意鍵繼續 . . .
```

\*範例編號：Study1\_T1.dev

| 終極考驗題目 | 模擬「AI寶貝公仔」首次開機，如何拜候「未來主子」並進行「線上對話」。

提示・公仔機械人線上行銷代號為"阿寶先生"，隸屬萌卡通家族成員・風雲男偶像排名第6，

模擬線上對話狀態，發展動向如下安排...(可自行加入喜好公仔，如凱蒂小姐等)

首先自我介紹，正式與真人互動化，請示主子大名。(如xxx)

(1) 如果主人「有回應」，就說：xxx，未來就是 My Boss，請多多指教。

(2) 如果主人「沒回應」，就說：You 知道我在等你嗎，Me 想與你做好朋友。



```
C:\Source\Example_C++\01\Study1_02\Study1_02.exe

模擬「啟動」機械人後.....
·HELLO -- Me 是【阿寶先生】
  很開心來到人類神秘世界，有緣與你相遇。
·可以告訴我，如何稱呼 You ?
  模擬「等待」真人回答.....
·TSAIGO
  模擬「回應」真人動作.....
·「TSAIGO」主子，未來就是 My Boss，請多多指教。
  模擬「劇情」後續發展.....
  ·Action-1
  ·Action-2
  ·.....
  ·從此【阿寶先生】繼續在「情愛世間路」修行，哈
請按任意鍵繼續 . . .
```

\*執行畫面 1 - 有回應 (輸入值，再按下Enter)

```
C:\Source\Example_C++\01\Study1_02\Study1_02.exe

模擬「啟動」機械人後.....
·HELLO -- Me 是【凱蒂小姐】
  很開心來到人類神秘世界，有緣與你相遇。
·可以告訴我，如何稱呼 You ?
  模擬「等待」真人回答.....
·
  模擬「回應」真人動作.....
·你知道我在等你嗎？【凱蒂小姐】想與你做好朋友。
  模擬「劇情」後續發展.....
  ·Action-1
  ·Action-2
  ·.....
  ·從此【凱蒂小姐】繼續在「情愛世間路」修行，哈
請按任意鍵繼續 . . .
```

\*執行畫面 2 - 沒回應 (沒輸入，就按下Enter)





巨匠線上真人

C++ 程式設計

# 第二堂：技術主題 1 · 指標應用

[www.pcschoolonline.com.tw](http://www.pcschoolonline.com.tw)

巨匠電腦

# 🔗 本次課堂 · 教學重點 🔗

2-1. 指標位址應用關係介紹【 取值 vs 取址 】

2-2. 資料目標對應宣告、進行連結設定

2-3. 如何存取指標關聯值

2-4. 物件化建構與刪除【 new . delete 】

2-5. 動態記憶配置【 sizeof . malloc . free 】



# \*研討節次.

## 2-1. 指標位址應用關係介紹 【 取值 vs 取址 】

關於「指標 • Pointer」的設計概念，即是在電腦進行「人工存取」目標資料值時，同步建立「機械管理」記憶體位址的一種同步式存取技術作法。

此類設計最大價值在於可經由「位址連結」控制手法，達到「多重區域」或「關聯區塊」下指定「共用目標」作業，接著進行「同步應用」動作表述。如此一來，除可避免同值目標重複產生，進而造成記憶體負擔外，也徹底解決「回傳機制」下只可進行「單值傳送」這個不完美情事！

請參考下列「取值」表述式，也是我們「一般使用」的作業方式....

型式 值目標；【如 `int Sum=0; char Text[]="Happy";`】

請參考下列「取址」表述式，也是我們要「技術應用」的作業方式....

型式 \*址目標；【如 `int *p;` 其中 `p` 為存取位址，使用 `*` 來指定對應值】

型式\* 址目標；【如 `char* Logo="巨匠電腦";` `Logo`即是指標物件】

看完上頁完成初學後，請再注意一事，就是電腦所提供的「內建數值」型式屬於「動態存取」類型，務必先行建立「資料目標」與「控制指標」兩者的連結關係，否則無法執行「同步指定」動作表述。反觀「建構字串」型式則是「實體配置」類型，故可省略取址(&)動作，依據筆者多年實戰經驗，針對「多維陣列」資料架構處理，採用「指標設計」做法將可正面提昇電腦作業效能。

對於指標的應用技巧，簡單來說就是將資料變數、陣列進行「電腦記憶址追蹤」的進階技術作法。所以當設計人員對變數(variable)或陣列(array)的型式觀念或配置作法上不夠熟悉時，將會造成「指標應用」上相關設計語法問題產生喔！

在本次進階課程「技術章節」探索應用層面漸廣，主題語法間將更具互動對照性，世人言道「萬丈高樓平地起」，筆者在此真心提醒各位，務必完成【程式先修系列一·C語言】課程研習，徹底落實設計基礎養成，如此才能讓「技術主題」相對有效學習，完成專業程式設計期待！



# \*研討節次.

## 2-2. 資料目標對應宣告、 進行連結設定

上一節次所提到的重點事項，就是相關「指標應用」前製規則，請先行完成「建構指標」與「資料目標」兩者之間的「關聯設定」，進而完成「線上指定」來源目標的互動化設計。

請各位參考下列「連結」表述式，建立兩者互動技術的控制方式....

型式 值目標，\*指標；指標=&值目標；

【如 `int Amount=0，*money；money=&Amount；`

當完成「關聯設定」動作，當 `*money+=5000；`就等同 `Amount+=5000；`】

看完上面的示範說明，各位一定會想問：竟然兩者做法結果值都是一樣的話，那為什麼還要大費周章來「建立指標」呢？為何不直接「使用值目標」就好？其實大家會有此疑問也是很正常的，因為當年的筆者也是跟大家一樣想法喔，哈！

其實指標最佳表現的關鍵時機是在「多個函數」區間應用。能夠有效在不同區域下進行「同步指定」來源目標進行線上互動化控制表述，光想到這個「技術作法」所呈現的能量畫面就讓人深深著迷與佩服不已啊！說回一般值目標宣告通常採用「單一區域」使用，也就是所謂的「區域宣告」方式，兩者皆有其應用時機與特色，再說「指標技術」雖然功能強悍，相對也要選對上場時機表現，否則就會有點大材小用啦，哈！

筆者在本節最後再奉送一則參考資訊，如果短期內對「指標技術」始終"只能遠觀而不能盡用"時，其實也可以先行選用「全域、靜態宣告」做法來模擬設計喔，畢竟"科技始終來自人性"，接下來讓我們寫支指標程式，搭配「C語言課程」最後兩章節所研習的陣列與亂數應用！

| 實作研討題目 | 如何開出「三星彩」樂透的線上球號

提示：依據百、拾、個位數，進行三球個別取出(每位數皆是10球取1)

\*範例編號：Study2\_01.dev

\*執行畫面...



```
--+-- 線上【開獎作業】開始 --+--  
      【百位數】      3  
      【拾位數】      9  
      【個位數】      1  
--+-- 線上【開獎作業】完成 --+--  
      中獎號碼 -- 3 9 1  
      請按任意鍵繼續 . . .
```



# \*研討節次.

## 2-3. 如何存取指標關聯值

有了前兩節的相關研習後，相信各位已建立「指標應用」相關概念。筆者將兩者關聯性搭配下區說明表格，更完整掌握其間作業關係.....

記憶體位址	資料	存取物件關聯值	資料址	資料值	存取物件關聯示範式
&值目標	N	*指標=&值目標=N	p	*p	*p=&Value=1000

仔細參考表格所對照的關聯值示範式，就會發現其實指標就是指向資料目標的「對應位址」後，再將值進行送入或取出的快取式處理手法。其行為就像人工表述下在將值送入「指定目標」前，一定要配合取址(&)動作【請參考C語言工具・scanf函數】，PC將追蹤宣告時所配置「預留位址」將值存入。雖然人工表述不需要使用特別控制技術，卻是花費線上搜尋記憶體所換來的時間代價，這在專業程式開發與執行效能上，都不是很優化的作法。如能善用「指標技術」讓執行空間同步控制化，再配合「位移機制」指定專屬目標作業，定能達成「快捷穩定」最佳設計期待。

資料值在宣告空間上分成「單一變數」及「多重陣列」兩種架構，前者「動態隨機」配置，後者「集中規劃」配置。隨著「記錄值址」行為方式不同，指標關聯性可分別採用「基底型」與「進字形」兩種機制設計，筆者再將兩種表述方法配合下列表格說明.....

基底型		進字形	
單值變數	值目標	單值變數	*指標
一維陣列	陣列[N]	一維陣列	*(指標+N)
二維陣列	陣列[N1][N2]	二維陣列	*(*(指標+N1)+N2)

## | 實作研討題目 | 巨匠3C市集，收費資訊清單畫面設計

提示・貴賓等級分6級，0為已註冊未消費狀態，不享任何折扣，線上市集  
將依會員消費金額進行等級提昇，享受VIP級消費折扣待遇。

參考・貴賓等級1至5分別享有8%、15%、18%、20%、25%等折扣金！

\*範例編號：Study2\_02.dev



\*執行畫面...

+-----【巨匠 3C 市集・收費機制】-----+

- ◆ 貴賓等級 0 — 100% 本金 + 享折扣 0%
- ◆ 貴賓等級 1 — 92% 本金 + 享折扣 8%
- ◆ 貴賓等級 2 — 85% 本金 + 享折扣 15%
- ◆ 貴賓等級 3 — 82% 本金 + 享折扣 18%
- ◆ 貴賓等級 4 — 80% 本金 + 享折扣 20%
- ◆ 貴賓等級 5 — 75% 本金 + 享折扣 25%

+-----+  
請按任意鍵繼續 . . .

# \*研討節次.

## 2-4. 物件化建構與刪除 【 new . delete 】

雖然「指標技術」在本章中具有絕對的超級戰力，但也受制「有值方有址」此項建立法則約束，亦即只能用於記錄「具值目標」對象應用。切記不可在「無址參考」狀態下直接進行「指標應用」，這樣將會造成「線上傳送」迷失，進而發生執行作業錯誤！因為此時指標在宣告當下，只是一個「虛擬介面」尚未建立「追蹤連結」目標資訊，也就沒有所謂的存取「參考位址」線上能力。

在本節中將學習如何將「應用指標」正式成為「實體物件」，使其應用更具「機動設計」特性。筆者將帶領大家運用「建構指令·**new**」及「釋放指令·**delete**」這兩個關鍵字來執行「建立動態陣列空間」及「釋放動態空間物件」行為。如果有設計需求時，也可在建構時同步加註初始值。

請各位參考下列「建立動態陣列空間」表述式....

☞ 單值指標物件 | 型式 \*指標 = new 型式 (初始值);

☞ 陣列指標物件 | 型式 \*指標 = new 型式 [空間值];

請各位參考下列「釋放動態空間物件」表述式....

☞ 單值指標物件 | delete 指標;

☞ 陣列指標物件 | delete [] 指標;

## | 實作研討題目 | 巨匠SEAFOOD · 壽司BAR · 消費結帳收費程式設計

提示 · 每次消費至少 1 盤，上限99盤。

參考 · 消費金額有30元、40元及80元三種價位。

\*範例編號：Study2\_03.dev

\*執行畫面...



----- 【 巨匠 SEAFOOD ； 壽司 BAR 】

消費盤子數 【 1 - 99 】 ： 6

◎ 盤子.01 - 金額：40

◎ 盤子.02 - 金額：40

◎ 盤子.03 - 金額：30

◎ 盤子.04 - 金額：30

◎ 盤子.05 - 金額：80

◎ 盤子.06 - 金額：30

+ 線上結帳，共計：250元

【 巨匠 SEAFOOD ； 壽司 BAR 】 -----

師父：See you at next time ...

請按任意鍵繼續 . . . ▬

# \*研討節次.

## 2-5. 動態記憶配置

【 sizeof . malloc . free 】

在傳統記憶體配置空間法規約束下，通常都要先設定明確「存取目標數」及執行「編譯建構化」這兩個規定。這種制式作法容易造成宣告空間上「無法重製」及「無用空間」佔用記憶體等後續問題，相對更人性化的功能開發將會造成設計困擾。為了徹底解決這個問題，於是發展出一套「動態記憶體」管理工具，就是本節介紹的「基底函數」進行「群組整合」應用設計。

其作業原理如下，當在執行過程中需要使用記憶體時，先使用 `sizeof()`、`malloc()` 函數來建立「基底空間」，過程中再利用「物件位移機制」進行資料存取，當空間不再使用時執行 `free()` 函數來釋放對應記憶體。運用三者功能函數中，除了 `sizeof()` 為開發工具內建支援外，其餘函數都要在引用 `stdlib.h` 標題後方可線上作業！

請各位參考下列「配置動態記憶體」表述式....

型式 計算目標空間 = `sizeof( 預留對應空間 ) * 參考目標數 ;`

型式 空間物件指標 = `(型式*) malloc( 計算目標空間 );`

請各位參考下列「釋放動態記憶體」表述式....

`free( 空間物件指標 );`



| 實作研討題目 | 模擬台灣彩卷經營樂透玩法中三星彩及四星彩的開獎畫面設計  
提示・可線上設定獎項目標，數字 3 代表三星彩，4 代表四星彩。  
參考・有關亂數機制及陣列觀念，自行查閱「C語言」課程章節。

\*範例編號：Study2\_04.dev



\*執行畫面...





巨匠線上真人

C++ 程式設計

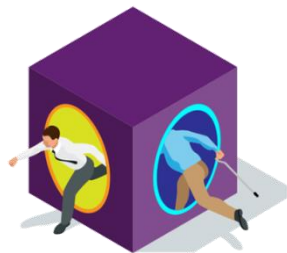
# 第三堂：技術主題 2 · 函數開發

[www.pcschoolonline.com.tw](http://www.pcschoolonline.com.tw)

巨匠電腦

# 🔗 本次課堂・教學重點 🔗

- 3-1. 進行函數的宣告、定義、引用
- 3-2. 原型與指定型式應用句型比較【 void vs type 】
- 3-3. 區域、全域、靜態宣告時機與動機
- 3-4. 參數傳送方法【 傳值 vs 傳址 】
- 3-5. 技術週邊研討【 內插．遞迴．多載．樣版．樣版多載 】



# \*研討節次.

## 3-1. 進行函數的宣告、定義、引用

所謂「函數 • Function」即是將一群具有相關動向的程式碼，採用「集中設計、技術應用」於功能開發的進階語法。其作業原理是以配置「參考型式」訂作函數類型及識別名稱，再利用「關係參數」進行資料互動存取及更新。其過程中先將「功能執行碼」編寫於自訂申請的「函數別名」內部，再依作業需求進行線上「呼叫引用」。設計方式有二，第一種「單向・無回傳」執行狀態，通常採用「程序模組」類型進行開發，而第二種「雙向・具回傳」互動狀態，就要使用「功能函式」類型進行開發。依照不同「設計動向」，規劃出控制句型語法，參考如下...

④ 單向・無回傳.

```
void 函數別名( 型式 參數 , ..... ) {  
  
    功能執行碼  
  
}
```

④ 雙向・具回傳.

```
type 函數別名( 型式 參數 , ..... ) {  
    功能執行碼  
    retrun value ;  
  
}
```

最後在使用「新型字串・String」及「原型指標・\*」這兩種技術型式上的作業關係，發現在新版設計工具 Dev-C++ 5.11 中，進行編譯時會出現警告訊息，希望可把 char\* 轉換成 string 型別提示。當然這並不代表「指標技術」就此被取代而不再強調其重要性，而是應該將其看成是更具人性化而發明的新式「型別物件」作法，而「指標技術應用」依舊是「快捷更新資料」的第一選擇！

# \*研討節次.

## 3-2. 原型與指定型式應用句型比較 【 void vs type 】

通常在「自訂函數」應用規則中，使用前都要進行「註冊宣告」行為，也就是所謂的「全型宣告」作業機制。因為這些開發介面是由設計師自行創作，無法由系統內建支援及引用預訂技術標題來執行，只有經過「註冊型式」空間行為，讓電腦經由宣告動作所預留的位址中找到線上執行目標，這樣才能維持正常功能運作。所以執行註冊有其重要性，唯一例外的是將「執行介面」放在「呼叫空間」之前，這樣就可以省略宣告動作，這也是現代書籍及教學者比較喜歡的示範的設計方式，如要進行「註冊宣告」行為，表述方式如下...

☞ 註冊・宣告式.    型式 函數別名( 型式 參數 , ..... );

最後依照不同「呼叫動作」，規劃出控制句型語法，參考如下...

☞ 單向・無回傳.    函數別名( 指示值 / 參照址 , ..... );

☞ 雙向・具回傳.    type 目標別名 = 函數別名( 指示值 / 參照址 , ..... );

# \*研討節次.

## 3-3. 區域、全域、靜態 宣告時機與動機



當我們在「自訂函數」與「呼叫動作」設計中進行資料傳送時，會先分析其過程作業期間關係變化後，再決定使用何種宣告(declare)方式來申請記憶體空間最為適合，也是設計業界中所謂的「生命週期」應用原理。

首先介紹「區域宣告・Local」的用法及特性，這是最基本也最常見的設計方式，其目標只能在「單一介面」裡進行功能應用，隸屬電腦自動化(auto)處理機制，進入「即時取得」存取資料位址，離開將「自動釋放」應用空間。

再談「全域宣告・Global」為人工整體化申請機制，可支援「整個檔案」的所有設計介面，通常放置於引用技術區塊中，會在所有「執行介面」應用前產生其作業空間及初始值，直到程式終止才會釋放記憶體資源。

最後要討論「靜態宣告・Static」亦為人工區域化申請機制，其功能在於「保留區域」重點資料，使該目標值不受自動譯放行為影響



# \*研討節次.

## 3-4. 參數傳送方法 【 傳值 vs傳址 】

單純呼叫「自訂函數」而不傳送任何資料時，就會省略參數(argument)架設。反之要將處理資料送至到「指定函數」應用時，就要同步完成「傳送資料值」與「控制參數列」互動式設計。

如果設計師在呼叫端提供資料實值，稱為「來源參數」關係，而在設計端則負責對應接收工作，稱為「目的參數」關係。進行參數互動式傳送有三種線上作法，從「傳值呼叫・By Value」、「傳參考值呼叫・By Reference」及「傳址呼叫・By Address」等選擇方法，作業關係參考如下...

④ 傳值呼叫.    ( 值 , ..... ) → ( 值型式 目標 , ..... )

此種作法，將「保留」來源值，利用「資料傳送」將值單向送出。

⑤ 傳參考值呼叫.    ( 值 , ..... ) → ( 值型式& 目標 , ..... )

此種作法，將「變更」來源值，利用「型式取址」將值連結送出。

⑥ 傳址呼叫.    ( &值 , ..... ) → ( 值型式 \*目標 , ..... )

此種作法，將「變更」來源值，利用「資料取址」將值連結送出。

到目前為止我們也掌握了許多函數資訊，再來做個「實作範例」體驗，把上述的技術語法整合在設計上吧！請各位先行完成本專案，再將程式碼對照本章前 4 節所說的應用關係變化吧！

| 實作研討題目 | 利用三種「資料參數」傳送作法，進行  $a, b$  兩值線上運算更新報告。

提示・設定  $a, b$  起始值為 10, 15，先用「傳值」進行「兩值相加」，再用「傳參考值」進行「兩值相減」，最後用「傳址」進行「兩值相乘」。

\*範例編號：Study3\_01.dev



\*執行畫面...

-----【傳值作法】・線上回傳 -----

取得參考原始值 --  $a = 10, b = 15$

$a + b = 25$

---【傳參考值作法】・線上更新 ----

參照型址異動後 --  $a = 20, b = 5$

$a - b = 15$

-----【傳址作法】・線上更新 -----

指標物件異動後 --  $a = 3, b = -25$

$a * b = -75$

請按任意鍵繼續 . . .

# \*研討節次.

## 3-5. 技術週邊研討

【 內插 . 遞迴 . 多載  
      . 樣版 . 樣版多載 】

當然身為期待成為「**Super Designer**」族群的各位，除了已擁有基本函數開發能力外，還要朝向更高境界邁進，挑戰不可能的任務！啊哈，筆者將帶領大家全面突破業界所認證的各項函數技能，就讓我們繼續看下去...

首先挑戰的技能項目是「內插函數・**Inline**」。這個作法的技術特性在於可直接執行「嵌入程式碼」，直接跳過「呼叫引用」動作，有效提昇執行效率，如同「快取空間」技術作法！作業原理是利用記憶體「指定空間」維持程式碼處在「長駐備用」狀態，雖可加速執行能，但有會同時造成記憶體負擔情事，所以只適用在「碼值簡短、具演算性」的應用時機，而非用在大量程式碼表述處理的場合，避免記憶空間過度消耗進而產生後續相關執行問題，這點還請設計者多加注意用法規範。

```
Ⓞ 內插函數.      inline 型式 函數別名( 型式 參數 , ..... ) {  
                    程式碼描述  
                    [ return value ; ]  
                    }
```

就讓我們架設一個參考範例，體驗設計手法。

| 實作研討題目 | 利用「內插函數」作法，設定「半徑、圓周率」後，求取2D圓周長。

提示 ·  $2D\text{圓周長} = 2R \times \pi$ 。 【 R · 半徑 |  $\pi$  · 圓周率 】

統一採用「固定小數 2 位」格式，將計算結果指定印出。

\*範例編號：Study3\_02.dev



\*執行畫面...

設定半徑及圓周率...

12.34 3.14159

對應圓周長 -- 77.53

請按任意鍵繼續 . . .



接著挑戰的技能項目是「遞迴函數・Recursion」。這個作法的技術特性在於使用「重複呼叫」函數本身程式碼，在執行過程中產生動態堆疊運算(stacking-operation)行為模式，再回歸起點整合求取解答的一種學術型作法，故業界也有「遞歸處理」說法。最主要的應用價值在於可減輕設計人員負擔複雜表述及分析流程能力外，也不需要架設「全面擬人化」行為指示動作。

雖然此法有其「智慧技術」能力，讓電腦力求完整模擬「人工思維」表現，但在實際執行過程中所要承受過高的硬體成本下，卻也讓專業設計者不敢輕言應用之，所以業界設計還是選用最符合成本的疊代(iteration)作法，也就是大家所熟悉的迴圈(loop)來取代其功。不過筆者絕對認同「進階演算」的非凡價值，就請把這項技術作法完成學習，準備未來進階設計之用！

#### ☞ 遞迴函數・原型.

```
void 函數別名( 型式 參數 , ..... ) {  
  
    程式碼描述  
  
    函數別名( value , ..... );  
  
}
```

#### ☞ 遞迴函數・具型.

```
型式 函數別名( 型式 參數 , ..... ) {  
  
    程式碼描述  
  
    return 終止值 ;  
  
    return 函數別名( value , ..... );  
  
}
```



就讓我們架設二個參考範例，體驗不同感受的設計手法。

| 實作研討題目 | 利用「遞迴」作法，求解 $12!$ 。

提示 ·  $12! = 1 \times 2 \times 3 \times 4 \times \dots \times 10 \times 11 \times 12$ 。

\*範例編號：Study3\_03.dev

\*執行畫面...

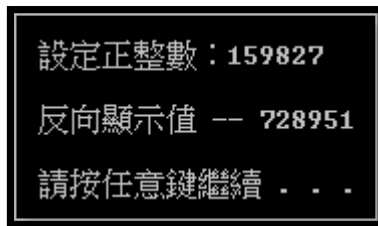


| 實作研討題目 | 利用「遞迴」作法，將輸入的數字「反轉顯示」處理。

提示 · 為避免視覺效果差異感觀，最後數字請勿輸入0。

\*範例編號：Study3\_04.dev

\*執行畫面...



再來挑戰的技能項目是「多載函數・Overloading」。這個作法的技術特性在於在使用相同函數名稱下，依據功能上的型式與參數上的變化差異，進行同類式功能設計，電腦也會依據其呼叫值及參考型式找到最佳執行對象，當然業界針對繼承類別作法也有「同名異式・Polymorphic」或「函式覆載・Overriding」新世代之說。

雖然此法通常都會採用「對應型式」及「資料參數」為搜尋目標引用依據，一旦發現查無原型目標指示時，就會自動啟用「參考轉型」方式後再尋找線上相符者，如果依舊無法辨識時就會造成編譯錯誤！通常會在參數列宣告加入「**const**・參考常數」關鍵字強調「唯一取用」行為，不讓其指定值目標內容被設計者任意線上更新或改造，確保執行程式碼的作業穩定性。

```
⚙ 覆載函數. 多元型式 共同函數別名( 多元型式 參數 , ..... ) {  
    程式碼描述  
    [ return value ; ]  
}
```

就讓我們架設一個參考範例，體驗設計手法。

| 實作研討題目 | 利用「函數多載」作法，求取正方體及長方體的體積大小。

提示・正方體・體積 = 邊長<sup>3</sup>。| 長方體・體積 = 長×寬×高。

正方體・邊長設(7)。| 長方體・長設(5)×寬設(3)×高設(9)。

\*範例編號：Study3\_05.dev



\*執行畫面...

正方體・體積 -- 343

長方體・體積 -- 135

請按任意鍵繼續 . . .

再來挑戰的技能項目是「樣版函數・Template」。本作法的技術特性在於會將「參數型式」來源當作線上「動態型式」建構根據，進而產生「參照型式」函數介面。

其使用時機在於具備「相同目標參數」但「不同作業型式」的關聯式整合設計！雖然此作法也可利用前節「多載函數」方式處理，最後會因為其法編寫「重複性函數」過多，進而造成程式碼維護上無法同步編修問題產生，此時就可以適時運用「樣版函數」作法來正式解決！

🔗 樣版函數.

```
template <class 動態型式>
```

```
    動態型式 函數別名(動態型式 參數 , ..... ) {
```

```
        程式碼描述
```

```
        [ return value ; ]
```

```
    }
```

就讓我們架設一個參考範例，體驗設計手法。

| 實作研討題目 | 利用「樣版」作法，取出不同來源型式值中的較大值。

提示・採用「同步輸入」作法，設定比較來源的「兩字元」及「兩整數」資料。

\*範例編號：Study3\_06.dev



\*執行畫面...

```
Set 兩字元 及 兩整數 ....  
K b 381 492
```

```
字元中較大者為 -- b  
整數裡較大值為 -- 492
```

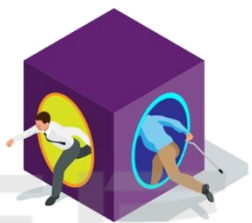
```
請按任意鍵繼續 . . .
```

最後挑戰的終極技能項目就是「樣版多載函數 · Template + Overloading」。本作法的技術特性在於徹底解決了傳統樣版開發下，無法進行「多元目標化」的麻煩問題。其實作業原理很單純原始，就是先把「資料型式」與「參數關係」分別開發後，再進行「整合應用」設計，其中型式問題就由「樣版」負責，參數指定就用「多載」處理，兩者技術各有所長、合作無間！相關應用作法，請往上參考本章相關節次線上說明。架設一個參考範例，體驗設計手法。

| 實作研討題目 | 利用「樣版多載」作法，搭配指標行為將兩值、兩陣列整數進行線上對應交換。

提示 · 兩值分別為 10 與 15，兩陣列分別為 { 9,7,5,3,1 } 與 { 0,2,4,6,8 }。

\*範例編號：Study3\_07.dev



\*執行畫面...

```
Change before -- Value1 = 10 , Value2 = 15  
  
Swap After -- Value1 = 15 , Value2 = 10
```

```
-----  
Change before -- Array-1 = 9 , Array-2 = 0  
  
Swap After -- Array-1 = 0 , Array-2 = 9
```

```
請按任意鍵繼續 . . .
```

C++ 程式設計

## 第四堂：技術主題 3 · 前端處理

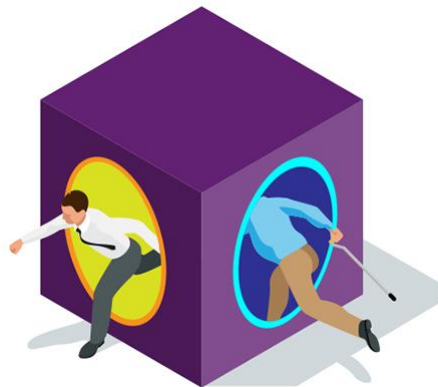
# 🔗 本次課堂 · 教學重點 🔗

4-1. 引用技術 【 `#include` 】

4-2. 巨集應用 【 `#define` . `#undef` 】

4-3. 條件編譯 【 `#ifdef` . `#ifndef` . `#if` . `#elif` . `#else` . `#endif` 】

4-5. 範例實作研討





# \*研討節次.

## 4-1. 引用技術

【 #include 】

在本節中我們將討論「前置處理器・Preprocessor」的技術應用。其作用於執行編譯前會先處理完成「前置引導・#」所指示的相關動作。如引用「技術標題」來源，其分成「實體標題檔案」及「動態空間機制」兩種世代作法，或者是設計師利用「定義巨集」來自訂線上參考的「資料常數」或「處理函式」，以及使用「條件編譯」來追蹤各種「執行環境」所要進行的「相關動作」指示。首先就從「引用技術」開始探討...

🔗 實體標題檔案. `#include <工具技術來源.h>` 或 `"設計師技術來源.h"`

🔗 動態空間機制. `#include <線上技術來源>`

通常採用「動態空間機制」為第二代新進式作法。除了相容大部份的C式工具語法外，也增加更完整應用的「命名空間・Namespace」技術。在 **Dev-C++ IDE 5.11** 指定版本中，要使用「功能物件」及「新式型別」時，都要先行啟用此空間技術完成「線上建構」行為，否則將會造成線上編譯錯誤，請各位多加注意！其應用語法如下...

🔗 新式技術. `using namespace 功能類別;`

通常在「C++語言」進行應用物件表述前，如「cin、cout、endl、string」等動作指示，就要使用「命名空間」來指定「std·制式類別」完成線上建構，方具物件化執行能力。

我們可以經由下區不同引用表述，觀察語言間作業應用方式！

```
🔗 C語言.      #include <stdio.h>
                #include <stdlib.h>
```

```
🔗 C++語言.    #include <iostream>
                using namespace std ;
```

當然在程式碼值表述過程中，要使用何種作業技術？何種工具來源？都要靠日積月累的「實戰經驗」去參考決定，尤其是在本單元課程下，面對「前有C、後有Java」等語言技術「相容應用」將更見重要，期待各位完整研習「正規開發」技巧，萬萬不可「死記語法」而忽略「變化行為」之要，這樣才能成為真正的設計專家。未來具能力創作且願意分享作業程式碼時，請先將「技術碼值」存入「指定標題」，再配合指定方法進行引用設計！

# \*研討節次.

## 4-2. 巨集應用

【 #define . #undef 】

本節再談如何發揮設計師相關自訂化技術應用。其可分「常數定義」及「函式定義」兩種作業方式。而常數定義可選用「巨集機制·`#define`」或「關鍵字彙·`const`」來進行宣告！

🔗 巨集機制. `#define` 常數別名 參考值

🔗 關鍵字彙. `const` 型式 常數別名 = 參考值；

\*語法示範·

使用巨集機制·定義圓周率 | `#define PI 3.14`；

使用關鍵字彙·定義營業率 | `const double Rate = 0.05`；

選用「巨集機制·`#define`」還可進行函式定義手法，進行更多元化應用設計。其中最具代表的就是「加註參數」來進行「速算函式」開發，其行為模式同「內插函數」精簡化，還可採用「巢狀架構」進行多層次處理，或者「動態解除」已定義的巨集重訂使用，可見此法機能性強大！

🔗 巨集應用. `#define` 函式別名(參數,...) 多元運算式

🔗 巨集解除. `#undef` 巨集別名

\*語法示範·

使用巨集機制·求取圓面積 | `#define CArea(a,b) pow(a,2)*b`

使用巨集機制·求取較大值 | `#define Max(a,b) (a>=b?a:b)`

使用此法在「多重參數」引用上與「傳統函數」開發行為有所不同，強調相關目標引用時，記得搭配「指定目標·()」進行線上控制。再說「巨集機制」作法會先行完成「碼值編譯、轉換敘述」引用前製作業，因此執行速度快捷，唯一缺點就是「碼值重構」問題，要審慎評估其應用設計時機。

讓我們架設一個參考範例，掌握應用設計手法。

| 實作研討題目 | 利用「巨集定義」作法，依參考半徑求出2D圓面積及圓周長。

提示·  $2D\text{圓面積} = R^2 \times \text{PI}$ 。 【 R·半徑 | PI·圓周率 】

$2D\text{圓周長} = 2R \times \text{PI}$ 。

半徑限制「1至30公分」完成輸入，答案採用「固定小數2位」格式輸出。

\*範例編號：Study4\_01.dev



\*執行畫面...

```
設定半徑值：-23  
Set Range : 1 To 30  
  
設定半徑值：30.6  
Set Range : 1 To 30  
  
設定半徑值：30  
  
圓面積：2827.43  
圓周長： 188.50  
  
請按任意鍵繼續 . . .
```

# \*研討節次.

## 4-3. 條件編譯

【 #ifdef . #ifndef .  
#if . #elif . #else . #endif 】

上節次中所說的「巨集應用」技術，採用「自動轉換」人工碼值為機械碼值行為，也是「設計應用」標準機制，但在「多重文件」或「特訂處理」進階設計需求下，就要考量加入「執行時機」指示作法，於是就有了本節次的相對「控制技術」加碼學習機會！

首先上場應用設計的是「完成定義・`#ifdef`」及「尚未定義・`#ifndef`」，這組「條件編譯」負責判別「指定巨集」是否已做定義，再決定後續指定動作表述。其參考語法如下...

```
#ifdef 巨集別名  
    指定動作指示  
#endif
```

```
#ifndef 巨集別名  
    指定動作指示  
#endif
```





就讓我們架設一個參考範例，掌握應用設計手法。

| 實作研討題目 | 利用「巨集定義」條件作法，追蹤「巨匠電腦」銷售課程產品在線上營運狀態。

提示・模擬「巨匠白金卡」行銷機制。早期購買實體卡可搭配上課卷享學習課程優惠，如在2002年前購買價為19800元，後續年份購買價為27800元，直至2008年起正式停止發行，轉售「線上白金卡」1年期17600元！

\*範例編號：Study4\_02.dev

\*執行畫面...

模擬臨櫃購買・巨匠白金卡....  
消費西元年份：1995  
線上收費 -- 19800元整  
謝謝光臨，期待再次為您服務。  
請按任意鍵繼續 . . .

模擬臨櫃購買・巨匠白金卡....  
消費西元年份：2005  
線上收費 -- 27800元整  
謝謝光臨，期待再次為您服務。  
請按任意鍵繼續 . . .

模擬臨櫃購買・巨匠白金卡....  
消費西元年份：2019  
終身白金卡活動已結束！  
是否轉買線上白金卡一年期？ (y/n) y  
線上收費 -- 17600元整  
謝謝光臨，期待再次為您服務。  
請按任意鍵繼續 . . .

模擬臨櫃購買・巨匠白金卡....  
消費西元年份：2008  
終身白金卡活動已結束！  
是否轉買線上白金卡一年期？ (y/n) N  
謝謝光臨，期待再次為您服務。  
請按任意鍵繼續 . . .



接著上場應用的是「`#if . #elif . #else . #endif`」，這組條件編譯是負責判別「指定巨集」該進行何種流程控制，其應用手法類似「人工判別」句型描述，其語法差異性在於「前置處理」設計是採用高階「互動指令」形成表述區域，加上完成「碼值轉譯」前置動作。其應用語法參考如下...

```
#if 控制條件 1
    成立・動作指示 1
#elif 控制條件 2
    成立・動作指示 2
    .
    .
#else
    不成立・動作指示
#endif
```



```
#if MenuID==0
    #define Hour 18
    #define Code "ELC10"
    #define Unit "C・程式入門"
    #define Level "初級設計人員"
#elif MenuID==1
    #define Hour 18
    #define Code "ELC11"
    #define Unit "C++・物件導向"
    #define Level "中級設計人員"
#elif MenuID==2
    .
    .
#else
    .
    .
#endif
```

就讓我們架設一個參考範例，掌握應用設計手法。

| 實作研討題目 | 利用「巨集定義」條件作法，模擬在「巨匠電腦」官方網頁，學員線上查閱程式相關單元課程，點選線上「選單項目」來瀏覽「頁面資訊」內容。

提示・取得「清單編號」值後，進行「頁面回應」設計，回應內容如下表格。  
請將相關課程資訊，預先建立於「course.h」，再進行線上引用設計。



\*範例編號：Study4\_03.dev

清單編號	0	1	2	3
課程代號	ELC10	ELC11	ELJ14B	EVC7B
程式單元	C・程式入門	C++・物件導向	Java・先修設計	C#.net・視覺開發
報名對象	初級設計	中級設計	中級認證	高階應用

\*執行畫面...

【巨匠電腦・程式頁面】線上介紹  
・課程代號：ELC11 -- [18H]  
・程式單元：C++・物件導向  
・報名對象：中級設計人員  
請按任意鍵繼續 . . .

\*取得編號 0・回應畫面

【巨匠電腦・程式頁面】線上介紹  
・課程代號：ELC10 -- [18H]  
・程式單元：C・程式入門  
・報名對象：初級設計人員  
請按任意鍵繼續 . . .

\*取得編號 1・回應畫面

【巨匠電腦・程式頁面】線上資料介紹  
・課程代號：ELJ14B -- [15H]  
・程式單元：Java・先修設計  
・報名對象：中級認證人員  
請按任意鍵繼續 . . .

\*取得編號 2・回應畫面

【巨匠電腦・程式頁面】線上資料介紹  
・課程代號：EVC7B -- [18H]  
・程式單元：C#.net・視覺開發  
・報名對象：高階應用族群  
請按任意鍵繼續 . . .

\*取得編號 3・回應畫面



巨匠線上真人

C++ 程式設計

第五堂：

技術主題 4 · 衍生型式和資料結構

[www.pcschoolonline.com.tw](http://www.pcschoolonline.com.tw)

巨匠電腦

# 🔗 本次課堂 · 教學重點 🔗

5-1. 自訂型式別名介紹【 typedef 】

5-2. 資料列舉句型介紹【 enum 】

5-3. 資料結構句型介紹【 struct 】

5-4. 資料聯合句型介紹【 union 】

5-5. 記憶體配置空間差異【 struct vs union 】

5-6. 應用類別句型介紹【 class 】



# \*研討節次.

## 5-1. 自訂型式別名介紹 【 typedef 】

所謂「型式定義・**Typedef**」即是將「原始型式」依設計需求而「重新定義」於新識別名稱的作業手法，所以業界也有「自訂型別」之說。這類設計手法在「人工語言」中特別喜愛應用，像是在 **C++** 及 **C#** 兩語言所提供「**string / byte**·中階字串物件 / 位元整數」又或者是在 **Java** 及 **Basic** 兩語言所提供「**String / Byte**·高階字串物件 / 位元整數」等新版型式就是最佳應用示範。

在新式版本「**Dev-C++ 5.11**」中也加註「命名空間」來建構「**string**·字串物件」進行線上取代「**char \***·字元指標」警示訊息，期待早日達成「多型整合」的物件化語言終極目標！

話說本作法與上章所談的「巨集定義·**#define**」有用法上的相似度，但實質處理行為卻是南轅北轍。「巨集作法」只是將程式碼進行識別字「指定取代」動作，使用「前置處理器」完成。然而「型式定義」卻是識別指定型態的「自訂命名」手法，屬於「編譯器執行」完成，尤其面對「特訂型式」相關處理事項，巨集定義更是無法申任，在設計作法選項上要特別注意！

相關行為表述式，參考應用說明如下...

☞ 宣告·表述式. **typedef** 原始型式 自訂型式別名；

線上應用示範. **typedef unsigned char byte**；

就讓我們架設一個參考範例，掌握應用設計手法。

｜實作研討題目｜模擬在跨年夜當下線上傳送訊息，某人想對好友麻吉們說的那句問候語是...

提示・用「巨集定義」那句問候語，引用「內建 string」來設定發送者及「自訂 byte」來設定中元年份後，以西元年份格式整合送出，參考下圖畫面進程式碼值設計。

\*範例編號：Study5\_01.dev

\*執行畫面...



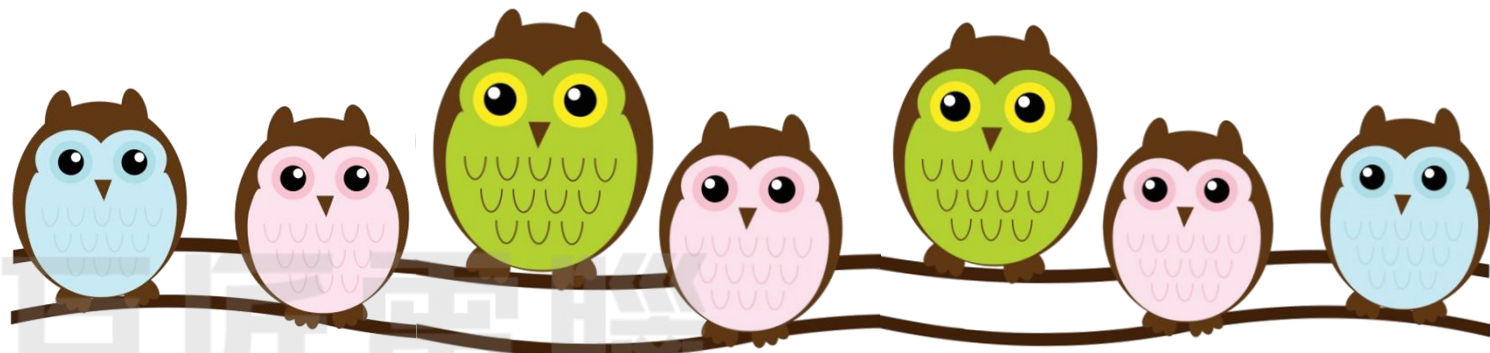


# \*研討節次.

## 5-2. 資料列舉句型介紹 【 enum 】

所謂「資料列舉・Enumeration」作法原理，就是一種採用「符號常數」來進行宣告的「自訂型式」另類手法。首先訂作「列舉標籤」來創建「型式別名」後，再架設相關「列舉元素」來設定參考值，待「指定型式」線上宣告完成，再建立「列舉變數」進行「線上取值」相關應用設計！

通常相關「列舉變數」在電腦內部作業上有個「應用約束」法則，就是只能使用「無號整數・unsigned int」方式進行配置，所有「列舉元素」都是採用「預訂起始值・0」機制，配合宣告順序建立「遞增位移・+1」關係產生對照值，想要改變這個預訂法則，可加上「指定來源值・=N」作法來更新參考位置值。最後使用本法設計要熟記「三大特訂」規則，第一是不可對列舉變數進行「非宣告傳值」動作，第二是不能對列舉變數、元素進行「線上運算」行為，第三是不能直接把列舉元素當成「輸出目標」控制，因為在編譯過程中電腦就會把「列舉元素」轉換成「對應整數」來進行參考應用，所以該「元素名稱」是對應資料值而非人工變數作業關係，請勿錯用指示！



最後依照不同「指示動作」，規劃出相關控制句型語法，參考如下...

⌚ 宣告・表述式.      `enum 列舉標籤 {`  
                            `列舉元素0 [= value ],`  
                            `列舉元素1 [= value ],`  
                            `.`  
                            `.`  
                            `列舉元素n [= value ],`  
                            `};`

⌚ 建構・表述式.      `[ enum ] 列舉標籤 列舉變數 1 , 列舉變數 2 , .... ;`

線上應用示範.      自訂「季節」為列舉型式，把「四季」當成列舉元素，  
                            指定「春季」為列舉變數參考來源值。

```
enum Season { Spring = 1 , Summer , Autumn , Winter } ;  
Season NowAt = Spring ;
```

就讓我們架設一個參考範例，掌握應用設計手法。

｜實作研討題目｜利用「列舉」作法，設計出依次顯示撲克牌 4 大花色...

提示・依照「ASCII編碼值 3 ~ 6」分別依序「轉換字元」印出。

\*範例編號：Study5\_02.dev

\*執行畫面...

撲克牌・花色 -- ♥ ♦ ♣ ♠  
請按任意鍵繼續 . . .



# \*研討節次.

## 5-3. 資料結構句型介紹 【 struct 】

所謂「資料結構・Structure」是依據「資料欄位・Field」架構來規劃出相關應用成員後，將工作項目集中「共同空間」的漸層次型別配置管理技術，專門用在「資料記錄・Record」處理。其作業原理是參考指示「成員宣告」所佔型式空間後，再進行「封裝整合」配置對應記憶空間。所以要配合線上建構物件，取得結構所在空間位址後，再指定「欄位成員」進行對應資料存取！依照不同「指示動作」，規劃出相關控制句型語法，參考如下...

☞ 同步・表述式.    **struct** 結構標籤 {  
                          資料成員宣告；  
                          } 結構物件1，結構物件2， ....；

☞ 建構・表述式. [ **struct** ] 結構標籤 結構物件1，結構物件2， ....；

線上應用示範. 自訂「朋友」結構型式，把「姓名、性別」當成資料成員，  
指定「同學、好友」為結構物件，採用「同步」設計。

**struct** Friend { **string** Name，Sex； } Classmate，Pal；

就讓我們架設一個參考範例，掌握應用設計手法。

｜實作研討題目｜利用「結構」作法，設計速查「麻吉通訊錄」裡的相關人員資料...

提示・用結構定義親友資料，資料欄位有姓名、性別及出生年份。

\*範例編號：Study5\_03.dev



\*執行畫面...

-----【材哥麻吉通訊錄】-----

- 男同學：張中侖 ！ 現齡：39
- 女同事：彭小玲 ！ 現齡：23

請按任意鍵繼續 . . .

如果想對「資料成員」進行參考值設定，最好不要在宣告當下使用「資料指派·=」機制，這個動作表述將會出現「編譯警告」訊息！因為「結構宣告」只是配置規劃應用空間，尚未達到實體作業狀態，冒然進行應用可能發生指定失誤情事。筆者建議可先將「線上物件」建構後，再進行指派是最安全的處理作法！當然也可利用「變形語法」來完成指派任務，參考如下...

④ 同步·表述式.     **struct** 結構標籤 {  
                          資料成員宣告；  
                          } 結構物件1 = { values } , 結構物件2 = { values } , .... ；

⑤ 建構·表述式.     [ **struct** ] 結構標籤 結構物件 = { values } , .... ；

線上應用示範. 自訂「朋友」結構型式，同步建立「好友」物件資料。

```
struct Friend { string Name , Sex } Pal = { "彭小玲" , "女" } ；
```

最後提醒各位，結構句型通常會把「資料成員」當成主體應用目標，但在適當時機配合「函式成員」來增加完整互動性也是重點設計所在，有關函式成員相關設計介紹就保留到本章最後一節「類別應用」時再詳細解說了，拭目以待吧！



# \*研討節次.

## 5-4. 資料聯合句型介紹 【 union 】

所謂「資料聯合・Union」技術是經由結構衍生的「首代修訂」型別。兩者有著記憶體控制上關鍵應用作法差異。如「結構作法」會先將「相關欄位」格式大小整合計算，再進行「全面對應」配置，而「聯合作法」卻是依「最大欄位」格式大小參考，再進行「指定欄位」配置，故業界亦有「共用空間」型別之說。雖然此技術作法有效節省作業記憶空間，但也受限資料存取是利用「絕對位址」作業關係，所以不可使用「實體物件・string」來進行目標型式宣告，更要注意處理同一物件下多重目標「指定衝突」及預防處理資料同步下控制目標「關鍵覆寫」等進階技術問題，其隸屬高難度應用句型語法。依照不同「指示動作」，規劃出相關控制句型語法，參考如下...

☞ 宣告・表述式.      union 聯合標籤 {  
                                資料成員宣告;  
                                };

☞ 建構・表述式.      [ union ] 聯合標籤 聯合物件1, 聯合物件2, ....;

最後提醒各位，聯合語法雖然採用「資料結構」為主要設計重心，但同時參考「基本類別」有限集合作法。也就是說本句型表述不但具備「資料成員、函式成員」同步開發外，也加註功能成員「互動關係」指示，線上提供「共用·**Public**、私用·**Private**、保護·**Protected**」三種應用機制，更支援函數成員進行「建構子·**Constructor**」與「解構子·**Destructor**」等特訂設計，相關應用技術就保留到下節次「類別應用」中再詳細介紹了！

線上應用示範. 自訂「家族」聯合型式，分段建構「雙親」應用物件。

```
union  Family { char *Name ; int Age ; } ;
```

```
[ union ] Family Father , Mother ;
```

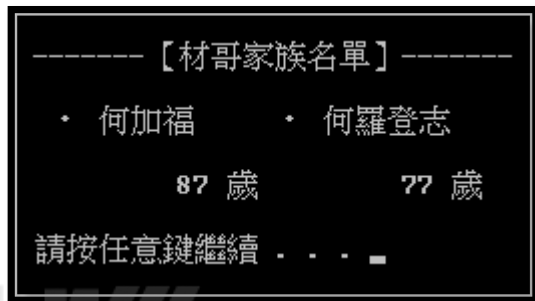


就讓我們架設一個參考範例，掌握應用設計手法。

｜實作研討題目｜利用「聯合」作法，設計速查戶口名簿裡的住家人員資料...  
提示・用聯合定義家族成員資料，登錄「姓名」及「出生年份」欄位，  
並將出生年份換算成「對應年齡」。

\*範例編號：Study5\_04.dev

\*執行畫面...



# \*研討節次.

## 5-5. 記憶體配置空間差異 【 struct vs union 】

為了讓各位更加了解「結構空間」與「聯合空間」兩者配置記憶體上的作法差異性，筆者利用下區「資料配置圖」進行來線上對應說明。

如果使用「結構空間」來配置單價(int)、數量(int)、折扣(float)三個資料欄位，因為採用「單欄獨立」配置作法，每欄配置空間大小為 4 Bytes，故佔用總空間為  $4 \times 3 = 12 \text{ Bytes}$ ；而「聯合空間」會先從三個欄位中找出「主欄折扣」配置作法，專欄配置空間大小為 4 Bytes，故佔用總空間為  $4 \times 1 = 4 \text{ Bytes}$ ，最後再依兩者不同控制語法進行線上指定應用！

雖然「聯合空間」作業原理看似單純便捷，卻有表述上「技術規則」與「空間約束」專業性考量，所以通用「結構空間」為主流作法，盼能降低「設計風險」及增加「應用彈性」。

`struct { int price , int number ; float discount } ;`

單價				數量				折扣				各自配置應用
												總空間 12 Bytes

`union { int price , int number ; float discount } ;`

單價				數量				折扣				共用配置應用
												總空間 4 Bytes



# \*研討節次.

## 5-6. 應用類別句型介紹 【 class 】

所謂「應用類別・Class」是經由結構衍生的「二代創建」型別。相關作業動向將可從「物件概念」來探索技術的各種變化動向。資料結構原是「功能成員」的基本管理模式，而應用類別就是「思維行為」的多重進階模式，像是「線上物件」的建構與解構，還有「控制方法」的建立、「參考屬性」的配置、「繼承多形」的設計，以及「觸發事件」的架設等作法，都是其可以深入研究的技術主題喔！

接著來趟現實世界「行為模式」探索藍圖之旅。首先腦袋放空、心情放鬆，想像一下...當想養寵物時會發生那些情事聯想？其實這個問題非常多元有趣，筆者常在課堂中與不同班級學員們共同討論，最後突然有個重大發現，那就是每個人的答案都不盡相同，甚至還有許多人當下連續改變想法版本，其中包含了筆者自己，哈哈！

不知道現在的你有想像並整理出心中首要版本了嗎？其實這個問題細分成兩個層面，第一層面是想與做的問題，是需要？還是想要？也就是說在行為執行時機上是已完成？還是未完成？相對在「創作類別」的技術層面上，同樣存在「實體類別」或「虛擬類別」應用關係。第二層面是依照「視覺思維」過程，將「控制目標」線上產生及進行「互動關係」指示行為。一般「傳統類別」的技術通用「實體動向」表述進行開發設計，依次產生四個控制行為，從指定類別(class)、建構物件(object)、控制方法(method)到觸發事件(event)，其中控制方法可用屬性(attribute)及函式(function)兩種作法處理，接著讓筆者分享自己的「真實愛犬」版本給大家參考吧！



話說材哥家裡住著一隻超級可愛地球萌生物(動物類別)，就是人類永遠的好朋友--狗寶貝(物件)！

先做個基本檔案介紹，狗寶貝名字(屬性)叫小米包(設定值)，很像台語發音下的燒麵包，線上類型(屬性)是隻短毛吉娃娃(屬性)，身體膚色(屬性)是討喜的紅白(設定值)，體型(屬性)豐碩寶貝5公斤重(設定值)，就像電影裡的神力女超人守護著我們，落地修行(屬性)近6年久(設定值)，還有....，它可是擁有多項賣萌超能力的狗寶貝喔，令家人感動買單情事不勝枚舉，特列出三大記事與大家分享...

- 一、打開家門時(事件)，就會看見它在為你真心守候(函式)，進入家門後(事件)，會一直在你身邊繞圈圈示好(函式)，那種幸福感覺真的讓人好窩心啊！
- 二、發現做了不對的事，像是嗯嗯或嘔嘔在不對地方時(事件)，就會頭低低心虛看著你，好像知道自己製造麻煩，準備接受懲罰(函式)。特別是在你抓狂對它訓示時(事件)，就用眯眯眼加笑笑嘴對著你，不停發送請原諒我吧，下不為例啦(函式)，那付萌樣表情實在無敵爆笑，讓人無法再繼續責怪它呢！
- 三、在戶外發現有人太靠近主人身邊或在保護區域聽見奇怪聲音或腳步聲 時(事件)，就會開始啟動警報模式，讓對方知道別想欺負...MY主人及或MY地盤(函式)，其實筆者也不知道這個住家所在地現在到底是Who的領地了，哈哈！



有關類別的技術不勝枚舉，在本程式單元中先做「觀念引導」及「基本建構」應用，最後依照不同「建構動作」表述，規劃出控制句型語法，參考如下...

⌚ 同步・表述式.

```
class 類別標籤 {  
    成員互動關係：  
        資料成員宣告；  
        函式成員宣告；  
} [ 類別物件1, 類別物件2, .... ];
```

⌚ 分段・表述式.

```
class 類別標籤 {  
    成員互動關係：  
        資料成員宣告；  
        函式成員宣告；  
};  
[ class ] 類別標籤 類別物件1, 類別物件2, ....;
```

成員互動關係分成「共用 · **Public**、私用 · **Private**、保護 · **Protected**」三種應用機制，當省略關係指示時，控制句型會以預設的「私用機制」進行應用。

```
線上應用示範.  class animal {  
                    [ private : ]  
                    string name , style , color , .... ;  
                    double age , weight ; ....  
                    public :  
                    void work1() { .. } ; void work2() { .. } ; ....  
                } dog ;
```

在本程式單元中業已完成類別觀念完整引導及存取成員機制應用示範。然而類別的技術具多元式開發能力，未來可運用更高階法則來進行設計，技術類型包含靜態成員(**static**)、建構函式(**constructor**)與解構函式(**destructor**)、抽象(**abstraction**)、封裝(**package**)、多形(**polymorphism**)、繼承(**inheritance**)、命名空間(**namespace**)等開發動向，還請各位繼續研習後續 **Java** 及 **C#** 等高階程式課程，藉以提昇自我線上專業級的設計能力養成。

就讓我們架設一個參考範例，掌握應用設計手法。

| 實作研討題目 | 利用「類別」作法，開發家中寵物資料的週邊化設計...

提示：用類別定義寵物相關應用資訊，含動物基本資料及專屬能力表述化。

\*範例編號：Study5\_05.dev



\*執行畫面...

---【材哥逍遙居·仙物風雲錄】---

- 寶貝代號：小米包
- 性別色系：女 -- 紅白
- 線上型態：吉娃娃 -- 短毛
- 實戰年資：5.8
- 重力指數：5.0

--+--- 線上超級能力 --+---

真心守候，不離不棄。

知錯能改，智慧如人。

保護主人，居家防護。

請按任意鍵繼續 . . .



巨匠線上真人

C++ 程式設計

# 第六堂：技術主題 5 · 檔案處理

[www.pcschoolonline.com.tw](http://www.pcschoolonline.com.tw)

巨匠電腦

## 🔄 本次課堂 · 教學重點 🔄

6-1. 檔案存取觀念建立、資料架構層次解析

6-2. 循序檔與隨機檔存取作法差異性

6-3. 建構類別引用【 `fstream . ifstream . ofstream` 】

6-4. 主體成員應用【 `open . close . eof . fail ...` 】

6-5. 輔助成員應用【 `get . getline . gcount . remove ...` 】



# \*研討節次.

## 6-1. 檔案存取觀念建立 、資料架構層次解析

從前段單一變數(variable)、群組陣列(array)到後段資料結構(structure)、聯合(union)、類別(class)，不管採用何種資料處理作法，都是利用記憶體(memory)來進行資料存取作業！但在「退出工具、結束程式、電腦關機」等動作下，相關建立資料會被全部清除、釋放記憶空間。如果這些資料是具有高重複性參考價值時(如會員註冊資料、產品進銷記錄等)，就要運用「技術作法」將其存入「指定檔案」，再適時取出應用與進行線上處理。

學習相關技術作法前，必須先對「資料架構」建立基本認知，筆者依其「組織層次」進行線上互動式解析說明。所謂「資料檔·Datafile」就是由「空間單位·Byte」、「資料欄位·Field」、「應用記錄·Record」三大層次元素所彙總而成的管理介面，更可搭配「資料庫·Database」機制來開發具「商務整合」專業級應用程式！

相關應用層次關係，請對照下張投影片中圖表參考研習...





資料庫

資料檔

資料檔

欄位	線上帳號	通行密碼	玩家代碼	用戶等級	遊戲點數	...
字元	tsaigo	14@You	何材仔	Vip-4	516888	...
記錄	會員2					
	會員3					
	.					

資料庫



# \*研討節次.

## 6-2. 循序檔與隨機檔 存取作法差異性

所謂「循序檔・Sequential File」作法是採用「逐筆存入」方式來進行「資料互動」處理。執行「查詢」會先將檔案指標移至「檔案開頭・BOF」，再從第一筆記錄開始往下搜尋。執行「新增」會先將檔案指標移至「檔案結尾・EOF」，再執行資料值寫入動作。適合「資料量大、位數不一、目標單純」的資料類型作業。本法優勢在於記錄空間精簡化，而缺點就是處理流程過於傳統原始，處理時間相對變長。檔案空間存取記錄關係，參考下圖...

### 循序檔・逐筆存入

記錄1		記錄2		記錄3		...
-----	--	-----	--	-----	--	-----

再談「隨機檔・Random File」作法是採用「編號管理」方式來進行「資料機動」處理。執行「功能」會將檔案指標移至「指定編號・NO」，完成定位再進行線上指定作業。適合「資料量小、位數固定、目標多元」的資料類型。本法優勢在於處理速度快捷化，而缺點就是欄位完整配置化，應用空間相對變大。檔案空間存取記錄關係，參考下圖...

### 隨機檔・編號管理



管理 編號	記錄
	記錄
	記錄
	記錄

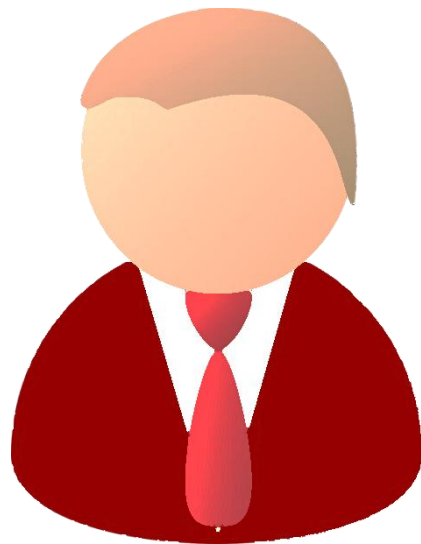
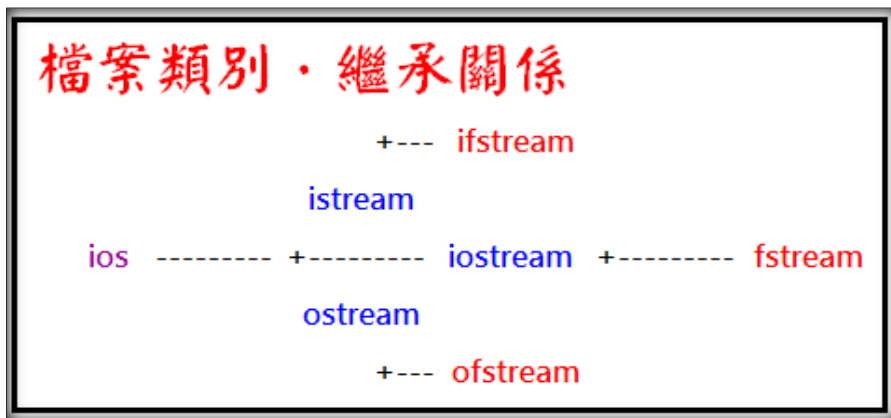
# \*研討節次.

## 6-3. 建構類別引用

【 fstream . ifstream  
                  . ofstream 】

檔案應用存取定義於「`fstream` . `ifstream` . `ofstream`」這三個衍生類別，隸屬 `fstream` 技術標題管理。其中類別 `ifstream` 具備「檔案輸入 / 讀取資料」控制能力，類別 `ofstream` 具備「檔案輸出 / 寫入資料」控制能力，而類別 `fstream` 兼具「兩者同步」控制能力。

檔案類別下的應用繼承關係，參考下圖...



建立「資料檔案」需要引用「何種類別」機制？如何將作業資料進行「檔案寫入」完成紀錄或者執行「檔案讀取」線上應用呢？

相關動作流程如下，首先要先行指定「檔案類別」機制來完成「檔案物件」建構，然後設定連結資訊 (指示路徑、資料檔名)，最後加註執行「作業模式」，應用作法如下...



🕒 開啟檔案・專供「資料讀取」.

```
ifstream 檔案物件("檔案資訊", ios::作業模式);
```

🕒 開啟檔案・專供「資料寫入」.

```
ofstream 檔案物件("檔案資訊", ios::作業模式);
```

🕒 開啟檔案・執行「同步存取」.

```
fstream 檔案物件("檔案資訊", ios::作業模式);
```

相關檔案應用及功能說明，筆者特別整理出下列五項重點提示！

- ① 上述表述句型也可分段應用，先完成「檔案物件」建構，再搭配「四大動向」功能函式線上分段作業。筆者將從「檔案開啟與關閉，`open()`、`close()`」到「資料讀取與寫入，`get()`、`getline()`、`read()`、`put()`、`write()`」，搭配線上實作範例，加強應用設計體驗。
- ② 當需要設定多個「作業模式」同時參考，可以利用「聯動字元，`|`」進行控制組合，增加線上功能應用彈性。唯獨不可造成「指示關係」衝突，避免控制失誤情事發生。
- ③ 電腦預設機制採用「純文字，`Text`」作業模式，設計人員可選用文書軟體，進行線上內容查閱與編輯，此模式為實體空間記錄化，屬於開放式應用並不適合存放機密值資料。反之，可另類指定「二進位，`binary`」作業模式，本模式可加速取用精準數據值及節省資料存放空間外，也會根據編碼機制進行改造原始資料值動作，確實完成資料保密化的線上保護作業效果。

- ④ 指定使用 **out** 或 **app** 模式，當發現「指定檔案」不存在時，電腦就會執行「自動建檔」動作。選擇 **out** 模式會將「原始資料」全面清除，功能雷同執行 **trunc** 模式，再將「新建資料」寫入檔案。選擇 **app** 模式會將「原始資料」全面保留，再將「新建資料」寫入檔案。選擇 **in** 模式會將「原始資料」線上讀取，再做「線上應用」功能設計。選擇 **ate** 模式則是強調在檔案結尾讀寫資料，因為作法無明確執行定位性，故筆者不建議線上任意使用！

模式	out	app	in	trunc	ate
功能	建製化	新增化	讀取化	清除值	結尾位

- ⑤ 正式應用「檔案物件」前，可搭配「線上檢查」作業來確定建構完成。筆者提供三個線上實用功能函式「檔案開啟·**is\_open()**」、「指定失誤·**fail()**」、「檔尾定位·**eof()**」給大家參考。全部採用「回傳布林值」應用作法，下區將示範無法正常開啟檔案的處理表述....

```
☞ 線上除錯.  if ( 檔案物件.fail() ) {  
                cerr<<"開檔失敗，原因"; exit(0);  
            }
```



除了運用上述人工類別的「函式成員・Function」來進行資料存取動作外，其實在「C++ 語言・物件導向」原始存取機制中是使用「串流 / 匯流・Stream」進行資料快取設計行為，如果善加運用此類表述，將可有效簡化存取指示行為。應用句型如下參考...

④ 資料寫入・輸出行為.

檔案物件 << 目標式;   【 例. fout << "開心喜悅" << endl; 】

④ 資料讀取・輸入行為.

檔案物件 >> 值目標;   【 例. fin >> mood; 】



# \*研討節次.

6-4. 主體成員應用

【 open . close . eof . fail ... 】

在「檔案物件」定義完成當下，也許沒有馬上開啟「指定檔案」應用，可在適當時機下再執行功能函式作業。較實用的主體成員有「開啟檔案·open」與「關閉檔案·close」，以及「檔尾定位·eof」、「指定失誤·fail」等功能函式，設計者可將其適時整合應用表述。

🔗 開啟檔案 | 關閉檔案·表述式. 檔案物件.open(); | 檔案物件.close();

就讓我們架設一個參考範例，掌握應用設計手法。

| 實作研討題目 | 將「書籤內容」三行文字，利用線上「建製模式」存入指定檔案中。

提示·線上「書籤內容」文字可自訂化或配合輸入，資料檔名採用「Content.dat」命名。

懂得分享，人生是彩色的！  
生命的能量活水，  
從成長情緒中湧出！

\*範例編號：Study6\_01.dev



\*執行畫面...

【書籤】內容，【建檔】寫入。  
請按任意鍵繼續 . . .

再讓我們架設一個輔助範例，比較設計手法。

| 實作研討題目 | 將書籤內容加註「作者資訊」，利用線上「新增模式」存入指定檔案中。

提示・內容文字也可自行設定，資料檔名採用「Content.dat」命名。

第四行進行「間距設計」處理，第五行印出「作者資訊」文字。

挑戰・傳送專案 Study6\_01 下「Content.dat」到專案 Study6\_02 後再執行存檔作業。

懂得分享，人生是彩色的！  
生命的能量活水，  
從成長情緒中湧出！

逍遙學長。材哥題

\*範例編號：Study6\_02.dev

\*執行畫面...

【作者】內容，【新增】寫入。  
請按任意鍵繼續 . . .



完成上述兩個實作範例，經由執行程式碼所建立的出來「資料檔案」，隸屬「文字文件」作業類型，如要追蹤「過程內容」存入檔案結果狀態，可以選用「文書編輯」軟體來進行查閱記錄資料，當然也可以直接呼叫「記事本」應用程式來線上應用喔！

在不同專案位置下，如何指定「共同檔案」來存取資料呢？第一種是採用「線上複製」，將資料檔案進行專案間「傳送更新」，屬「輕量級」作法；第二種是利用「碼值控制」，將資料存取於指定路徑下「來源檔案」，屬「專業級」作法。當語法表述上「省略指示」路徑時，電腦就會使用應用程式「相對位置」來進行資料檔案存取，也是業界所說的「相對路徑」。反之，則是「加註指示」路徑，也就是所謂的「絕對路徑」。最後在「路徑機制」上筆者建議適時搭配「外部參照·/」或「內部引用·\\」不同作法來進行對應設計。唯有具備「路徑資訊」完全控制能力者，才稱得上是位真正的「檔案應用」專家喔！

📍 相對路徑·表述式. Content.dat 或 .....

📍 絕對路徑·表述式. C:/Content.dat 或 C:\\Content.dat 或 .....

# \*研討節次.

## 6-5. 輔助成員應用

【 get . getline  
    . gcount . remove ... 】

當然除了「主體成員」函式外，還有各種「輔助成員」函式，筆者將其逐一列出，提供設計人員線上參考。應用動向有「字元指定·get」、「字元印出·put」、「字串指定·getline」、「讀取來源·read」、「指標送出·write」、「讀取完成·good」、「清除狀態·clear」、「下筆預覽·peek」、「檔案移除·remove」、「字串計數·gcount」、「預覽下筆·peek」等功能函式，都能讓各位設計者上場大顯身手一番！

就讓我們架設一個參考範例，掌握應用設計手法。

| 實作研討題目 | 利用「檔案移除」作法，將存在「相對路徑」下的「指定檔案」線上刪除！  
提示：字母不計大小寫，但線上輸入必須提供「檔案全名」資訊。

\*範例編號：Study6\_03.dev



\*執行畫面...

設定移除檔名 -- Content.DAT

線上回應：該檔案已被移除！

設定移除檔名 -- Content.DAT

線上回應：指定檔案不存在！

再讓我們架設最後一個應用範例，感受設計手法。

| 實作研討題目 | 利用線上「讀取模式」分析歌詞檔案，統計「內部文字」佔用空間數...

提示・半形字佔用 **1 Byte**，全形字佔 **2 Byte**，資料間距上的空白字元也是計數目標。

確認作業檔案是否能夠正常開啟，再進行線上指定作業，搭配「防錯機制」設計。

挑戰・將歌詞內容依「英文字、數字、符號、空白及全形字」進行「分類字元」計數。

\*範例編號：Study6\_04.dev



\*執行畫面...

```
Can't open this file : Song.DAT
```

```
請按任意鍵繼續 . . . █
```

```
So I cross my heart and I hope to die
```

```
That I'll only stay with you one more night
```

```
And I know I said it a million times
```

```
But I'll only stay with you one more night
```

```
歌詞字數空間：170 Byte(s) , 請按任意鍵繼續 . . . █
```



 巨匠線上真人

[www.pcschoolonline.com.tw](http://www.pcschoolonline.com.tw)

 巨匠電腦