



# C/C++基礎程式設計

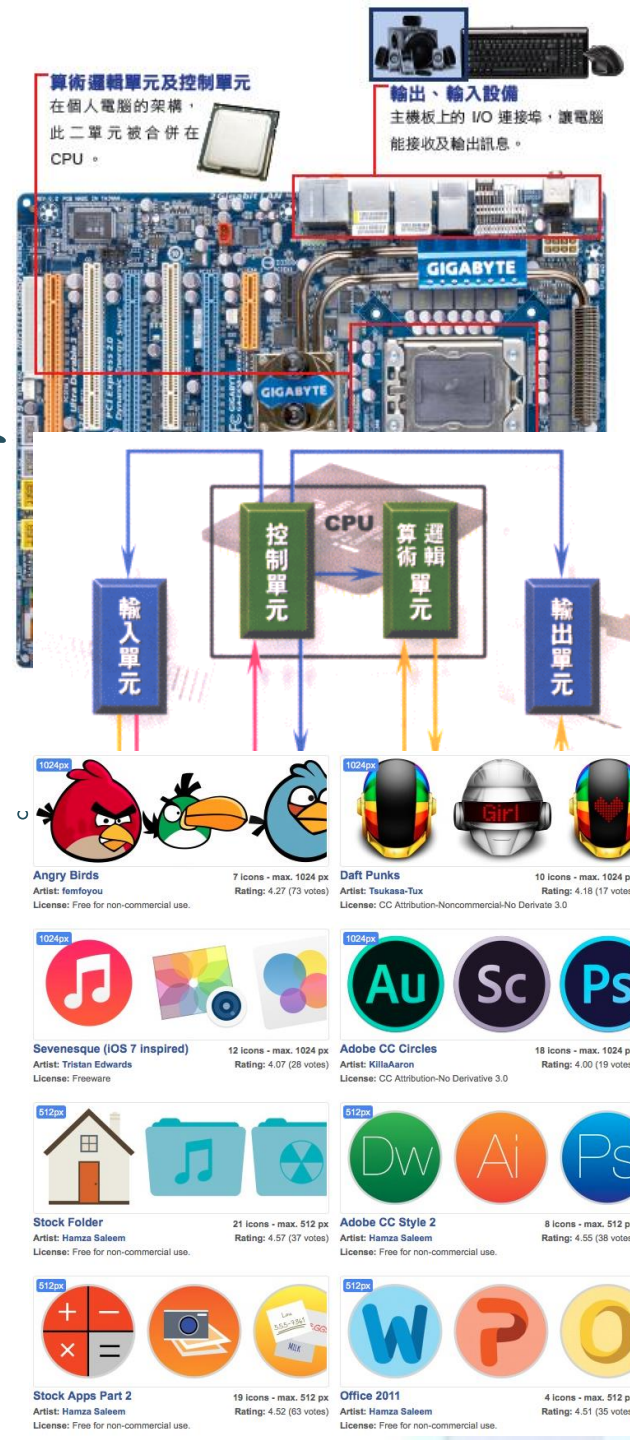
C語言概觀

講師：張傑帆

CSIE, NTU

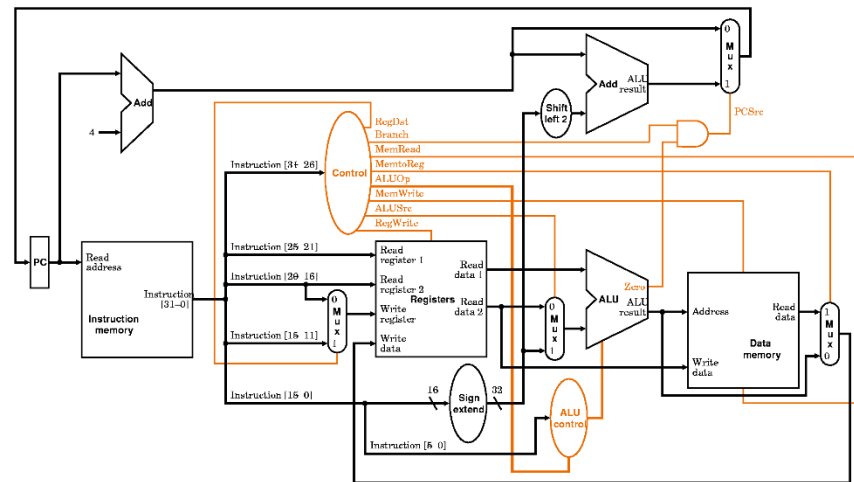
# 程式語言介紹

- 電腦是人類二十世紀最重要發明之一
- 電腦是由硬體與軟體構成。
- **硬體(Hardware)**
  - 負責執行解決問題所必須的基本運算和處理。
  - 由中央處理單元、記憶裝置、輸出入裝置構成。
  - 目前朝輕、薄、短、小及發展。
- **軟體(Software)**
  - 用來指揮硬體運作，為解決問題的指令集合。
  - 將這些指令的集合稱為程式(Program)。
  - 將軟體分成系統軟體和應用軟體。



# 程式語言的分類

- 人要如何與電腦溝通？
- 程式語言



- 第一代語言：機器語言 (Machine Language)
- 第二代語言：組合語言
- 第三代語言：高階語言
- 第四代語言：查詢語言 (Query Language)
- 第五代語言：物件導向與自然語言





# 程式語言的分類

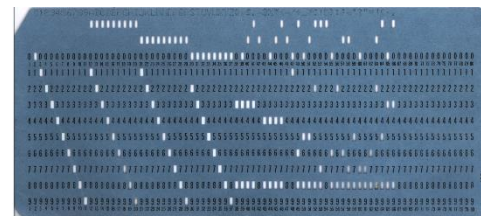


11010110  
00101100  
10101110

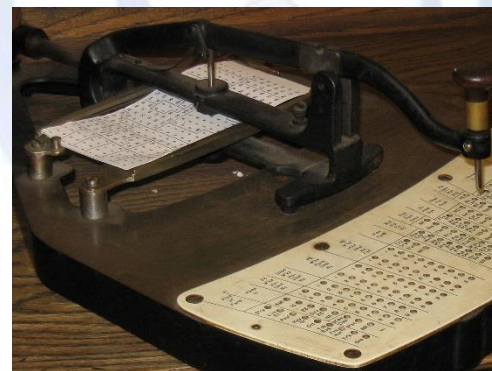


## 第一代：機器語言 (Machine Language)

- 以連續0、1來編寫程式，**執行速度最快**。
- 屬機器導向語言。CPU 的架構不同，此種語言與機器相依度高，**可攜性極低**。
- 0、1組合而成，費時費力，實用性差且**難維護**。



記憶位址	內容(2 進制)	內容(16 進制)
1000	1010 0011 0000 0001	A301
1002	0000 0001 1011 0010	01B2
1004	0001 0011 1101 0101	13D5



# 第二代語言：組合語言 (Assembly Language)



組合  
語言

組合程式

機器  
語言



- 亦稱低階語言，屬於一種符號式語言。
- 是使用助憶碼，由字母和數字組合而成。
- $sum = 10 + 20$ ，組合語言寫法：

```
mov ax, 10 ;
add ax, 20 ;
mov sum, ax ;
```

組合語言：

```
mov ax,WORD PTR [bp-4];final
mov dx,WORD PTR [bp-2]
mov WORD PTR [bp+8],ax;new
mov WORD PTR [bp+10],ax
$|194
```

組譯器

機器語言：

```
10001011 01000110 11111100
10001011 01010110 11111110
10001001 01000110 00001000
10001001 01010110 00001010
```

AL 1 1 1 1 0 1 1 0

(a)

SHR AL, 1 CF  
AL 0 1 1 1 1 0 1 1 → 0

(b)

SAR AL, 1 CF  
AL 1 1 1 1 1 0 1 1 → 0

(c)

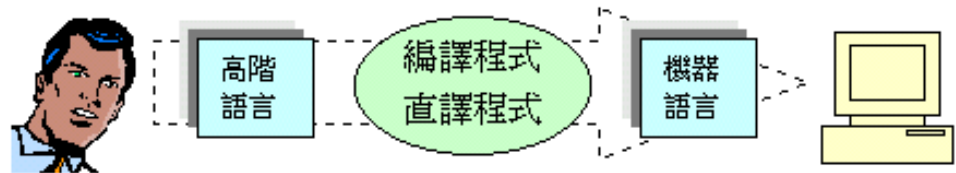
CF SHL/SAL AL, 1  
1 1 1 1 0 1 1 0 0 0 AL

- 屬於機器導向語言，和電腦硬體相依性高，不同CPU，語法不同，可攜性低。
- 適用於電腦專業人員編寫有關電腦系統或輸出入介面的驅動程式。
- 使用組譯器將撰寫的程式碼逐行翻譯成機器語言才能執行。

# 第三代語言：高階語言 (High-Level Language)

- 語法更**接近人類語言**與數學表示式，程式稍加修改，可在不同電腦系統上執行，**可攜性高**。

- 屬於程序導向語言



- 如：

**BASIC**(交談式操作環境)、**FORTRAN**(工程)、**COBOL**(商業應用)、**PASCAL**、**C**均屬之，由於都是屬於傳統高階語言，共同特點就是**按照指令的邏輯順序執行**。

```
#include <asm-insn.h>
```

```
int foo(int cond_p)
```

```
{  
    int i, sum = 0;  
    extern int inp1, inp2;
```

```
    if (cond_p)
```

```
    {  
        st_acX_direct32_dw (inp1, 1234);  
    }
```

```
    else
```

```
    {  
        st_acX_direct32_dw (inp1, 4321);  
    }
```

```
    for (i = 0 ; i < 1000; i++)
```

```
    {  
        sum = msu_acXl_acYl_acZ (inp2, inp1, sum);  
    }
```

```
    return sum;  
}
```

```
A.M0.cmp {dw,eq} a0, #0, prg01
```

```
|| LS0.ld {dw} [_inp2], a4
```

```
|| LS1.ld {dw} [_inp1], a5
```

```
SQ.bkrep {ds1} #124
```

```
|| A.S.clr a2
```

```
|| A.L.clr a3
```

```
LS0.ld {dw} [_inp1], a0
```

```
{
```

```
    A.M1.mpy a4l, a5l
```

```
|| A.M0.msu3 a4l, a5l, a2
```

```
|| A.L.mpy a4l, a5l
```

```
|| A.S.msu3 a4l, a5l, a3
```

```
    A.M1.mpy a4l, a5l
```

```
|| A.M0.msu3 a4l, a5l, a2
```

```
|| A.L.mpy a4l, a5l
```

```
|| A.S.msu3 a4l, a5l, a3
```

```
}
```

```
SQ.ret {ds2, t}
```

```
LS0.st {dw} a0, [#1234], ?prg1
```

```
|| LS1.st {dw} a0, [#4321], ?prg0
```

```
A.S.add a2, a3, a0
```

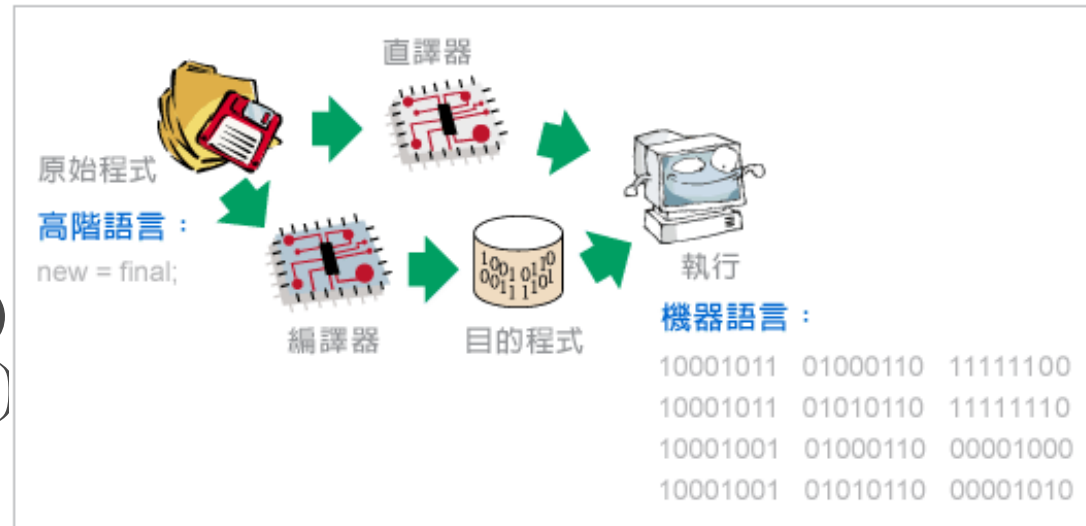
## 第四代語言：查詢語言(Query Language)

- 屬於非程序語言以問題為導向，只描述問題不必敘述解決問題的步驟。
- 先透過前置處理器轉換成第三代的程序語言才能編譯成可執行碼。
- 包括結構化查詢語言SQL(Structural Query Language)適用於資料庫查詢或 AutoCAD適用於工程繪圖。

# 第五代語言：物件導向與自然語言

- 物件導向語言，是一種比程序導向更進階的語言。
- C++ 是在 C 中加入物件導向語法的程式語言。
- 此種語言每個物件擁有自己的屬性和方法，具有下列特性：

- 再利用(Reused)
- 繼承(Inheritance)
- 封裝(Encapsulation)
- 多形(Polymorphism)



- 使得物件有如積木一樣都具有某些小功能，物件與物件間利用呼叫可互傳資訊或兜成一個大程式。



- 由於網際網路蓬勃發展、超媒體與網路資訊服務充斥全球資訊網、以及智慧型裝置(智慧型手機與平板電腦)硬體設備愈來愈進步。
- 因此許多大廠紛紛提供能開發**Web 應用程式**(網頁程式設計)與智慧型裝置應用程式的程式語言，像這類的程式語言有：Java、VB、C#、ActionScript 3.0...等物件導向程式語言。
- **自然語言**(Natural Language)屬於**人工智慧語言**，近似人類的語言是程式語言的終極目標。如：LISP(LIST Processing)、PROLOG(LOGic PROgramming)。
- 鋼鐵人中的 Jarvis

**自然語言：**

Tell me the telephone number of KZ.

I don't know what you mean about KZ.

Kevin Zheng.

2933-1789.

## 翻譯器的分類

- 編譯器(Compiler)
- 直譯器(Interpreter)
- 組譯器(Assembler)



# 編譯器 (Compiler)

- 是電腦廠商提供的系統軟體(程式)。
- 將高階語言所寫的程式碼轉換成能直接被機器接受之目的程式。
- 優點：  
是程式經編譯過存成目的檔，下次執行時程式若未修改過可馬上執行，較節省編譯和執行時間。
- 缺點：  
編譯和連結時間較長而且程式有修改過必須重新編譯程式執行時必須將整個執行檔一次載入，需要較大的記憶體、程式存檔時亦需要較大的輔助儲存體空間、執行階段發生錯誤時除錯較難處理。

# 直譯程式 (Interpreter)

- 亦是電腦廠商提供的系統程式之一。
- 將高階語言所編寫的程式碼，依其敘述的邏輯順序，將指令逐一轉為機器語言指令後執行。
- 優點：  
執行時所需記憶體空間和存檔時所需磁碟空間較小，且程式較易除錯適合初學者。
- 缺點：  
每次執行均須重新翻譯，執行所需的時間較長，程式若供多人使用時效率較差。



# C 語言的沿革

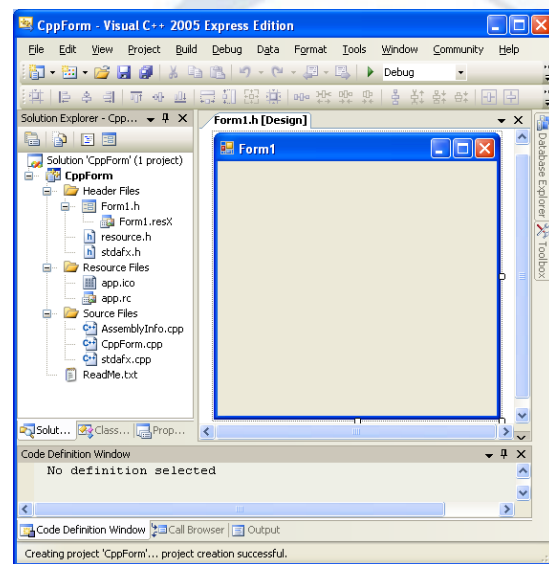
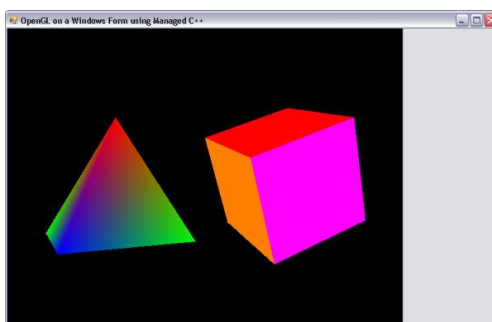
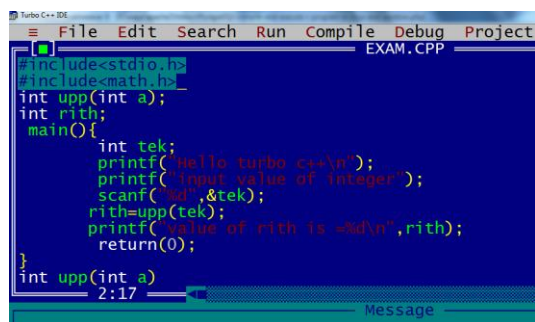
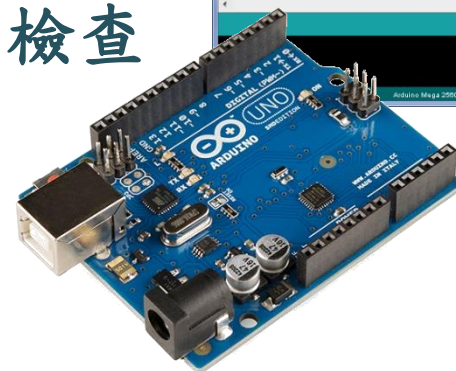
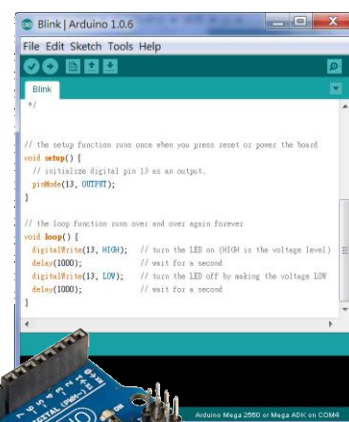
- C 語言的前身追溯到 1960 年以解決問題為導向的高階語言 - ALGOL 60，當時博得好評價，不適合撰寫系統軟體。
- 1963 年英國劍橋和倫敦大學以 ALGOL60 為基礎，共同推出與硬體有關 CPL (Combined Programming Language) 語言，由於當時考慮層面過於寬廣，造成不方便撰寫系統軟體。
- 1968 年 Martin Richards (世界公認 C 語言的鼻祖) 在英格蘭劍橋簡化 CPL 語言而發展出 BCPL (Basic Combined Programming Language)。

- 1970年  
Ken Thompson於美國Bell 實驗室再度精簡BCPL語言設計出接近硬體的 B 語言。
- 1972~1973 年間  
Dennis Ritchie於美國 Bell 實驗室，為新型 PDP-11 電腦重新改寫 Unix 作業系統，結合B 語言和 BCPL語言重要觀念加上資料型別以及一些其他概念發展成 C 語言。
- 1973年  
K. Thompson 和 D. M. Ritchie 兩人合作把 UNIX的 90% 以上用 C 改寫成即UNIX-5，直到 1975 年 UNIX-6公佈後，才引起注意。
- 1978年  
Ritchie 和 Brian Kernighan於出版「The C Programming Language」一書，奠定 C 語言完整架構，將此版本的 C 語言稱為 K&R C 語言。

- 隨後 C 語言百家爭名，產生出多種版本的C語言如：Lattic C、MS-C、Quick C 等，
- 為使 C 語言標準化，1983年夏，美國國家標準協會 (ANSI) 制定一套 ANSI 的 C 語言標準。
- 標準化過程達六年之久，最後於 1989/12 ANSI 標準終於完成，1991年初 ANSI C 第一版 終於出現。
- 1987年美國寶蘭(Borland)公司結合了 K&R C 和ANSI C 推出Turbo C，深受當時程式設計者的喜愛隨著資訊科技的進步，導致物件導向程式設計的流行。

# C 語言的特色

- 程式具有區塊結構及不嚴謹的資料型別檢查
- 為UNIX 作業系統所採用的程式語言
- 介於低階和高階語言中階程式語言
- 可呼叫處理硬體函式庫或自行設計需要函式庫來直接控制硬體，以提升硬體執行速度。另方面 C 語言可用來發展高階軟體介面
- C 語言具有高階架構和低階功能



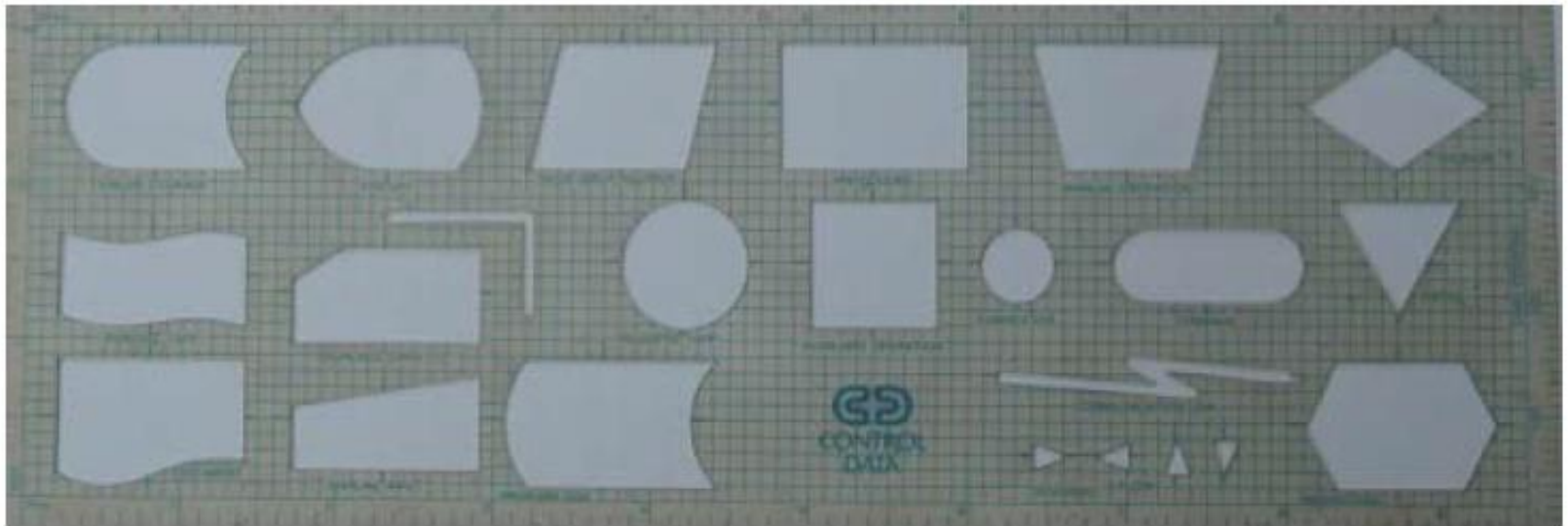



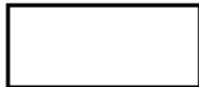




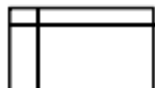
- 為一種可攜性的系統程式發展語言
- 具可攜性及高跨平台功能
- 結構化程式設計
- 提供指標及位址運算能力
- 允許使用動態資料結構

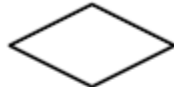


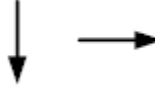


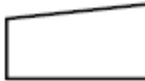


# 流程圖

- 所謂的「流程圖」即是使用各種不同的圖示符號來描述問題的解決步驟以及進行的順序。流程圖中所使用各種符號的繪製都已標準化。

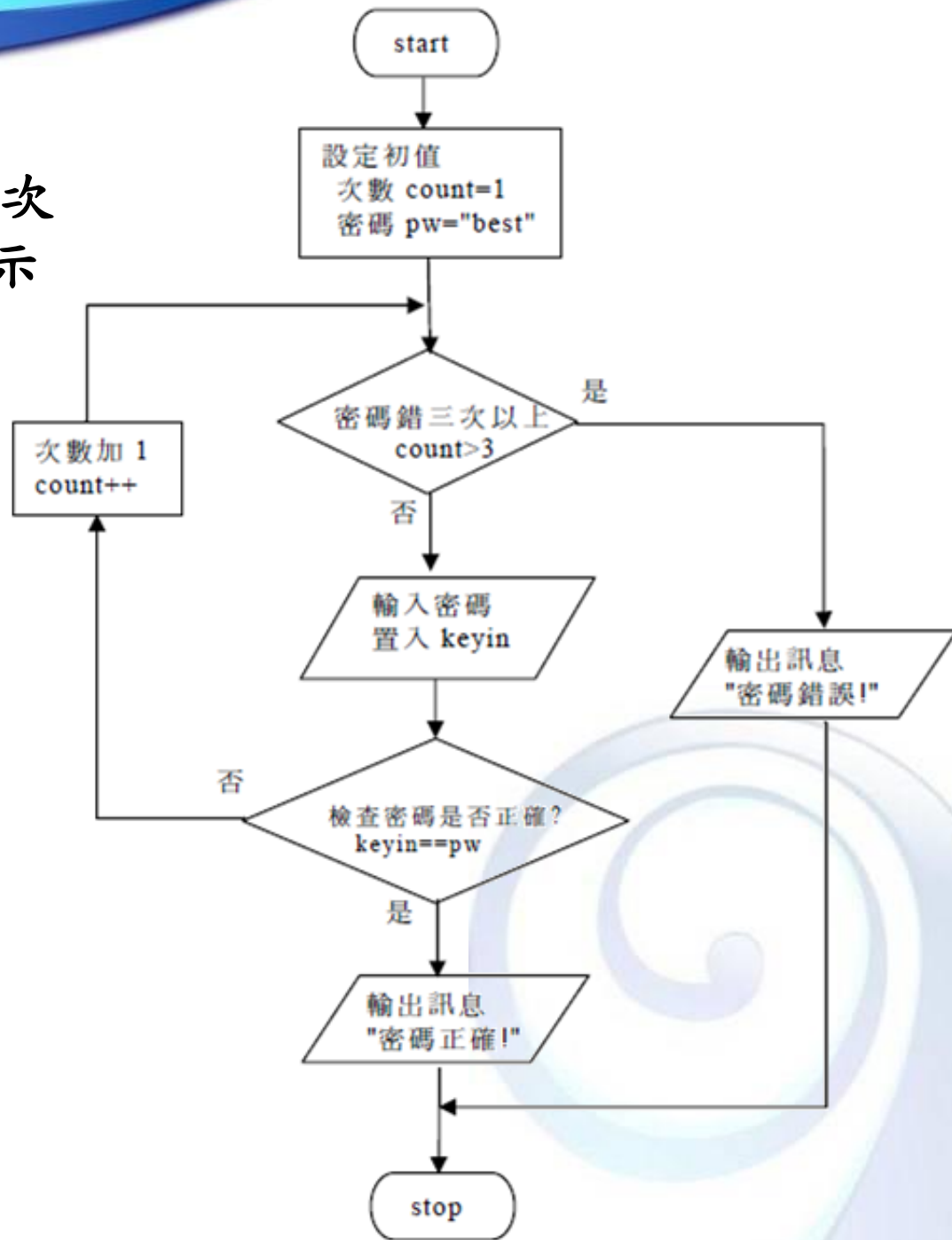


符號	功能
	開始/結束
	程序處理
	輸入或輸出
	連結符號
	儲存資料
	準備作業
	內部儲存裝置

符號	功能
	判斷比較
	隔頁連結
	預設處理作業
	工作流向符號
	文件
	多重文件
	人工輸入

由鍵盤輸入密碼，**限**輸入三次  
**若**在三次內答對密碼，**則**顯示  
“密碼正確！”  
**若**連續答錯三次，**則**顯示  
“密碼錯誤！”

其流程圖表示方式：







# C/C++基礎程式設計

整合開發環境

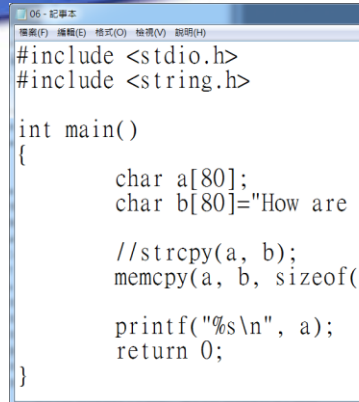
講師：張傑帆

CSIE, NTU

# C++開發工具

- 整合式開發環境

## Integrated Development Environment (IDE)

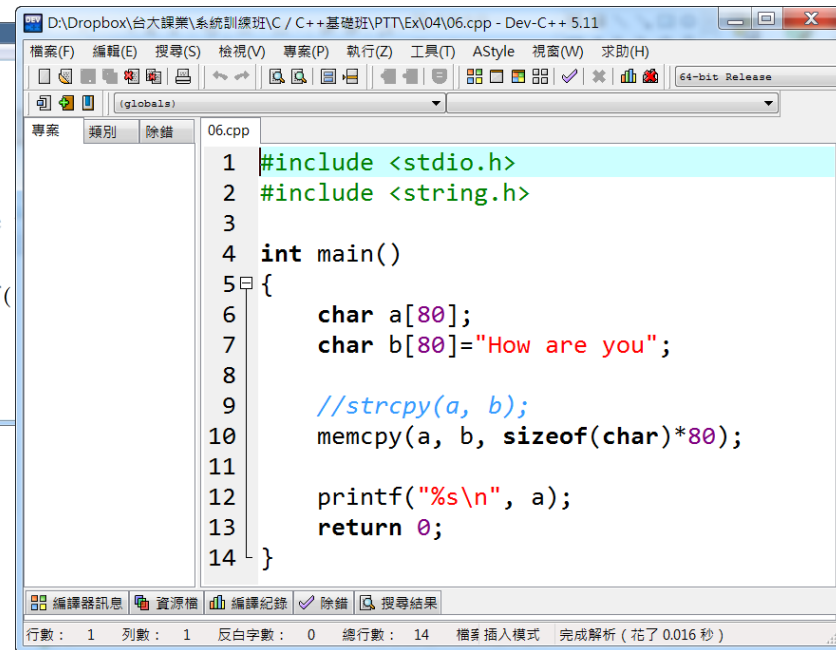


```
#include <stdio.h>
#include <string.h>

int main()
{
    char a[80];
    char b[80]="How are

    //strcpy(a, b);
    memcpy(a, b, sizeof(

    printf("%s\n", a);
    return 0;
}
```



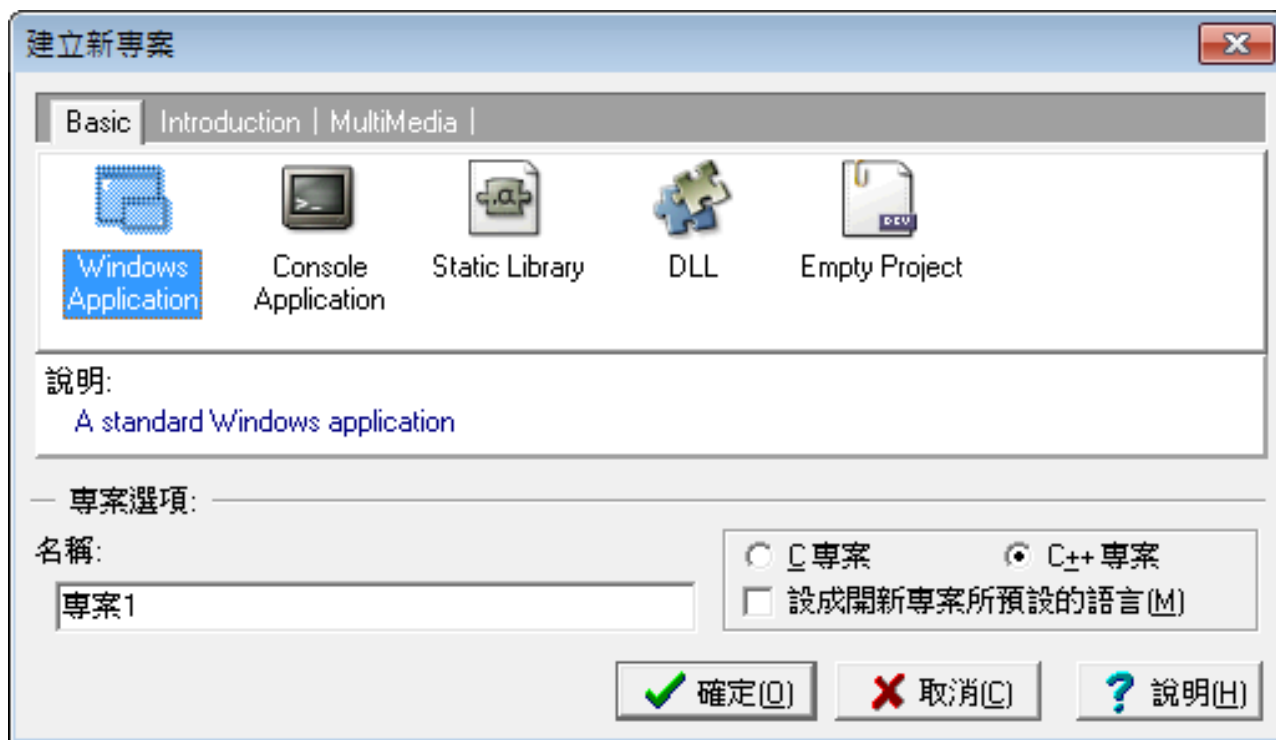
```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main()
5 {
6     char a[80];
7     char b[80]="How are you";
8
9     //strcpy(a, b);
10    memcpy(a, b, sizeof(char)*80);
11
12    printf("%s\n", a);
13    return 0;
14 }
```

- 是整合編輯、編譯、測試、除錯、與執行等功能的程式開發軟體
- 例如Borland公司的C++ Builder、IBM公司的VisualAge C++、Microsoft公司的Visual C++ 等都是整合式的C++ 程式開發軟體

## 下載Dev C++

- 在 Google 搜尋 “Dev C++”，並選擇搜尋所有網頁，則可找到BloodShed Software Dev C++的官方網站。
- 衍生版本
- 下載最新版的Orwell Dev-C++ 5.11版  
阿榮免安裝版  
Orwell's Engine 官網

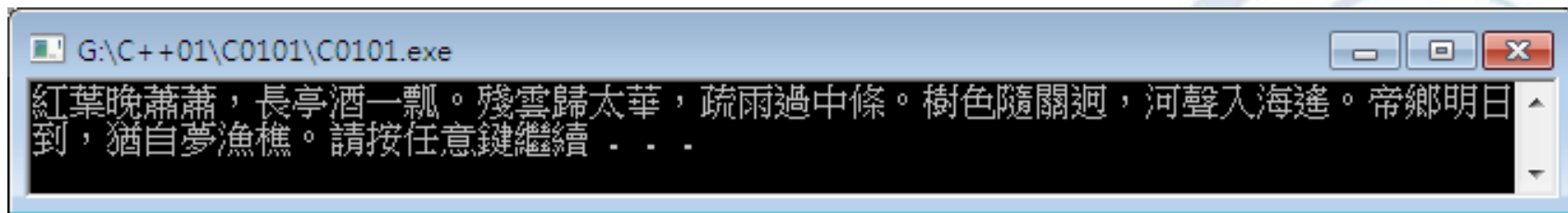
# 使用 Dev C++ 5.0





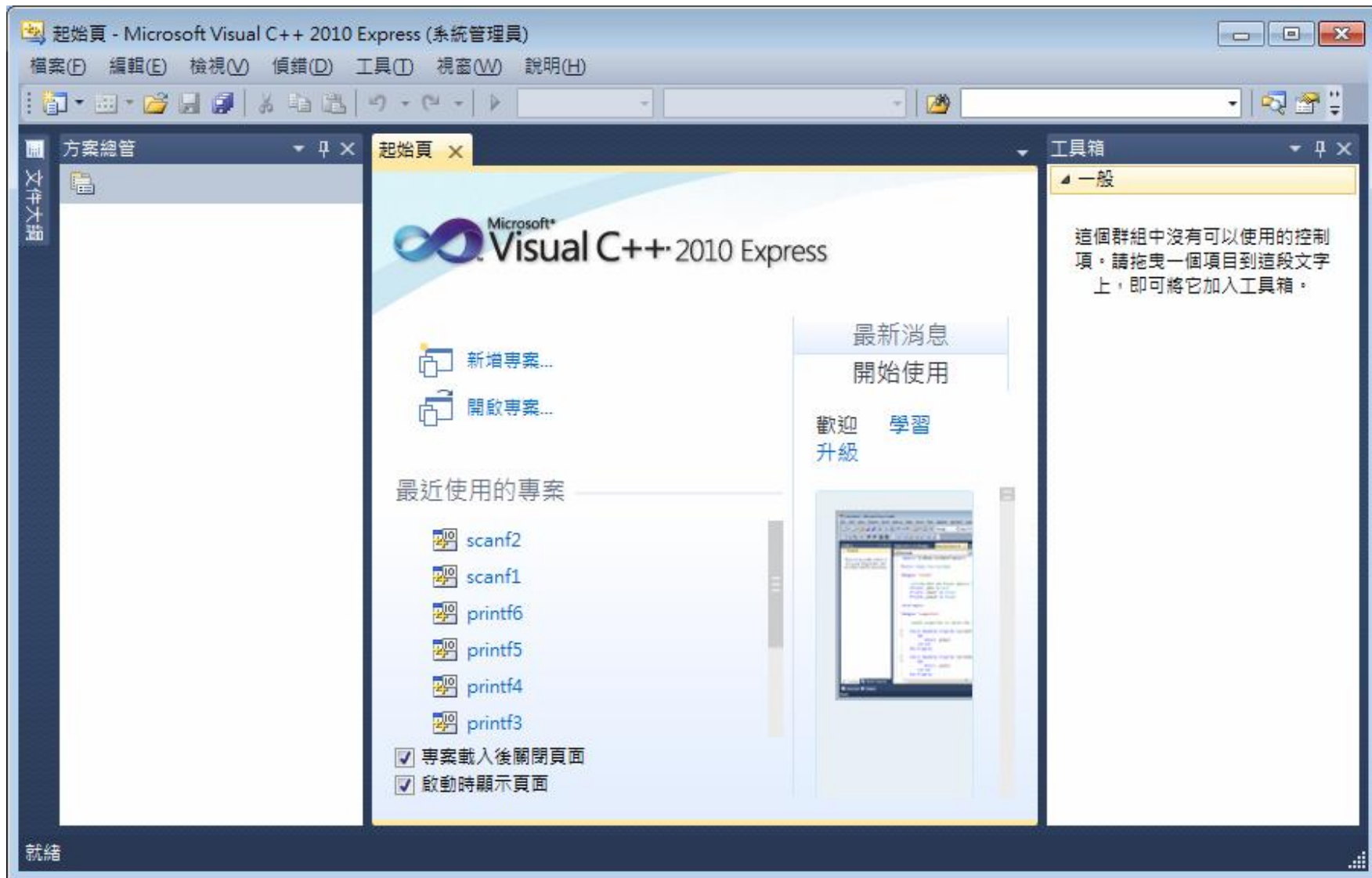
## 編譯與執行

2. 由於這是新專案，所以Dev-C++ 會要求選擇儲存main.cpp的位置，預設儲存位置是與專案相同的目錄，因此按存檔就可以了。
3. 編譯完成且程式沒有語法錯誤後，出現命令提示字元視窗，並根據cout指令輸出字串如下：



```
G:\C++01\C0101\C0101.exe
紅葉晚蕭蕭，長亭酒一瓢。殘雲歸太華，疏雨過中條。樹色隨關迴，河聲入海遙。帝鄉明日
到，猶自夢漁樵。請按任意鍵繼續 . . .
```

# 使用 Visual C++ 2010



# Visual C++

- Visual C++ 2010，可以上微軟（Microsoft）官網下載及安裝 Visual C++ 2010/2012/2013 Express。



# C/C++基礎程式設計

C語言入門、變數、基本處理與輸入輸出

講師：張傑帆  
CSIE, NTU

求知若飢，虛心若愚。

*Stay hungry. Stay foolish. -Steve Jobs 2005*

# 課程大綱

- C語言入門
- 資料型態與變數
- 基本資料處理
- 基本輸出入函式



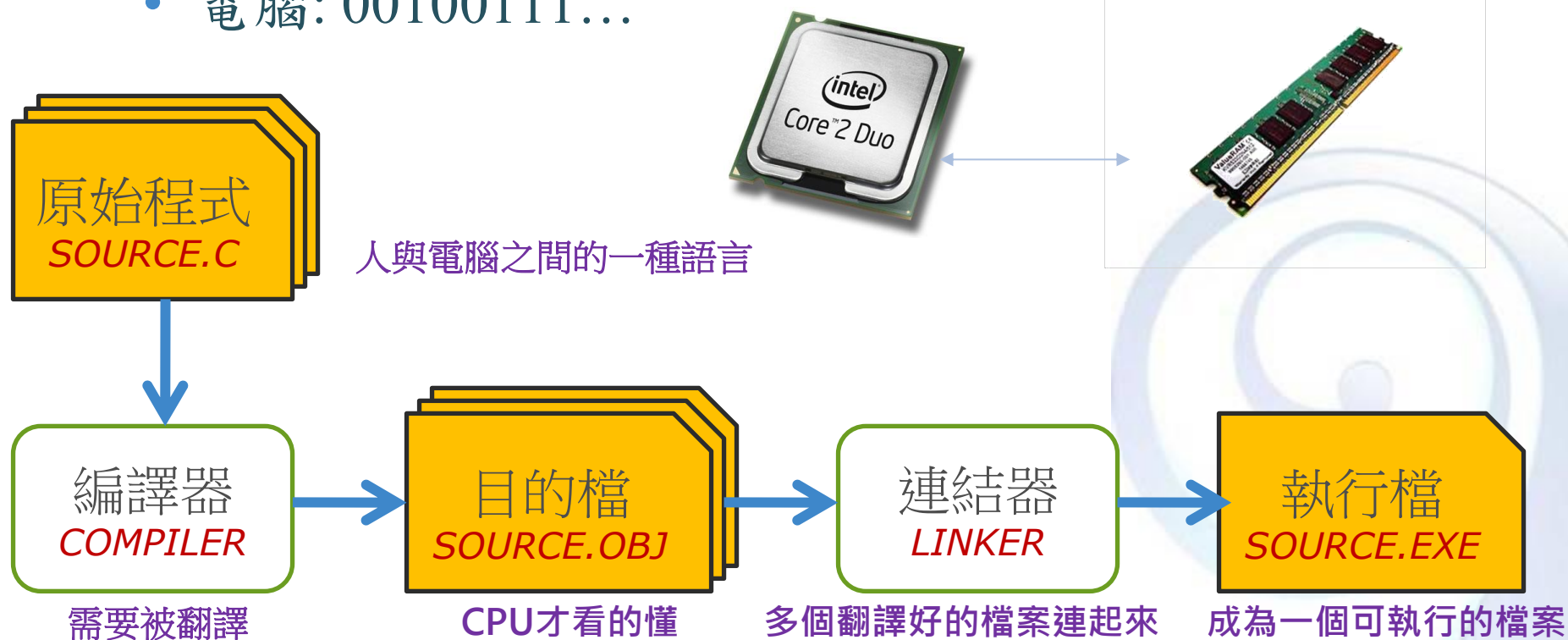
# C語言執行過程

- 程式語言？語法介於人與電腦之間，  
用來命令電腦做事的一種語言！

- 人類：中文，英文，...
- 電腦：00100111...

控制 (邏輯),  
運算 (數學)

(暫時)儲存資料





# 標準C語言格式

- 要點

- 程式有開始與結束
- 一個口令一個動作 (以分號 ; 代表一個動作的結束)
- 宣告要用的資料(記憶體), 然後寫程式操作它們(CPU)

<b>#include &lt;stdio.h&gt;</b>	----> 呼叫標準輸入輸出函式庫
<b>int main()</b>	----> 主程式 (開始程式)
<b>{</b>	
[宣告要用到的資料]	
[寫程式]	
<b>return 0;</b>	----> 回傳0 (結束程式)
<b>}</b>	

# 第一個C程式 – 編譯示範

- **HelloWorld**
  - 使用 Visual C++ / Dev C++
  - 編譯並執行

```
#include <stdio.h>
int main()
{
    printf("Hello World !\n");
    return 0;
}
```

# 註解

- 單行註解: **//** 內容
- 將一範圍註解: **/\*** 內容 **\*/**

```
#include <stdio.h> //這行是呼叫標準輸入輸出函式庫
int main() //這行是主程式
{
    printf("Hello World !\n");
    //printf("這行不會真的印出來!\n");
    /*printf("這行跟下行也是!\n");
    printf("不會印!\n");*/

    return 0;
}
```

# 課程大綱

- C語言入門
- 資料型態與變數
- 基本資料處理
- 基本輸出入函式

# 資料型態

- 字元(char)

- Ex. char a = 'A'

- 整數(int)

- Ex. int b = 123

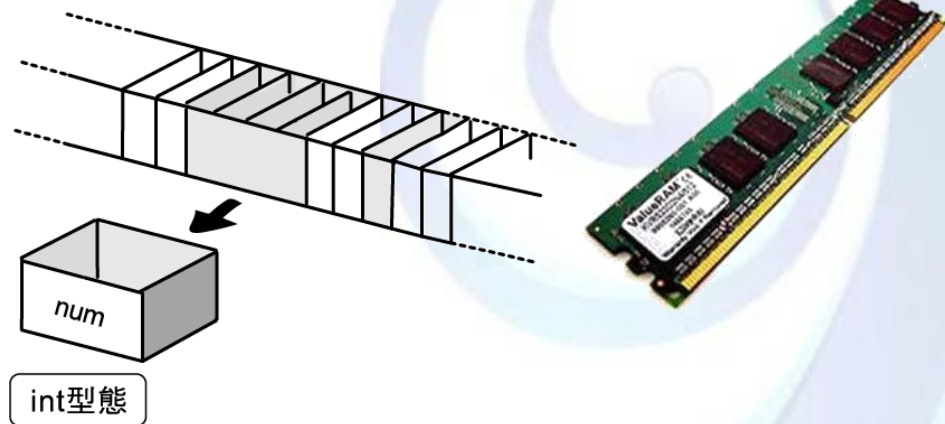
- 浮點數(double)

- Ex. double c = 123.456

- 資料可以透過變數存取!

變數名稱	型態	內容	記憶體位址
a	char	'A'	0x1000
b	int	123	0x1001
c	double	123.456	0x1005

A (字元)  
123 (整數)  
123.456 (浮點數) ...



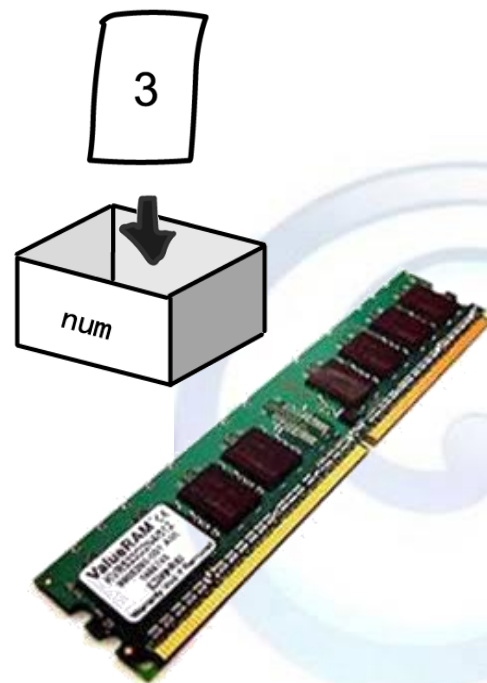


# 變數

- 當我們使用CPU處理資料，常會需要把資料做儲存與讀取的動作
- 使用對的資料型態的變數存放對的資料！
- 使用等號儲存 ex: `int num = 3;`
- 規則：先宣告，再使用！



123 (整數)  
123.456 (浮點數)  
A (字元)



# 宣告變數

- 語法:

- 宣告一個變數: 資料型態 變數名稱;
- 宣告多個變數: 資料型態 變數名稱1, 變數名稱2, ..., 變數名稱n;
- 宣告變數並初始化: 資料型態 變數名稱 = 內容;

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int num1 = 123;
```

```
    double num2 = 123.456;
```

```
    char ch1 = 'A';
```

```
    double num3;
```

```
    num3 = num1 + num2;
```

```
    return 0;
```

```
}
```

```
//宣告num1為整數變數
```

```
//宣告num2為小數變數
```

```
//宣告ch1為字元變數
```

```
//宣告num3為小數變數
```

```
//將num1+num2之結果存到num3
```

# 變數命名的原則

- 開頭必須是英文字母或底線
- 字元的大寫小寫所代表的意義不同。
- 不得使用關鍵字(Keyword)

```
#include <stdio.h>
int main()
{
    int abc = 10;
    int ABC = 20;
    int 123a = 30; //錯誤: 變數開頭為數字
    int char = 40; //錯誤: 變數為關鍵字char
    return 0;
}
```

# 關鍵字(Keyword)

- 關鍵字(Keyword)又稱保留字。
- 是事先賦予某個識別字特別的用途，以供程式設計呼叫或使用，**不允許重複使用**。
- 下表為ANSI C所提供的保留字，透過這些保留字、配合運算子(Operator)和分隔符號(Seperator)，就定義出C語言所提供的各種敘述(Statement)：

## 常見關鍵字(Keyword)

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
Short	signed	sizeof	static	struct
switch	typeof	union	unsigned	void
volatile	while	typedef		



# 變數的資料型態種類

- 1 byte可放0~255大小之任一數字
- 不同變數能存的值之範圍有限
- 操作資料請注意所宣告的變數特性

類別	符號位元	容量(bytes)	表示法	數值範圍
整數	有	2	short	-32768~32767
		4	<b>int</b>	-2147483648~2147483647
	無	2	unsigned short	0~65535
		4	unsigned int	0~4294967295
浮點數	有	4	float	$10^{-38} \sim 10^{38}$
		8	<b>double</b>	$10^{-308} \sim 10^{308}$
字元	有	1	<b>char</b>	-128~127
	無	1	unsigned char	0~255

## 各資料型別的有效範圍 (視編譯器不同)

資料型別	常值種類	長 度	有效範圍
char	字元	1 Byte(8 Bits)	-128~127
short	短整數	2 Bytes(16 Bits)	-32768~+32677
int	整數	4 Bytes(32 Bits)	-23147483648 ~ 2147483647
long int	長整數	4 Bytes(32 Bits)	-23147483648 ~ 2147483647
unsigned char	無正負號字元	1 Byte(8 Bits)	0~255
unsigned short	無正負號整數	2 Bytes(16 Bits)	0 ~ 65535
unsigned int	無正負號整數	4 Bytes(32 Bits)	0 ~ 4294967295
unsigned long int	無正負號整數	4 Bytes(32 Bits)	0 ~ 4294967295
float	單精確實數	4 Bytes(32 Bits)	$-3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$
double	倍精確實數	8 Bytes(64 Bits)	$-1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$
long double	長倍精確實數	12 Bytes(96 Bits)	$-3.47 \times 10^{-4932} \sim 1.1 \times 10^{4932}$

# 小練習 觀察不同型態的變數大小-Debugger介紹

```
#include <iostream>
```

```
int main()
```

```
{
```

```
    sizeof(int);
```

```
    sizeof(double);
```

```
    return 0;
```

```
}
```

- 不同編譯器略有不同
- 用 `sizeof()` 函式來觀察變數大小(byte)

D:\Dropbox\台大課業\系

```
char 1
short 2
short int 2
int 4
long int 4
unsigned char 1
unsigned short 2
unsigned int 4
unsigned long int 4
float 4
double 8
long double 16
long long 8
```

D:\Dropbox\台大課業\系統

```
char 1
short 2
short int 2
int 4
long int 4
unsigned char 1
unsigned short 2
unsigned int 4
unsigned long int 4
float 4
double 8
long double 12
long long 8
```

C:\Windows\system32\cr

```
char 1
short 2
short int 2
int 4
long int 4
unsigned char 1
unsigned short 2
unsigned int 4
unsigned long int 4
float 4
double 8
long double 8
long long 8
```

# 課程大綱

- C語言入門
- 資料型態與變數
- 基本輸出入函式
- 基本資料處理



# 基本輸入輸出

- `scanf`
  - 預設為鍵盤輸入
- `printf`
  - 預設為螢幕輸出

123 (整數)





# 基本輸入輸出

- 範例：用鍵盤輸入一個整數，再將它由螢幕輸出

```
#include <stdio.h>
int main()
{
    int num;

    scanf("%d", &num); // 鍵盤輸入之整數存到num
    printf("%d\n", num); // 將整數num由螢幕輸出

    return 0;
}
```

```
printf("%c 是文字， %d 是整數。 \n", 'A', 123);
```

# 基本輸出函數

```
printf("%c 是文字。 \n", 'A');  
printf("%d 是整數。 \n", 123);  
printf("%f 是小數。 \n", 10.5);
```

輸出文字

輸出整數

輸出小數

- 格式化輸出函數 printf()

- 語法如下：`printf("%d %d \n", num1, num2);`

- `printf("格式化輸出內容", 參數1, 參數2, ... 參數n);`
  - 格式化輸出內容, 可加入列印格式、控制字元、修飾子
  - 參數, 為對應格式之資料內容 (可為變數, 運算式, 常數)

- 要訣:

- 參數: 一個%對應一個參數
- 列印格式: %後可加對應參數型態之列印格式
- 修飾子: %後可加修飾子對列印格式做排版
- 控制(逸出)字元: 一般鍵盤無法輸入的字元要用控制字元

會顯示在這個位置

```
printf("%C 為字元。 \n", 'A');
```



A 為字元。

- Ex. 在螢幕印一整數變數內容並斷行

- `printf("%d\n", num);`

## ● 格式化輸出內容

```
printf("歡迎來到C語言！\n開始使用C語言吧！\n");
```

控制字元	功能
\a	警告音
\b	倒退
\f	換頁
\n	換行
\r	歸位
\t	跳格
\'	印出單引號
\"	印出雙引號
\\	反斜線
\/	斜線
\d	八進位 Ascii 碼
\x	十六進位 Ascii 碼--



\

# 修飾子

1 2 3 4 5

%3d

3.141592

%.3

## • 格式化輸出內容

修飾子	功能	範例
-	向左對齊	%-3d
+	將數值的正負號顯示出來	%+5d
空白	數值為正值時，留一格空白；為負值時，顯示負號	% 6f
0	將固定欄位長度的數值前空白處填上 0； 與 - 修飾子同時使用時，此修飾子無效	%07.2f
數字	欄位長度，當數值的位數大於所定的欄位長度時， 欄位會自動加寬它的長度	%9d
.	數值以 %e, %E, %f 型式表示時，決定小數點後所要顯示的位數	%4.3f
h	表示 short int 或是 unsigned short int	%h
l	表示 long int 或是 unsigned long int	%l

# 基本輸出函數

- 範例：

```
#include <stdio.h>

int main()
{
    int x=42;
    double y=12.345;
    char c='A';

    printf("%d %lf %c\n",x, y, c);
    printf("|%15d|\n", x); //空15格
    printf("|%-15d|\n", x); //向左對齊
    printf("|%015d|\n", x); //前面空白處填0
    printf("|%10lf|\n", y); //long用，不是也沒關係
    printf("|%10.3lf|\n", y); //限小數點以下第三位
    return 0;
}
```



# 基本輸入函數

- 格式化輸入函數 `scanf()`

- 語法如下：

- `scanf("格式化輸入內容", &參數1, &參數2, ...&參數n);`

- 格式化輸入內容, 格式與printf同

- `&參數`, 為對應格式之變數記憶體位置

- 要訣:

- `&參數`:

- 一個%對應一個變數記憶體位置, 輸入時以空白鍵, TAB, Enter 區隔, 以Enter做為輸入結束

- 列印格式: %後可加對應參數型態之格式

- 修飾子: 一般不使用

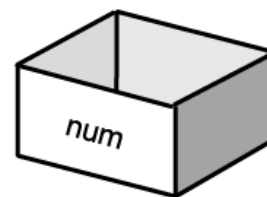
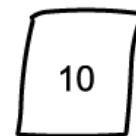
- 控制字元: 一般不使用

- Ex. 將鍵盤輸入之兩個整數存到兩個整數變數

- `scanf("%d%d", &num1, &num2);`

當程式處理到scanf時, 會呈現等待由鍵盤輸入的狀態

輸入完畢後按下Enter鍵便會讀入為變數



讀入為此變數

```
scanf("%d", &num);
```



# 基本輸入函數

- 範例：

```
#include <stdio.h>

int main()
{
    int num1;
    double num2;
    char ch;

    scanf("%c", &ch);
    scanf("%d", &num1);
    scanf("%lf", &num2);

    printf("%c\n", ch);
    printf("%d\n", num1);
    printf("%lf\n", num2);
    return 0;
}
```

# 課程大綱

- C語言入門
- 資料型態與變數
- 基本輸出入函式
- 基本資料處理

# 算數運算子

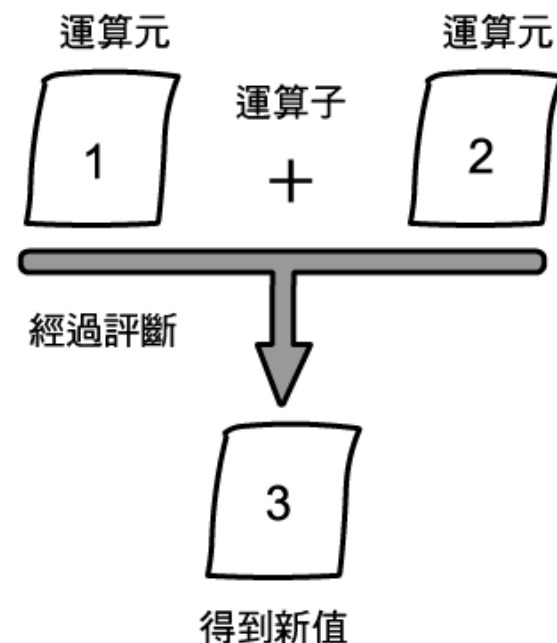
- 加、減、乘、除、負數及餘數運算子
  - 加(+)、減(-)、乘(\*)、除(/)為一般的四則運算
  - 而餘數運算(%)是經由兩整數相除所得的餘數稱之。  
負數是在某個常數前面加個減號所成的組合稱之。
- 運算子的優先權

- 運算子，其運算的優先順序如下：

負號 (-)
乘 (*)、除 (/)、餘數 (%)
加 (+)、減 (-)

←高優先順序  
←中優先順序  
←低優先順序

- 使用括號()改變優先順序！

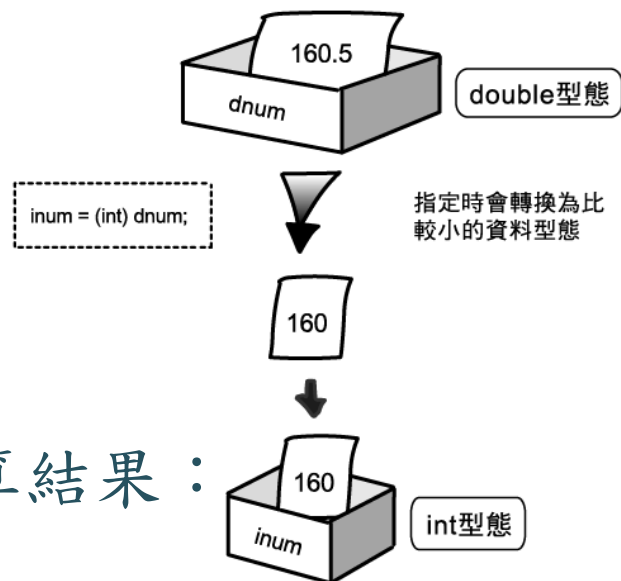


下表為各種運算子在運算式中優先執行順序：

優先次序	運算子 (Operator)	運算次序
1	.(成員存取運算子)、f(x)、a[x]、()	由內至外
2	!、~、(cast)、+(正號)、-(負號)、++x、--x	由內至外
3	*(乘)、/(除)、%(取餘數)	由左至右
4	+(加)、-(減)	由左至右
5	<<(左移)、>>(右移) (註)	由左至右
6	<、<=、>、>=(關係運算子)、	由左至右
7	==(相等)、!=(不等於)	由左至右
8	&(Address of) &a(變數 a 的位址)	由左至右
9	&(位元 AND)	由左至右
10	^(位元 XOR)	由左至右
11	(位元 OR)	由左至右
12	&&(條件式 AND)	由左至右
13	(條件式 OR)	由左至右
14	? : (條件運算子)	由右至左
15	=、+=、-=、*=、/=、%=、<<=、>==、&=、^=、!=	由右而左
16	, (逗號)	由左至右

# 整數與小數的除法

- 注意以下範例之**整數之除法**!!
  - 整數除法結果為整數
  - 使用型態轉換語法改變運算結果：  
(資料型態) 運算式



```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
int a1=46, a2=5;
```

```
double b1=46, b2=5;
```

```
double x, y;
```

```
x = a1/a2; // x = 9
```

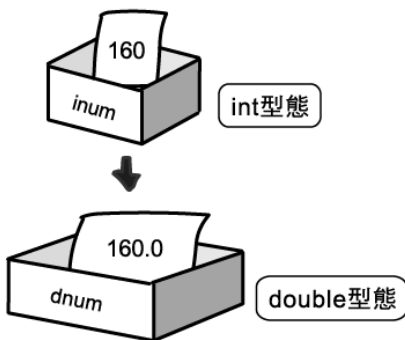
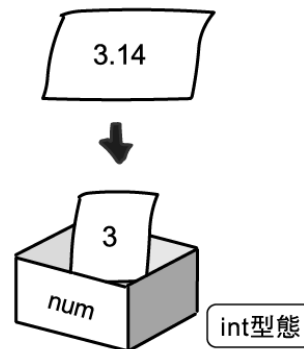
```
y = b1/b2; // y = 9.2
```

```
//使用型態轉換把a1/a2用小數運算表示
```

```
x = (double)a1/a2; // x = 9.2
```

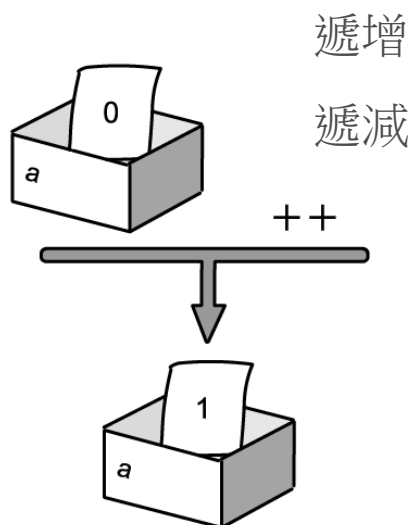
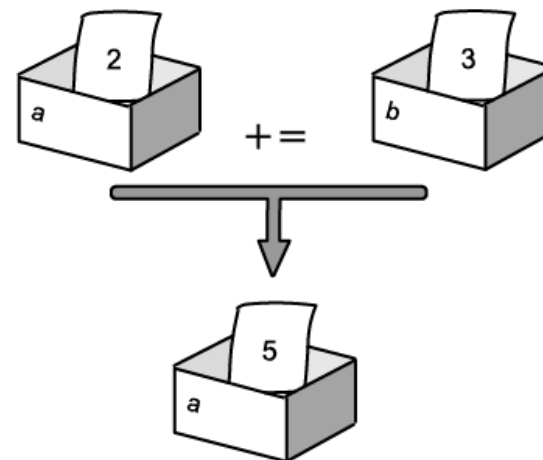
```
return 0;
```

```
}
```



# 特殊運算式

- C/C++常用的特殊運算式:



特殊運算式	基本運算式
$a++$	$a = a + 1$
$a--$	$a = a - 1$
$a += b$	$a = a + b$
$a -= b$	$a = a - b$
$a *= b$	$a = a * b$
$a /= b$	$a = a / b$
$a \% = b$	$a = a \% b$



# 特殊運算式

- ▶ 觀察下面整數a的變化

```
#include<stdio.h>
int main()
{
    int a = 0;

    a+=5;
    a++;
    a/=3;
    a*=5;
    a-=4;
    a%=3;
    return 0;
}
```

# 複合指定運算子

## 語法

變數 = 變數 運算子 運算式 ;      (x = x + 3;)

變數 運算子 = 運算式 ;      (x += 3;)

運算子	功能	實例(假設每列 x 初值為 5)	結果
=	指定	x = 5	x $\Leftarrow$ 5
+=	相加後再指定	x += 4 相當於 x = x + 4	x $\Leftarrow$ 5+4=9
-=	相減後再指定	x -= 2 相當於 x = x - 2	x $\Leftarrow$ 5-2=3
*=	相乘後再指定	x *= 3 相當於 x = x * 3	x $\Leftarrow$ 5*3=15
/=	相除後再指定	x /= 2 相當於 x = x / 2	x $\Leftarrow$ 5/2=2.5
%=	相除取餘數後再指定	x %= 2 相當於 x = x % 2	x $\Leftarrow$ 5%2=1

# 特殊運算式

- ▶ 觀察下面整數a與b的變化

```
#include<stdio.h>
int main()
{
    int a = 0;
    int b;

    a+=10;
    b = a++;
    b = ++a;
    b = a--;
    b = --a;
}
```

## 遞增和遞減運算子

遞增運算式	一般運算式	結果
<code>b=++a;</code>	<code>a=a+1;</code> <code>b=a;</code>	<code>a=11 , b=11</code>
<code>b=a++;</code>	<code>b=a;</code> <code>a=a+1;</code>	<code>a=11 , b=10</code>
<code>b=--a;</code>	<code>a=a-1;</code> <code>b=a;</code>	<code>a=9 , b=9</code>
<code>b=a--;</code>	<code>b=a;</code> <code>a=a-1;</code>	<code>a=10 , b=9</code>

# 邏輯與關係運算子

- 一般用於**控制流程**等有條件的敘述當中
- 常用來**比較或判斷**變數內容資料為多少

運算子	功能敘述
&&	AND(且)
	OR(或)
!	NOT(非)

關係運算子	
運算子	功能敘述
<	小於
<=	小於等於
>	大於
>=	大於等於
==	等於
!=	不等於

# 基本運算子優先順序表

- 可使用括號改變優先順序

高



低

!, 負號 (-), ++, --
乘 (*), 除 (/), 餘數 (%)
加 (+), 減 (-)
<, <=, >, >=
==, !=
&&



## 練習

- 計算梯形面積存入一個整數變數, 並使用偵錯模式觀察結果  
(上底=2, 下底=6, 高=4)
- 梯型公式:  $(\text{上底} + \text{下底}) \times \text{高} \div 2$

## 練習

- 輸入兩個小數, 印出兩者相加的結果  
(輸出小數後兩位)
- 例如:
  - 輸入: 5.1 及 2.3
  - 輸出:  $5.10 + 2.30 = 7.40$



# 回家作業 (Hw1-calc.c)

- 製作一個簡單的計算機
- 功能
  - 數字的+, -, \*, / (不需要判斷除以0之情況)
- 輸入輸出格式 (格式必須與下面程式結果一致)
  - 連續輸入兩個數字，以空白鍵分開
  - 按下Enter後算出結果 (顯示小數點後兩位)
- 程式結果:

```
請輸入兩個數字: 5.1 2.3
計算結果:
5.10 + 2.30 = 7.40
5.10 - 2.30 = 2.80
5.10 * 2.30 = 11.73
5.10 / 2.30 = 2.22
請按任意鍵繼續 . . . _
```