

01 環境

- [1.1 C語言的發展歷史](#)
- [1.2 C語言特性](#)
- [1.3 電腦的組成](#)
- [1.4 作業系統](#)
- [1.5 C的環境](#)
- [1.6 Dev-Cpp整合發展環境](#)
- [1.7 為什麼要學習程式設計](#)

◀ ◇ ▶ ▷

C 語言與電腦的生命力

- 電腦的表現令多數人覺得不可思議...
 - 電腦的硬體是死的、沒有生命的
 - 電腦的軟體是活的，讓它動起來的
- **目前使用最多的軟體發展工具非C語言莫屬**
- 本書提供「ANSI C」語言的程式設計理念
- 「ANSI C」版本是在公元一九八九年由美國「American National Standards Institute」簡稱「ANSI」以及「International Standards Organization」簡稱「ISO」，所製定的。
- 公元一九九九年更新過，稱為「C99」版，其標準文件可參考「ISO/IEC 9899:1999」，其網址如下：
<http://www.ansi.org>

01 C/C++ Common sense

◀ ◇ ▶ ▷

1.1 C 語言的發展歷史

年代	記事
1963	Combridge發展出CPL語言。
1967	Martin Richards發展出BCPL語言。
1970	美國AT&T公司貝爾實驗室的Ken Thompson將BCPL語言改良成B語言。
1972	Kenighan及Ritchie以PDP-11電腦為基礎，在UNIX作業系統下將B語言改為C語言，C語言提供各種資料型態。
1978	由Brian W.Kernighan及Dennis W.Ritchie（簡稱K&R）合著一本C語言的書，「THE C PROGRAMMING LANGUAGE」，為C語言的經典。
1979	各單位陸續推出各種不同版本之C語言，如C86、MS C、Run C、Quick C、... 等等。
1983	為了統合各種C語言，美國國家標準協會ANSI，成立「ANSI委員會」，定義了一套ANSI的C語言標準。
1999	更新ANSI的C語言標準，稱為C99版。

01 C/C++ Common sense

◀ ◇ ▶ ▷

1.2 C語言特性

1. 高階架構，低階功能
2. 可攜性高
3. 函式導向
4. 指標運算 → 系統程式設計大部份使用C語言
5. 遞迴呼叫
6. 動態資料結構
7. 結構化程式設計
8. 自由格式
9. 函式不允許巢狀結構

01 C/C++ Common sense

◀ ◇ ▶ ▷

1.3 電腦的組成 VS Programming

Programming skill

- Prompt (hint/ notation)
- Input (read)
- Retrieve from file
- Process (calculation/mapping/transfer)
- Output (report from/ tabularization)
- Store into file

01 C/C++ Common sense

◀ ◇ ▶ ▷

1.4 C and Operating systems

- 作業系統有很多種，如 UNIX、Linux、Windows 95/98/2000/NT/XP/2003等等，您要安裝那一種作業系統，要看您的需求而定，大體上Windows作業系統偏向個人單機使用，NT、UNIX、以及Linux等提供多人同時使用。
- 本書使用的「ANSI C」語言這些作業系統都支援，因此您安裝那一個作業系統都沒有關係。

01 C/C++ Common sense

1.5.1 Windows環境 編輯編譯及執行

```

1 // Filename: hello.c
2 // purpose: A first program in C
3 // Author: Ming-Chang Chen, name@classmate
4 // Date: 2008/03/01. Version 1.00
5
6 #include <stdio.h>
7 #include <stro.h>
8
9 /* Function main begins program execution */
10 int main()
11 {
12     printf("Hello, my sweet\n");
13     return 0;
14 }
15 system("pause");
16 exit(0) /* indicate that program ended successfully */
17 /* (c) creation main */

```

01 C/C++ Common sense

1.5.2 Linux環境的編輯編譯及執行

您使用vi編輯hello.c，然後編譯、執行，其步驟如下：

```

$ vi test.c [註] 步驟 1
/****** hello.c *****/
int main()
{
    printf("Hello, world!!!");
    return 0;
}
$ cc hello.c [註] 步驟 2、3、4
$ ./a.out [註] 步驟 5、6
Hello world!!

```

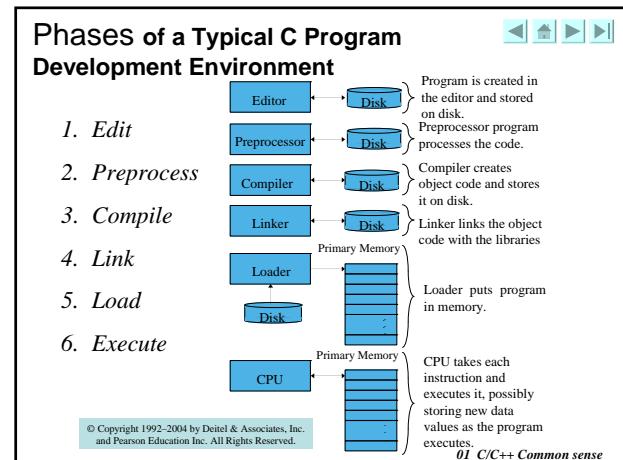
01 C/C++ Common sense

1.6 C語言的程式設計

- 啓動 Dev-C++ 軟體。
- 編輯/載入/除錯 hello.c 程式。
- 編譯 hello.c 程式。
- 執行 hello.exe 程式。
- 關閉 hello.c 視窗(DOS emulation)。

Dev-Cpp整合發展環境

01 C/C++ Common sense



1.6 C語言的程式設計(Cont.)

How to develop a workable program

- Problem analysis
- Algorithm design
- Coding
 - ◆ Structural programming
 - ◆ Object oriented programming
- Program validation/verification

01 C/C++ Common sense



Textbook for C/C++ programming



林邦傑，C/C++程式語言，全華書局。

References for C/C++ programming (Cont.)

陳錦輝，C/C++初學指引，上奇書局。
陳會安，C/C++程式設計範例教本，學貢書局。
吳國樑，C:程式設計藝術(第四版)，全華書局。

01 C/C++ Common sense

References for C/C++ programming (Cont.)



楊建貴，C入門與Turbo系統
蔡明志譯，C語言詳論，東華書局。
吳明哲等人，最新C&C++學習範本，松嶺書局
莊益瑞，吳權威，C語言程式設計，碁峰資訊。

01 C/C++ Common sense

Valuable Web sites for C/C++ learning



<http://www.cci.pu.edu.tw/~ychu/class962/962-C-Programming.htm#4>
<http://www2.lssh.tp.edu.tw/~lib/teach/note/c/c.htm>
<http://csuweb.csu.edu.tw/~k0024/cclass/c.htm>
<http://www.tcgs.tc.edu.tw/~sagit/cpp/>
<http://dhcp.tcgs.tc.edu.tw/c/>
<http://www.cppreference.com/>

Learning from experience.

01 C/C++ Common sense

1.7 為什麼要學習程式設計



1. 了解智慧型設備的運作，
2. 讓自己的邏輯概念更清晰完整，
3. 培養開發智慧型設備的基本準備功夫，
4. 透過程式設計的訓練，了解物質世界運作的道理，
5. 賺錢，改善生活；創業，造福人群。

01 C/C++ Common sense

**程式設計風格
(Programming Styles)**

- 非結構化程式設計
(Unstructured Programming)
- 結構化程式設計
(Structured Programming)
- 模組化程式設計
(Modular Programming)
- 物件導向程式設計
(Object-Oriented Programming)

02 程式設計簡介

2.0 純粹DOS環境下的程式設計

- [2.1 顯示字串程式](#)
- [2.2 兩數相加程式](#)
- [2.3 變數與記憶體](#)
- [2.4 運算式評值](#)
- [2.5 邏輯判斷](#)

2.0 純粹DOS環境下的程式設計

文字型嵌入式電子設備

- 電子字典
- 手機選單選項
- 跑馬燈字幕機
- 學生成績管理系統
- 收銀機
- 倉庫管理系統
- 點菜機
- 警察查詢警車系統
- 俄羅斯方塊遊戲機
- 跳舞機
- 簡易型計算機
- 數位電話機管理
- 「ㄅ一ㄅ一」叮呼叫器
- 電視機遙控器
- 火車列車長補票機
- 運動健身器材
- 火車站班次出發時間
- 車廂編號班次啓程站終點站

02-程式設計簡介

模組化程式設計

- C語言的功能都大部分是由函式庫提供，模組可以將實際處理的程式碼和資料隱藏起來達成「資訊隱藏」(Information Hiding)。

陳會安，C/C++程式設計範例教本，學貢書局

02-程式設計簡介

Tips for C program writing

- 程式碼要正確縮排，有益於程式內容的了解
- 程式註解有益於程式內容的了解及設計理念的追蹤
- 程式區塊 (Blocks) 是程式的基本單元
 - 由「{」和「}」符號包圍，是由多個敘述所組成
- main()是C語言第一個執行的函數名稱
- 函數是可再用的程式基本單元(reusable function unit)
 - 輸入 (Input)
 - 輸出 (Output)
- 演算法必須留意
 - 明確性 (Definiteness)
 - 有限性 (Finiteness)
 - 有效性 (Effectiveness)

02-程式設計簡介

2.1 顯示字串程式

```
// filename: first.cpp
// purpose: A first program in C
// Authors / ID code: Ping-Sung Liao/A123456
// Date: 2008/03/01, Version: 1.00

#include <stdlib.h>
#include <stdio.h>

int main()
{
    printf("歡迎到 阿拉丁魔幻世界!\n");

    system("pause");
    return 0; /* indicate that program ended successfully */
} /* end function main */

```

02-程式設計簡介

C 語言逸出符號

逸出順序	名稱
\	單引號 (single quote)
\"	雙引號 (double quote)
\?	問號 (question mark)
\\\	背斜線 (back slash)
\a	鈴鈴 (ring the bell)
\b	退位 (backspace)
\f	換頁 (form feed)
\n	新列 (newline)
\r	返回 (carriage return)
\t	水平定位 (horizontal tab)
\v	垂直定位 (vertical tab)
\NNN	三位八進位數字 (octal digit)，如 \007 表示 beep 聲音。
\xNN	x 後跟著兩位十六進位數字，如 \x07 表示 beep 聲音。

02-程式設計簡介

C 語言逸出符號範例

```
printf("我的");
printf("第一個C程式\n");
```

```
printf("我的");
printf("第一個");
printf("C程式\n");
```

```
printf("我的\n第一個\nC程式\n");
```

02-程式設計簡介

2.2 兩數相加程式

```
//Filename: add2number.cpp
//Purpose: 兩數相加程式
//Authors: Ping-Sung Liao/A123456
//Date: 2008/03/01, Version: 1.00

#include <stdlib.h>
#include <stdio.h>

int main()
{
    int a, b, sum;
    printf("請輸入第一個整數："); // 提示使用者
    scanf("%d", &a); // 使用者輸入資料
    printf("請輸入第二個整數："); // 提示使用者
    scanf("%d", &b); // 使用者輸入資料
    sum=a+b; // 資料處理中
    printf("兩數之和為 %d\n", sum); // 列印資料處理結果
    system("pause"); // 程式暫停一下
    return 0; // 程式執行完畢，回到視窗作業系統
}
```

02-程式設計簡介

Naming rules for variable/function (1)

- 名稱是一個合法的「識別字」（Identifiers）
 - 第一個字母必須是底線 _ 或字母 a - z, A - Z
 - 第二個字母可以是英文字母、數字和底線 _ 字元組成的名稱。
 - C語言的識別字至少前31個字元是有效字元
- 名稱區分英文字母的大小寫：
 - count、Count和COUNT屬於不同的識別字。
- 名稱不能使用C語法的「關鍵字」（Keywords）
 - 關鍵字對於編譯程式而言，擁有特殊意義。
- 名稱必須是有意義：建議採用匈牙利命名；例如 myStudentNumber, classIdentifier;
- 名稱擁有其有效「範圍」（Scope）
 - 在有效範圍的程式碼中名稱必需是唯一的

02-程式設計簡介

Naming rules for variable/function (2)

合法名稱	不合法名稱
T、c	1、2
Size、test123、count、_hight	1count、hi!world
Long_name、helloWord	Long...name、hello World

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

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

02-程式設計簡介

C 語言關鍵字

- 識別字對C COMPILER來講有其特別的意義，
- 我們取變數名稱時不能跟C語言關鍵字相同，C語言關鍵字如下表所示。

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

02-程式設計簡介

2.3 變數與記憶體

02-程式設計簡介

◀ ▶ 🔍

2.4 運算式計值

- 許多C程式都會使用算術計算，C語言的算術運算子有加'+'、減'-'、乘'*'、除'/'、及餘數'%'，如下表所示。

表 2.2 C 語言算術運算子

名稱	運算子	數學表示法	C程式表示法
加	+	$a+b+2$	$a + b + 2$
減	-	$a-b-3$	$a - b - 3$
乘	*	$ab5$	$a * b * 5$
除	/	a/b	a / b
餘數	%	$a \bmod b$	$a \% b$

02-程式設計簡介

◀ ▶ 🔍

C語言數學運算子優先順序

C語言運算式計值是依運算子優先順序的規則計算的，這一點與數學上的計算是一樣的。規則如下：

- 在括號內的運算式優先計值。
若括號有好幾組，則最內層的括號優先計值。
- 乘、除、餘數第二優先計值。
乘、除、餘數是同一等級，無所謂誰先誰後，以左邊的運算子優先計值。
- 加與減第三優先計值。
加與減是同一等級，無所謂誰先誰後，以左邊的運算子優先計值。

02-程式設計簡介

◀ ▶ 🔍

Operators precedence

Operators	[]					Associativity	Type
O	[]					left to right	highest
+	-	++	--	!	*	right to left	unary
*	/	%				left to right	multiplicative
+	-					left to right	additive
<	=<	>	>=			left to right	relational
==	!=					left to right	equality
&&						left to right	logical and
						left to right	logical or
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment
,						left to right	comma

Fig. 7.5 Operator precedence.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

02-程式設計簡介

◀ ▶ 🔍

2.5 邏輯判斷 --- if控制結構

- if控制敘述可以改變程式的流程，其語法如下：

if (條件) 敘述1

「條件」若成立則執行其後的「敘述1」，
「條件」若不成立則不執行。

02-程式設計簡介

◀ ▶ 🔍

【程式ifscore.c】

```
***** ifscore.c *****/
#include <stdio.h>
int main()
{
    int score;
    printf("請輸入成績：");
    scanf("%d", &score);
    if (score >= 60)
        printf("恭喜, 您及格了!");
    return 0;
}
```

【執行於Windows環境】

```
ifscore.exe <Enter>
請輸入成績：88 <Enter>
恭喜, 您及格了!
```

02-程式設計簡介

◀ ▶ 🔍

【程式ifblock.c】

```
***** ifblock.c *****/
#include <stdio.h>
int main()
{
    int a, b, c, d;
    printf("請輸入 a,b,c 之值：");
    scanf("%d %d %d", &a, &b, &c);
    d = b * b - 4 * a * c;
    if (d > 0)
    {
        printf("判別式 d 的值 = %d\n", d);
        printf("方程式有兩個不同的實根");
    }
    return 0;
}
```

【執行於Windows環境】

```
請輸入 a,b,c 之值：1 2 3 <Enter>
判別式 d 的值 = 16
方程式有兩個不同的實根
```

02-程式設計簡介

03 流程圖與演算法

3.0 Requirement & Description

3.1 流程圖符號

3.2 演算法

◀ ◇ ▶ ▷

3.0 Requirement and Description

- 在使用電腦解決您的問題之前，您總要對問題做一個完整的了解，然後仔細的規劃解決問題的步驟，將這些步驟轉換為電腦的程式，最後執行這些程式，就可得到結果。
- **演算法(algorithm)**
 - 解決問題的步驟
 - 通常要藉助流程圖(flow chart)或虛擬碼(pseudo code)來說明

03 流程圖與演算法

◀ ◇ ▶ ▷

3.1 流程圖符號

開始或結束
處理
判斷或比較
輸入或輸出
連接
流程方向

03 流程圖與演算法

◀ ◇ ▶ ▷

3.2 演算法

演算法是指使用有限的指令以解決某一特定問題的步驟。演算法有下列幾個特性：

- 1. 輸入(input):** 可以有零個或多個輸入資料。
- 2. 輸出(output):** 至少有一個輸出資料。
- 3. 明確性(definiteness):** 每個指令必須明確，不可模稜兩可。
- 4. 有限性(finiteness):** 演算法必須經過有限步驟後停止。
- 5. 有效性(effectiveness):** 可以在紙上作業追蹤其結果。

03 流程圖與演算法

◀ ◇ ▶ ▷

【例一】 輸入三個整數(變數名稱a、b及c)，求其平均值(變數名稱ave)。

【以文字敘述說明演算法】

- 輸入三個整數a、b及c。
- 總和sum為a、b及c之和。
- 平均avg為總和sum除於3之商。
- 輸出平均值ave。
- 結束。

【以演算法語言說明演算法】

```

1. input a, b, c
2. sum = a + b + c
3. avg = sum / 3
4. output avg
5. end
    
```

```

graph TD
    Start([開始]) --> Input[/輸入a,b,c/]
    Input --> Sum[sum←a+b+c]
    Sum --> Avg[avg←sum/3]
    Avg --> Output[/輸出avg/]
    Output --> End([結束])
    
```

03 流程圖與演算法

◀ ◇ ▶ ▷

【例二】 重複輸入一個整數a，若a為正數則累加起來，否則輸出總和，之後停止。

【以文字敘述說明演算法】

- 設總和為0。
- 輸入一個整數 a。
- 若 a 為正數則
- 將 a 加至總和。
- 輸入一個整數 a.
- 輸出總和。
- 結束。

【以演算法語言說明演算法】

```

1. sum = 0
2. input a
3. while a > 0
4.     sum = sum + a
5.     input a
6. output sum
7. end
    
```

指定運算子「=」可以將右邊的運算式值指定給左邊的變數。

03 流程圖與演算法

◀ ◁ ▶ ▷

指定運算子「=」

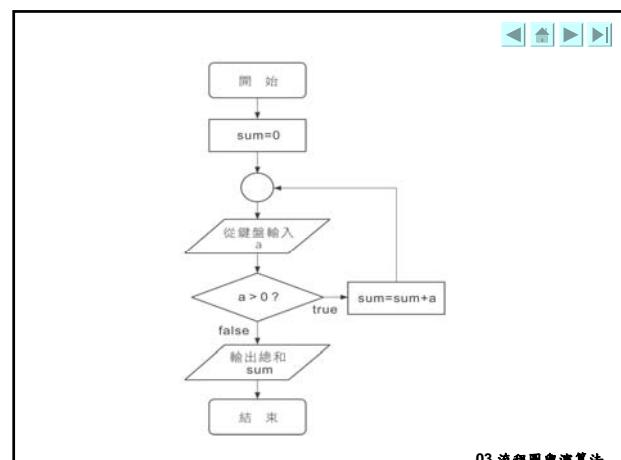
指定運算子「=」可以將右邊的運算式值指定給左邊的變數。

指定敘述的語法如下：

變數 = 運算式；

- 變數
變數名稱，為合乎規定的識別字。
- 運算式
為運算元(operand)與運算子(operator)及小括號的結合，運算結果只產生一個值，這個值指定給左邊的變數。

03 流程圖與演算法



◀ ◁ ▶ ▷

【例三】由 1 加至 100.

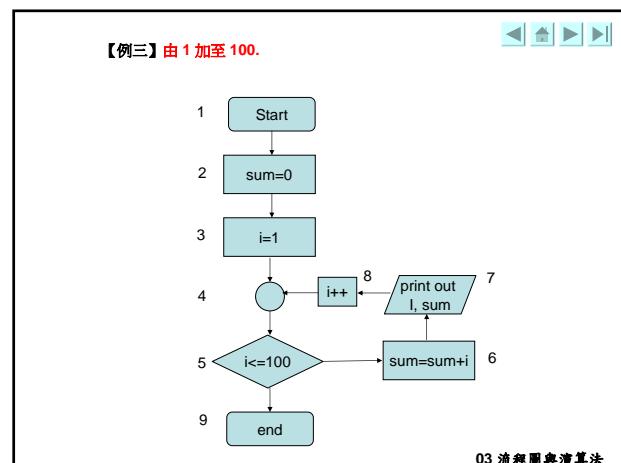
【以文字敘述說明演算法】

1. 設定二個整數*i*=1及*sum*=0.
2. *i* 小於或等於100繼續下一步，否則跳到步驟6
3. 將*i*加到總和*sum*.
4. 印出*i*及總和*sum*
5. *i*增加1，回到步驟2.
6. 結束.

【以演算法語言說明演算法】

1. set *i*=1, *sum*=0
2. While *i*<=100
3. *sum*= *sum*+*i*
4. output *i* and *sum*
5. Increase *i*
6. end

03 流程圖與演算法



◀ ◁ ▶ ▷

C code

```

// Filename: add1to100.cpp
// Purpose:由 1 加至 100.
// Authors: Ping-Sung Liao/A123456
// Date: 2008/03/01, Version: 1.00

#include <stdlib.h>
#include <stdio.h>
int main()
{
    int i, sum;
    sum=0;
    i=1;
    while (i<=100)
    {
        sum=sum+i; // 儲存
        i=i+1;
    }
    printf("%d\n", sum);
    return 0;
}
  
```

由 1 加至 100.
以下列方式寫出解答步驟

a. 文字敘述
b. 演算法語言
c. 流程圖

03 流程圖與演算法

◀ ◁ ▶ ▷

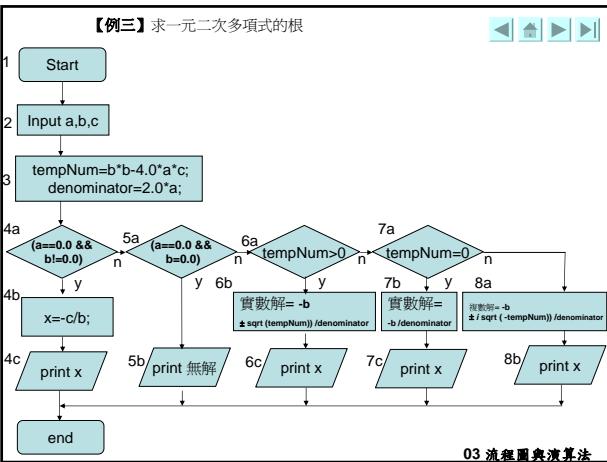
【例三】求一元二次多項式的根

【以文字敘述說明演算法】

```

//一元二次多項式的模型: ax^2+bx+c=0;
//輸入第一個整數a, b, c;
tempNumber=b*b-4.0*a*c;
denominator=2.0*a;
if (a==0.0 && b!=0.0)
    x=-c/b;
else if(a==0.0 && b==0.0)
    x無解;
else
    { if (tempNumber>0.0)
        兩個實數解為 -b ± sqrt (tempNumber) /denominator
        /denominator );
        else if (tempNumber==0.0)
        重根= -b/denominator
        else
        兩個複數解為
        -b/denominator ± i( sqrt (- tempNumber) /denominator )
    }
  
```

03 流程圖與演算法



03 流程图与清算法

```

// Filename: completeRootPolynomial.cpp
// Purpose: 求一元二次多项式的根
// Authors: XXXXX.Liao/A123456
// Date: 2010/03/12, 17:25, Version: 1.0
// 图形流程图的重要性，及文字编程型演算法描述
// 非常清楚地解决一元二次多项式的根(实根，虚根，复数根)
// (a,b,c) ∈ (1,0,2) / (1,A,10) / (2, 8, 10);
#include <stdlib.h>
#include <stdio.h>
#include <math.h> // necessary for sqrt() function

int main()
{
    float a, b, c;
    float tempNumber, numerator, denominator;
    printf("完整地解决一元二次多项式的根(实根，虚根，复数根)之求解演算法\n");
    printf("一元二次多项式的模型: ax^2+bx+c=0\n");
    printf("请输入第一个整数(a): "); // 提示使用者
    scanf("%f", &a); // 使用者输入资料
    printf("请输入第二个整数(b): "); // 提示使用者
    scanf("%f", &b); // 使用者输入资料
    printf("请输入第三个整数(c): ");
    scanf("%f", &c);
    tempNumber=b*b-4.0*a*c;
    denominator=2.0*a;
    printf("b=%f\n", b);
    printf("2*a=%f\n", denominator);
    printf("c=%f\n", c);
}

```

03 流程图与清算法

```

if(a==0.0 && b!=0.0)
printf("x=%f\n", -c/b);
else if(a==0.0 && b==0.0)
printf("无解\n");
else
{
    if(tempNumber>0.0)
        printf("两个实数根\n");
    printf("第一个根是 %f\n", (-b + sqrt(tempNumber)) / denominator); //输出
    printf("第二个根是 %f\n", (-b - sqrt(tempNumber)) / denominator); //输出
}
else if(tempNumber==0.0)
{
    printf("一个实数根\n");
    printf("第一个根是 %f\n", -b/denominator);
    printf("第二个根是 %f\n", -b/denominator);
}
else // (tempNumber<0.0)
{
    printf("两个虚数根\n");
    printf("第一个根是 %f + %f i\n", -b/denominator, sqrt(-tempNumber)/denominator ); //输出
    printf("第二个根是 %f - %f i\n", -b/denominator, -sqrt(-tempNumber)/denominator ); //输出
    printf("第二个根是 %f + %f i\n", -b/denominator, sqrt(-tempNumber)/denominator ); //输出
}
}
system("pause"); //程式暂停一下
return 0; //程式执行完毕，回到视窗作业系统
}

```

03 流程图与清算法

04 程式結構

- 4.1 循序結構
- 4.2 選擇結構
- 4.3 重複結構
- 4.4 運算式敘述
- 4.5 if 敘述
- 4.6 if ... else 敘述
- 4.7 switch & case 敘述
- 4.8 while 敘述
- 4.9 do/while 敘述
- 4.10 for 敘述

結構化程式設計的基本結構

- **循序結構**
- 依您撰寫程式的順序一個接一個的執行，稱為循序執行
- **選擇結構**
- 簡單控制轉向
- if/else, switch ... case (default, break)
- **重複結構**
- 帶有重複性的控制轉向
- for, while, do ... while, break, continue

04 程式結構

結構化程式設計優點

1. 架構清楚明白，
2. 容易除錯及修改，
3. 設計速度快速，
4. 設計費用相對的降低。

結構化程式設計

絕對要排除使用goto的敘述在程式裡跳來跳去，讓看程式的人眼花撩亂，增加程式維護的困難。

04 程式結構

4.1 循序結構

您撰寫程式的順序一個敘述接一個敘述執行

- 首先執行
sum=a+b+c;
- 然後執行
avg=sum/3;

04 程式結構

4.2 選擇結構

C語言提供三種選擇結構：if、if/else、switch。

```
if 敘述的語法
if (expression) statement;
或
if (expression)
{ statement 1;
  statement 2;
  statement i;
}
expression 表示條件運算式，statement 表示敘述。
```

04 程式結構

4.2 選擇結構 - if ... else

```
if ...else 敘述的語法
if (expression) statement 1;
else statement 2;
或
if (expression)
{ statement 1;
  statement i;
}
else
{ statement 2;
  statement k;
}
```

expression 表示條件運算式，statement / 表示敘述

04 程式結構

4.2 選擇結構 - switch (1)

```

switch (expression)
{ case 1: statement_1;
  break;
  case 2: statement_2;
  break;
  case 3: statement_3;
  break;
  ...
  case n: statement_n;
  break;
  default: statement;
}

```

04 程式結構

4.2 選擇結構 - switch (2)

```

switch (expression)
{ case 1: statement_1;
  case 2: statement_2;
  case 3: statement_3;
  ...
  case n: statement_n;
  break;
  default: statement;
}

```

04 程式結構

4.3 重複結構 - while

- C語言提供三種重複結構：while、do/while、for。

```

while (expression)
  statement;
或
while (expression)
{ statement 1;
  ...
  statement i;
}

```

04 程式結構

4.3 重複結構 - do/while

```

do
  statement
while (expression);

或

do
{ statement 1;
  ...
  statement i;
} while (expression);

```

04 程式結構

4.3 重複結構 - for

```

for (expression1; expression2; expression3)
  statement;
或
for (expression1; expression2; expression3)
{ statement 1;
  ...
  statement i;
}

```

04 程式結構

4.4.1 指定敘述

指定運算子「=」可以將右邊的運算式值指定給左邊的變數。

指定敘述的語法如下：

變數 = 運算式；

- 變數
變數名稱，為合乎規定的識別字。
- 運算式
為運算元（operand）與運算子（operator）及小括號的結合，運算結果只產生一個值，這個值指定給左邊的變數。

04 程式結構

◀ ◁ ▶ ▶|

等號為指定運算子中的一個運算子

符號	名稱	舉例 說明
=	簡單指定運算子	$y=x*2+3;$ 將右邊運算式計算後指定給左邊之變數 y 。
=	乘複合指定運算子	$y=x;$ 相當於 $y=y*x;$
/=	除複合指定運算子	$y/=2;$ 相當於 $y=y/2;$
%=	餘數複合指定運算子	$y\%=5;$ 相當於 $y=y\%5;$
+=	加複合指定運算子	$y+=x*4+6;$ 相當於 $y=y+x*4+6;$
-=	減複合指定運算子	$y-=x\%3+8;$ 相當於 $y=y-x\%3+8;$
<<=	左移複合指定運算子	$y<<2;$ 相當於 $y=y<<2;$
>>=	右移複合指定運算子	$y>>n;$ 相當於 $y=y>>n;$
&=	位元AND指定運算子	$i\&=j;$ 相當於 $i=i\&j;$
=	位元OR指定運算子	$i =j;$ 相當於 $i=i j;$
^=	位元XOR指定運算子	$i^=j;$ 相當於 $i=i\^j;$

04 程式結構

Logical Operators (1)

- **&&** (logical AND)
 - Returns true if both conditions are true
- **||** (logical OR)
 - Returns true if either of its conditions are true
- **!** (logical NOT, logical negation)
 - Reverses the truth/falsity of its condition
 - Unary operator, has one operand
- Useful as conditions in loops

Expression	Result
true && false	false
true false	true
!false	true

04 程式結構

Logical Operators (2)

expression1	expression2	expression1 && expression2
0	0	0
0	nonzero	0
nonzero	0	0
nonzero	nonzero	1

expression1	expression2	expression1 expression2
0	0	0
0	nonzero	1
nonzero	0	1
nonzero	nonzero	1

Fig. 4.13 Truth table for the && (logical AND) operator.

expression	expression
0	1
nonzero	0

Fig. 4.14 Truth table for the logical OR (||) operator.

expression	! expression
0	1
nonzero	0

Fig. 4.15 Truth table for operator ! (logical negation).

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Logical Operators (3)

Operators						Associativity	Type
++	--	+	-	!	(type)	right to left	unary
*	/	%				left to right	multiplicative
+	-					left to right	additive
<	<=	>	>=			left to right	relational
==	!=					left to right	equality
&&						left to right	logical AND
						left to right	logical OR
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment
,						left to right	comma

Fig. 4.16 Operator precedence and associativity.

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Confusing Equality (==) and Assignment (=) Operators (1)

• Dangerous error

- Does not ordinarily cause syntax errors
- Any expression that produces a value can be used in control structures
- Nonzero values are true, zero values are false
- Example using ==:


```
if ( payCode == 4 )
    printf( "You get a bonus!\n" );
```
- Checks payCode, if it is 4 then a bonus is awarded

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

Confusing Equality (==) and Assignment (=) Operators (2)

– Example, replacing == with =

```
if ( payCode = 4 )
printf( "You get a bonus!\n" );
```

- This sets payCode to 4
- 4 is nonzero, so expression is true, and bonus awarded no matter what the payCode was

– Logic error, not a syntax error

© Copyright 1992–2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

◀ ▶ ⟲ ⟳

4.4.2 遲增與遞減運算子

符號	名稱	舉例說明
++	遞增運算子	<code>++x;</code> x先增1後再與其他運算元計值，相當於 $x=x+1$;
		<code>x++;</code> x先與其他運算元計值後再增1，相當於 $x=x+1$;
--	遞減運算子	<code>--x;</code> x先減1後再與其他運算元計值，相當於 $x=x-1$;
		<code>x--;</code> x先與其他運算元計值後再減1，相當於 $x=x-1$;

04 程式結構

◀ ▶ ⟲ ⟳

4.5 if 敘述

問題描述：我想從電腦的鍵盤輸入張三的成績若張三的成績大於或等於 60 分則請電腦顯示 "及格" 兩個字

程式虛擬碼如下：

```
input grade
if grade >= 60
    output "及格"
```

流程圖如右圖所示

04 程式結構

◀ ▶ ⟲ ⟳

【程式ifgrade.c】

```
***** ifgrade.c *****
#include <stdio.h>
int main()
{
    int grade;
    printf("輸入成績 grade: ");      /*步驟2 提示語*/
    scanf("%d", &grade);             /*步驟2 讀取資料*/
    if (grade >= 60) printf("及格"); /*步驟3及4 執行及輸出*/
    return 0;
}
```

【執行】

```
gcc iftest.c -o ifgrade.exe <Enter>
ifgrade.exe <Enter>
輸入成績 grade: 55 <Enter>
ifgrade.exe <Enter>
輸入成績 grade: 66 <Enter>
及格
```

04 程式結構

◀ ▶ ⟲ ⟳

問題描述：我想從電腦的鍵盤輸入一個整數若該整數為偶數，則請電腦顯示 "偶數" 兩個字

程式虛擬碼如下：

```
input number
even = number % 2
if even == 0
    output "偶數"
```

流程圖如右圖所示

04 程式結構

◀ ▶ ⟲ ⟳

【程式ifeven.c】

```
***** ifeven.c *****
#include <stdio.h>
int main()
{
    int number;
    int even;
    printf("輸入整數 number: ");   /*步驟2*/
    scanf("%d", &number);          /*步驟2*/
    even = number % 2;            /*步驟3*/
    if (even == 0) printf("偶數"); /*步驟4及5*/
    return 0;
}
```

【執行】

```
gcc ifeven.c -o ifeven.exe <Enter>
ifeven.exe <Enter>
輸入整數 number: 5 <Enter>
ifeven.exe <Enter>
輸入整數 number: 6 <Enter>
偶數
```

04 程式結構

◀ ▶ ⟲ ⟳

4.6 if/else 敘述

問題描述：拿張三的成績來講，成績若超過(含) 60 分，就輸出 "及格" 兩個字，否則的話並沒有處理。現在加上否則就輸出 "不及格" 三個字。

程式碼虛擬如下：

```
if (grade >= 60)
    printf("及格");
else
    printf("不及格");
```

流程圖如右圖所示

04 程式結構

【程式ifpass.c】

```
***** ifpass.c *****/
#include <stdio.h>
int main()
{
    int grade;
    printf("輸入成績 grade: "); /*步驟2*/
    scanf("%d", &grade); /*步驟2*/
    if (grade >= 60) /*步驟3*/
        printf("及格"); /*步驟4*/
    else
        printf("不及格"); /*步驟5*/
    return 0;
}
```

04 程式結構

4.7 switch & case 敘述

- switch選擇結構讓您可在多種情況中選一種，
- 例如您可從鍵盤輸入一個大寫的英文字母到電腦記憶體中的ch位置，那麼ch位置的內含值就有26種之多，但是您想將這26種分為三組，第一組只包括A，第二組只包括B，第三組只包括其他的字母，這種結構表示如下：

```
switch (ch)
{ case 'A':
    /*插入第一組所要執行的動作*/;
    break;
case 'B':
    /*插入第二組所要執行的動作*/;
    break;
default :/*第三組所要執行的動作*/;
}
```

04 程式結構

- switch敘述的區塊中出現兩個標記（label）
– case標記敘述以及一個default標記敘述。

case標記敘述的寫法如下：

case 常數：敘述;

- case後面的敘述可以由若干個敘述組成，最後通常以「break;」結束，跳出switch敘述，緊接著執行switch的下一個敘述。若不以break結束case敘述則會繼續執行下一個case或default敘述，就失去多選一的意義了。

04 程式結構

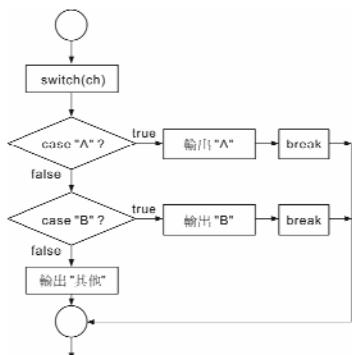
defalut標記敘述的寫法如下：

default :敘述;

- default後面的敘述可以由若干個敘述組成，但最後不須以break結束，因為它已經是switch的最後一個敘述了，不過您加上break也沒錯。

04 程式結構

switch (ch) 的流程圖如下所示。



04 程式結構

【程式】

```
***** charswitch.c *****/
#include <stdio.h>
int main()
{
    int ch;
    printf("輸入一個大寫英文字母 ch: ");
    scanf("%c", &ch);
    switch (ch)
    {
        case 'A': printf("A"); break;
        case 'B': printf("B"); break;
        default: printf("其他");
    }
    return 0;
}
```

04 程式結構

【程式gradeswitch.c】

```
***** gradeswitch.c *****/
#include <stdio.h>
int main()
{
    int grade;
    printf("輸入成績 grade: ");
    scanf("%d", &grade);
    switch (grade/10)
    {
        case 10:
        case 9: printf("甲"); break;
        case 8: printf("乙"); break;
        case 7: printf("丙"); break;
        case 6: printf("丁"); break;
        default: printf("戊");
    }
    return 0;
}
```

【執行】

```
gcc gradeswitch.c -o gradeswitch.exe <Enter>
gradeswitch.exe <Enter>
輸入成績 grade: 99 <Enter>
甲
gradeswitch.exe <Enter>
輸入成績 grade: 77 <Enter>
丙
gradeswitch.exe <Enter>
輸入成績 grade: 33 <Enter>
戊
```

04 程式結構

4.8 while 敘述

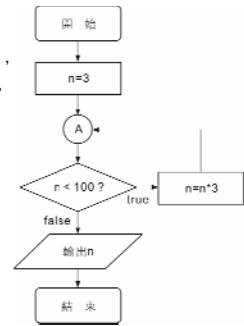
while重複結構

首先測試條件運算式expression的值，若為真則執行其後的statement敘述，執行完後，重複測試條件運算式

語法如下：

```
while (expression)
    statement;
```

- 一般statement常使用複合敘述。



04 程式結構

【程式triple.c】

```
***** triple.c *****/
#include <stdio.h>
int main()
{
    int n;
    n = 3;
    while (n < 100)
    {
        n = n * 3;
    }
    printf("%d\n", n);
    return 0;
}
```

【執行】

```
gcc triple.c -o triple.exe <Enter>
triple.exe <Enter>
243
```

04 程式結構

4.9 do/while 敘述

• do/while敘述與while敘述唯一不同之處

- while先判斷條件運算式 expression之值是否為真，再決定是否要執行statement敘述，
- do/while敘述剛好相反，它先執行statement敘述後再判斷條件運算式expression之值是否為真，再決定是否重複執行statement敘述。

- while後的statement敘述可能一次都沒有執行（開始時條件運算式expression之值為假），但do/while後的statement敘述最少要執行一次。
- do/while敘述一樣要注意無窮迴圈的問題。

04 程式結構

- do/while敘述的語法如下：

```
do
    statement
while (expression);
```

- 一般statement常使用複合敘述。

```

graph TD
    Start([開始]) --> SetN1[n=1]
    SetN1 --> Cond{n <= 10 ?}
    Cond -- true --> OutputN[/輸出n/]
    OutputN --> IncN1[n = n + 1]
    IncN1 --> Cond
    Cond -- false --> End([結束])
    
```

04 程式結構

【程式increment.c】

```
***** increment.c *****/
#include <stdio.h>
int main()
{
    int n;
    n = 1;
    do
    {
        printf("%d ", n);
        n = n + 1;
    } while (n <= 3);
    return 0;
}
```

【執行】

```
gcc increment.c -o increment.exe <Enter>
increment.exe <Enter>
1 2 3
```

04 程式結構

6

◀ ◁ ▶ ▷

4.10 for 敘述

- for敘述是最常用到的重複敘述，一般稱為for迴圈。
- for敘述語法如下：

```

for (expression1; expression2; expression3)
    statement;

```

運算式expression1為for迴圈控制變數的初值，
 運算式expression3為for迴圈控制變數的修正值，
 運算式expression2是for迴圈是否重複執行的條件運算式，若條件運算式的值為真則迴圈重複執行敘述statement，若為假值則終止for迴圈之執行。

04 程式結構

◀ ◁ ▶ ▷

• 以sum.c程式為例

```

graph TD
    1[開始] --> 2[sum=0]
    2 --> 3[i=1]
    3 --> 4(( ))
    4 --> 5{i<=3?}
    5 -- true --> 6[sum=sum+i]
    6 --> 7[輸出i,sum]
    7 --> 8[i++]
    8 --> 5
    5 -- false --> 9[結束]

```

04 程式結構

◀ ◁ ▶ ▷

【程式sum.c】

```

***** sum.c *****/
#include <stdio.h>
int main()
{ int i, sum;
    sum = 0;
    for (i=1; i<=100; i++)
    { sum = sum + i;
        printf("i=%d sum=%d\n", i, sum);
    }
    return 0;
}

【執行】
gcc sum.c -o sum.exe <Enter>
sum.exe <Enter>
i=1 sum=1
i=2 sum=3
i=3 sum=6

```

04 程式結構

◀ ◁ ▶ ▷

```

#include <cslib.h>
#include <stdio.h>
#include <iostream.h>
4
5 int main()
6 { int i,j;
7     for (i = 1 ; i<=9; i++)
8         for (j = 1 ; j<=9; j++)
9             cout<< i << " " << j << " " << i*j << " ";
10    cout <<"\n";
11
12    for (i = 1 ; i<=9; i++)
13        for (j = 1 ; j<=9; j++)
14            cout<< i << " " << j << " " << i*j << " ";
15    cout <<"\n";
16    }
17    cout <<"\n";
18    cout<<"bye! bye!";
19
20    system("pause");
21    return 0;

```

04 程式結構

◀ ◁ ▶ ▷

應用範例

- 因數判斷(if)
- 質數判斷(if, for)
- 質因數列舉，質因數判斷(if, for for ...)
- 質因數列舉，質因數分解(if, for for ...)
- DEV C++最大質因數探索(研究)

163.27.149.1/information/96%20teaching%20community/math/6/6-4.doc

04 程式結構

05 流程控制(二)

- [5.1 關係運算子與邏輯運算子](#)
- [5.2 關係運算子與運算式](#)
- [5.3 邏輯運算子與運算式](#)
- [5.4 複合敘述](#)
- [5.5 空白敘述](#)
- [5.6 break及continue敘述](#)
- [5.7 逗號運算子](#)
- [5.8 條件運算子](#)

◀ ◆ ▶ ▷

- C語言的七種基本結構
 - 1種循序的結構
 - 3種選擇的結構
 - 3種重複的結構。

其中，選擇與重複的結構都會改變程式執行的流程

- break及continue敘述會改變程式執行的流程。
- 關係運算子與邏輯運算子

構成條件運算式的運算子，而條件運算式值的真與假決定程式流程的走向。

2
05 流程控制(二)

◀ ◆ ▶ ▷

5.1 關係運算子與邏輯運算子(1)

關係運算子表

運算子	名稱	說明
<	小於	左運算元值若小於右運算元，則其結果為真。
>	大於	左運算元值若大於右運算元，則其結果為真。
<=	小於等於	左運算元值若小於等於右運算元，則其結果為真。
>=	大於等於	左運算元值若大於等於右運算元，則其結果為真。
==	相等	左運算元值若相等右運算元，則其結果為真。
!=	不相等	左運算元值若不相等右運算元，則其結果為真。

3
05 流程控制(二)

◀ ◆ ▶ ▷

5.1 關係運算子與邏輯運算子(2)

邏輯運算子表

運算子	名稱	說明
&&	邏輯 AND	若兩運算元值均為真（非0），結果為1，否則為0。
	邏輯 OR	若兩運算元值均為0，結果為0，否則為1。
!	邏輯 NOT	產生邏輯值0或1。若運算元之值非0則產生0。若運算元之值為0則產生1。

4
05 流程控制(二)

◀ ◆ ▶ ▷

運算子優先順序表

優先順序	運算子	結合
1	() [] > -	左至右
2	! ~ ++ -- (運算元) * & sizeof	右至左
3	* / %	左至右
4	+ -	左至右
5	<< >>	左至右
6	< <= > >=	左至右
7	== !=	左至右
8	&	左至右
9	^	左至右
10		左至右
11	&&	左至右
12		左至右
13	? :	右至左
14	= += -= *= /= %= <<= >>= &= = ^=	右至左
15	,	左至右

5
05 流程控制(二)

◀ ◆ ▶ ▷

5.2 關係運算子與運算式

- 六個關係運算子均為**二元運算子**
- 六個關係運算子**執行結果**
 - 關係運算式為真，值為整數值1，
 - 關係運算式為假，值為整數值0。

範例	a-b	a<b	a<=b	a==b	a!=b	a>=b	a>b
a=5,b=3	正數	0	0	1	1	1	
a=3,b=3	零	0	1	0	1	0	
a=3,b=5	負數	1	1	0	1	0	

6
05 流程控制(二)

```

***** reexpr.c *****/
#include <stdio.h>
int main()
{
    int a, b;
    printf("輸入兩個整數 a, b: ");
    scanf("%d%d", &a,&b);
    printf("a=%d b=%d\n", a,b);
    printf("a<b 值為 %d\n", (a<b));
    printf("a<=b 值為 %d\n", (a<=b));
    printf("a==b 值為 %d\n", (a==b));
    printf("a!=b 值為 %d\n", (a!=b));
    printf("a>b 值為 %d\n", (a>b));
    printf("a>=b 值為 %d\n", (a>=b));
    return 0;
}

```

7
05 流程控制(二)

8
05 流程控制(二)

5.3 邏輯運算子與運算式(1)

- 邏輯運算子!為一元運算子，
- &&及||為二元運算子，
- !、&&及||作用於運算式結果為true (整數值1) 或 false (整數值0) 。

a	b	a&&b	a b
0	0	0	0
0	非零整數	0	1
非零整數	0	0	1
非零整數	非零整數	1	1

9
05 流程控制(二)

5.3 邏輯運算子與運算式(2)

a	b	a && b	a b
F	F	F	F
F	T	F	T
T	F	F	T
T	T	T	T

10
05 流程控制(二)

```

***** logexpr.c *****/
#include <stdio.h>
int main()
{
    int a, b;
    printf("輸入兩個整數 a, b: ");
    scanf("%d%d", &a,&b);
    printf("a=%d b=%d\n", a,b);
    printf("!a 值為 %d\n", (!a));
    printf("!b 值為 %d\n", (!b));
    printf("a&&b 值為 %d\n", (a&&b));
    printf("a||b 值為 %d\n", (a||b));
    return 0;
}

```

11
05 流程控制(二)

5.4 複合敘述

- 複合敘述由宣告及敘述組成
- 包含於一對大括號{}裡面。從左大括號開始，其後跟著敘述，最後以大括號結束。

```

/*複合敘述開始*/
int a, b, c;
a = 1;
b = 2;
c = a + b;
/*複合敘述結束*/

```

12
05 流程控制(二)

if and composite statements

```

printf("輸入兩個整數 a,b : ");
scanf("%d%d", &a, &b);
if (a > b)
{
    c = a;
    printf("a>b");
}
else
{
    c = b;
    printf("a<=b");
}

```

◀ ◁ ▶ ▷

5.5 空白敘述

- 空白敘述只包含一個分號「;」。表示沒有動作。例如：

```

if (a > b)
    ;
else
    printf("a<=b");

```

[說明] $a > b$ 時沒有動作。
只有在 $a \leq b$ 時才輸出 "a<=b" 字串。

13
05 流程控制(二)

◀ ◁ ▶ ▷

5.6 break及continue敘述

- break及continue敘述均會終止正常的控制流程。
- break敘述終止目前的迴圈，執行目前迴圈的下一個敘述。
- continue敘述終止目前這一回，而執行本迴圈的下一回。

注意

- 一個迴圈通常要重複執行好幾回的，break敘述終止目前的迴圈。
- 一個迴圈通常要重複執行好幾回的，但continue只是終止目前這一回而已。

14
05 流程控制(二)

◀ ◁ ▶ ▷

```

***** negbreak.c *****/
#include <stdio.h>
int main()
{
    int true = 1;
    float x;
    while (true)
    {
        printf("輸入浮點數 x : ");
        scanf("%f", &x);
        if (x < 0.0)
            break;
        printf("平方=%10.6f\n", x*x);
    }
    return 0;
}

```

gcc negbreak.c -o negbreak.exe ↵
negbreak.exe ↵

輸入浮點數 x : 2.0 ↵
平方= 4.000000
輸入浮點數 x : 2.345 ↵
平方= 5.499025
輸入浮點數 x : -1.0 ↵

15
05 流程控制(二)

◀ ◁ ▶ ▷

```

***** poscont.c *****/
#include <stdio.h>
int main()
{
    int true = 1;
    float x;
    while (true)
    {
        printf("輸入浮點數 x : ");
        scanf("%f", &x);
        if (x > 10.0)
            continue;
        if (x < 0.0)
            break;
        printf("平方=%10.6f\n", x*x);
    }
    return 0;
}

```

gcc poscont.c -o poscont.exe ↵
poscont.exe ↵

輸入浮點數 x : 3.0 ↵
平方= 9.000000
輸入浮點數 x : 33.0 ↵
輸入浮點數 x : 2.0 ↵
平方= 4.000000
輸入浮點數 x : -1.0 ↵

16
05 流程控制(二)

◀ ◁ ▶ ▷

5.7 逗號運算子(1)

- 在C語言所有運算子當中，逗號運算子的優先順序最低，它是二元運算子，其運算元為運算式，由左到右計算。
- 逗號運算子的格式如下：

```

expression1 , expression2

```

expression1運算式先執行，
然後再執行expression2運算式，
逗號運算子的值為最後一個運算式的值。

```

float a;
int b, c;
c = a = 1.0, b = 2;

```

Confuse , 少用為妙

17
05 流程控制(二)

◀ ◁ ▶ ▷

5.7 逗號運算子(2)

```

***** sum.c *****/
#include <stdio.h>
int main()
{
    int i, sum;

    for (sum = 0, i=1 ; i<=100; i++)
    {
        sum = sum + i;
        printf("i=%d sum=%d\n", i, sum);
    }
    system("pause");
    return 0;
}

```

18
05 流程控制(二)

5.8 條件運算子

- 條件運算子「?:」為三元運算子，其格式如下：

運算元1 ? 運算元2 : 運算元3

- 若「運算元1」為非零值，則條件運算子執行的結果為「運算元2」之值，否則為「運算元3」之值，如下例：

```
int i=-5, absi;  
absi = i<0 ? -i : i;
```

- 「 $i < 0$ 」為真值，故 $absi = -i = -(-5) = 5$ 。

19

05 流程控制(二)

【程式condexpr.c】

```
***** condexpr.c *****  
#include <stdio.h>  
int main()  
{  
    int i=-5, absi, k;  
    float a;  
    absi = i<0 ? -i : i;  
    a = i<0 ? -1.0 : 1;  
    k = i>0 ? -1.0 : 1;  
    printf("i=%d absi=%d k=%d a=%f\n", i,absi,k,a);  
    return 0;  
}
```

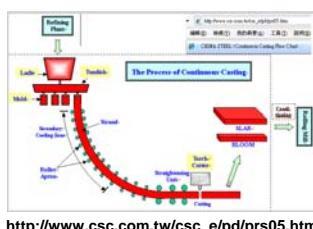
【執行】

```
gcc condexpr.c -o condexpr.exe <Enter>  
condexpr.exe <Enter>  
i=-5 absi=5 k=1 a=-1.000000
```

20

05 流程控制(二)

06 函式



- 6.1 何謂函式
- 6.2 函式宣告
- 6.3 函式定義
- 6.4 函式呼叫
- 6.5 數學函式庫
- 6.6 隨機數產生器
- 6.7 儲存類別
- 6.8 範圍
- 6.9 遞迴

- 在C語言中的模組（module）稱為函式（function），它沒有數學上函數那麼單純，但在C語言程式設計上很重要。

- 如何設計自己的函式：函式的宣告（declaration）、函式的定義（definition）、函式的呼叫（call）。
- main() 函式是C語言的唯一且第一個執行之函式。
- 函式的呼叫（call）：called by value, called by address (including called by pointer and called by reference.)

06 函式

6.1 何謂函式

- 函式乃是為完成特定工作的獨立模組。
- 函式的架構：

```
//函式宣告, appeared in the front of a program  
int main()  
{  
    ...  
    //函式呼叫  
    ...  
}  
//函式定義
```

06 函式

```
***** square.c *****/  
#include <stdio.h>  
int square(int); /*函式宣告*/  
  
int main()  
{ int s;  
    s = square(3); /*函式呼叫*/  
    printf("3*3=%d\n", s);  
    return 0;  
}  
int square(int a) /*函式定義*/  
{ int ret;  
    ret = a * a;  
    return (ret);  
}
```

```
***** square.c *****/  
#include <stdio.h>  
int square(int); /*函式宣告*/  
int square(int a) /*函式定義*/  
{ int ret;  
    ret = a * a;  
    return (ret);  
}  
  
/*函式定義」寫在main()函式之前  
int main()  
{ int s;  
    s = square(3); /*函式呼叫*/  
    printf("3*3=%d\n", s);  
    return 0;  
}
```

【執行】
square.exe <Enter>
3*3=9

06 函式

6.2 函式宣告

- 函式宣告的格式如下：

傳回值資料型態 函式名稱 (參數資料型態);
• 如下例：

int square (int);
傳回值資料型態 函式名稱 參數資料型態 分號結束

- 「傳回值型態」及「參數型態」可為C語言任何合乎語法的資料型態，如字元型態的char、整數型態的int、浮點數型態的float等等，若沒有傳回值型態則以關鍵字void表示之。

06 函式

6.3 函式定義

```
傳回值資料型態 函式名稱 ( 參數資料型態 參數名稱 )  
{  
    宣告  
    ...  
    敘述  
    ...  
}
```

- 函式定義的第一列除了最後的符號；，與函式宣告相同
- 若沒有參數則以void表示。「參數名稱」稱為形式參數，或簡稱為參數。執行時參數的值將被「函式呼叫」時的引數值所取代。

06 函式

6.4 函式呼叫

函式名稱(引數);

或

變數 = 函式名稱(引數);

引數為實際的值，引數間以逗號「，」隔開。

```
s = square( 3 );      /*square()函式呼叫*/
s2 = sum ( 2, 4 );   /*sum()函式呼叫*/
```

06 函式

```
***** callfunc.c *****/
#include <stdio.h>
int square(int a)
{ a = a * a;
  return a;
}
int main()
{ int i=3, s;
  printf("呼叫|square(i)前 i=%d\n", i);
  s = square(i);
  printf("呼叫|square(i)後 i=%d s=%d\n", i,s);
  return 0;
}
```

【執行】
callfunc.exe ←
呼叫|square(i)前 i=3
呼叫|square(i)後 i=3 s=9

06 函式

6.5 數學函式庫

- 由於C語言提供了前置處理器藉著「#include」指引指令可將一些事先設計好的常數、函式，儲存於函式庫，使用時才將它引入使用者的應用程式中，一起編譯、連結、執行，大大的提高程式設計的生產能力。
- 數學函式庫math.h表頭檔讓使用者輕易地執行某些常用的數學運算。

06 函式

Standard C Math

Display all entries for Standard C Math on one page, or view entries individually:

abs	absolute value
acos	arc cosine
asin	arc sine
atan	arc tangent
atan2	arc tangent, using signs to determine quadrants
ceil	the smallest integer not less than a certain value
cosh	hyperbolic cosine
div	returns the quotient and remainder of a division
exp	returns "e" raised to a given power
fabs	absolute value for floating-point numbers
floor	returns the largest integer not greater than a given value
fmod	returns the remainder of a division
frexp	decomposes a number into scientific notation
labs	absolute value for long integers
ldexp	computes a number in scientific notation
ld2e	returns the quotient and remainder of a division, in long integer form
log	natural logarithm (to base e)
log10	common logarithm (to base 10)

http://www.cppreference.com/stdmath/index.html

06 函式

```
***** fabs.c *****/
#include <stdio.h>
#include <math.h>
int main()
{ double x;
  printf("輸入一個倍精確浮點數 x: ");
  scanf("%lf", &x);
  printf("fabs(%lf)=%lf\n", x, fabs(x));
  return 0;
}
```

【執行】
fabs.exe ←
輸入一個倍精確浮點數 x: -123.0 ←
fabs(-123.000000)=123.000000
fabs.exe ←
輸入一個倍精確浮點數 x: 246.8 ←
fabs(246.800000)=246.800000

06 函式

```
***** trifunc.c *****/
#include <stdio.h>
#include <math.h>
int main()
{ double ang, x;
  printf("輸入一個倍精確浮點數角度 ang: ");
  scanf("%lf", &ang);
  x = M_PI * ang / 180.0;
  printf("cos(%lf)=%lf\n", x, cos(x));
  printf("sin(%lf)=%lf\n", x, sin(x));
  printf("tan(%lf)=%lf\n", x, tan(x));
  return 0;
}
```

【執行】
trifunc.exe ←
輸入一個倍精確浮點數角度 ang: 60.0 ←
cos(1.047198)=0.500000
sin(1.047198)=0.866025
tan(1.047198)=1.732051

06 函式

```

***** squareroot.c *****/
#include <stdio.h>
#include <math.h>
int main()
{ double x;
    printf("輸入一個倍精確浮點數 x: ");
    scanf("%lf", &x);
    printf("sqrt(%lf)=%lf\n", x, sqrt(x));
    return 0;
}

```

【執行】

squareroot.exe ↵
輸入一個倍精確浮點數 x: 2.0 ↵
sqrt(2.000000)=1.414214

06 函式

```

***** ceilfloor.c *****/
#include <stdio.h>
#include <math.h>
int main()
{ double x;
    printf("輸入一個倍精確浮點數 x: ");
    scanf("%lf", &x);
    printf("ceil(%lf)=%lf\n", x, ceil(x));
    printf("floor(%lf)=%lf\n", x, floor(x));
    return 0;
}

```

ceilfloor.exe <Enter>
輸入一個倍精確浮點數 x: -16.3 ↵
ceil(-16.300000)=-16.000000
floor(-16.300000)=-17.000000

06 函式

6.6 隨機數產生器

- 在科學研究及遊戲的領域，經常需要做適當的模擬，才能分析問題，找出解決問題的方案。
- 在C語言提供模擬作業，通常使用隨機數產生器兩個函式：rand() 及 srand()。
- rand() 函式傳回一個介於0（含）與RAND_MAX（含）之間的整數。RAND_MAX為定義於stdlib.h表頭檔裡的符號常數，隨著C編譯器的不同而異。
- srand() 函式可以讓您改變隨機種子而取得不同系列的隨機數。

06 函式

```

***** diepoint.c *****/
#include <stdio.h>
#include <stdlib.h>
int main()
{ int i;
    printf("RAND_MAX=%d\n", RAND_MAX);
    for (i=1; i<=10; i++)
    { printf("%5d", (rand()%6)+1);
        if (i%5==0) printf("\n");
    }
    system("pause");
    return 0;
}

```

c:\ DAI_2008_962CSU\2008_962C\diepoint.exe
RAND_MAX=32767
6 6 5 5 6
5 1 1 5 3
請按任意鍵繼續 . . .

06 函式

```

***** randseed.c *****/
#include <stdio.h>
#include <stdlib.h>
int main()
{ int i;
    unsigned seed;
    printf("輸入一個隨機數產生器種子 seed: ");
    scanf("%u", &seed);
    srand(seed);
    for (i=1; i<=10; i++)
    { printf("%5d", (rand()%6)+1);
        if (i%5==0) printf("\n");
    }
    return 0;
}

```

ex DAI_2008_962CSU\2008_962C\randseed.exe
輸入一個隨機數產生器種子 seed: 7
2 5 6 3 6
1 3 5 2 5
請按任意鍵繼續 . . .

ex DAI_2008_962CSU\2008_962C\randseed.exe
輸入一個隨機數產生器種子 seed: 123
3 4 6 1 6
2 6 3 5 3
請按任意鍵繼續 . . .

06 函式

6.7 儲存類別

- 識別字當變數的名稱使用時**，識別字的屬性包括名稱、型態及內含值。
- 識別字用為函式的名稱使用時**，在程式中的識別字除了名稱、型態、及內含值之外還包括儲存類別、儲存期間、範圍及連結等屬性。
- C語言提供四種儲存類別：**

auto
register
extern
static

06 函式

```

/************* duration.c *****/
#include <stdio.h>
int total;
int main()
{ int a, b, i;
for (i=1; i<=2; i++)
{
    printf("請輸入兩個區域整數變數 a,b: ");
    scanf("%d %d", &a, &b);
    total = total + sum(a,b);
}
printf("全域變數 total=%d\n", total);
return 0;
}
int sum (int a, int b)
{
static int c;
c = c + a + b;
printf("區域變數宣告為static者 c=%d\n", c);
return (a+b);
}

```

06 函式

6.8 範圍

- 識別字的範圍（scope）：指可被參考到的程式部份，例如在函式中宣告的區域變數可被參考到的程式部份只限於該函式而已。
- 識別字有四種範圍：函式範圍、檔案範圍、區塊範圍及函式原型範圍。**
- 標記只適用於函式範圍，在函式裡標記可出現於任何位置，但出了該函式就消失不見了。**標記通常用於switch敘述中的case標記敘述。**

06 函式

範圍(續)

- 識別字宣告在函式外面的屬於檔案範圍，這種識別字從宣告位置之後的所有函式均可使用，像**全域變數**、函式定義及函式原型均屬於**檔案範圍**。
- 識別字宣告於區塊之內屬於區塊範圍「{ ... }」，區塊範圍到區塊結束符號「}」出現時就結束了。宣告於區塊內的變數稱之為**區域變數**。
- 唯一屬於函式原型範圍的識別字是參數列中的識別字，但**函式原型的參數列只須列出型態就可以了**，標示參數名稱也未嘗不可，不過參數名稱會被忽略。

06 函式

明確宣告變數等級

- 隱含式的變數宣告：編譯器將自動指定內定的變數等級給該變數，例如在函式內使用隱含式變數宣告的變數等級內定為auto。
- 在宣告變數時，明確宣告變數等級的詳細語法：

變數等級 變數資料型態 變數名稱:

範例：
static int a;
extern
double b;
auto char c;
register d;



陳錦輝, C/C++初學指引

static 區域變數等級

- static 區域變數等級的生命週期**
 - 將由宣告後直到整個程式執行完畢。
 - 【註】：auto區域變數是用堆疊的方式來儲存變數，所以函式執行結束後，auto區域變數就跟著被釋放。而static區域變數則使用固定的位址來存放變數，**所以要等到整個程式執行完畢，static區域變數才會消失**。
- static 區域變數等級的視野**
 - static區域變數視野與auto區域變數相同，也就是僅限於宣告該變數的區段。

陳錦輝, C/C++初學指引, 金禾書局(R501)

- 執行結果：

```

var1 = 101
=====
var1 = 102

```

- 範例說明：

- 一般使用情況
- 計次
- 狀態的設定及查詢

```

/*檔名:ch9_09.cpp
功能:static區域變數*/
#include <iostream>
#include <stdlib.h>
using namespace std;
void func1(void)
{ // cout << "var1 = " << var1 << endl;
{ static int var1=100;
var1=var1+1;
cout << "var1 = " << var1 << endl;
}
// cout << "var1 = " << var1 << endl;
}
int main(void)
{ func1();
cout << "======" << endl;
func1();
system("pause");
return 0;
}

```

陳錦輝, C/C++初學指引, 金禾書局(R501)

◀ ▶ ⟲ ⟳

C語言基本四種變數等級

變數等級	生命週期	視野
auto變數 (普通內在變數)	從函式(或區段)執行開始，直到函式(或區段)執行結束。	僅限於區段之內。 (區段視野)
static auto變數 (static內在變數)	直到程式結束為止。	僅限於區段之內。 (區段視野)
extern變數 (普通外在變數)	從程式執行到程式結束為止。	跨檔案的所有函式。 (檔案視野)
static extern變數 (static外在變數)	從程式執行到程式結束為止。	同一檔案內的所有函式。 (檔案視野)

陳錦輝, C/C++初學指引, 金禾書局(R501)

◀ ▶ ⟲ ⟳

6.9 遞迴

- 一般函式通常都呼叫別的函式；
- 遞迴函式一定會呼叫函式本身，稱為遞迴 (recursive) 呼叫。
- 遞迴呼叫的必備機制
 - 在遞迴呼叫的函式中必須有結束條件設計(停止遞迴的機制)，否則會無窮無盡的遞迴，
 - 未到達結束條件之前，一定會呼叫函式本身。

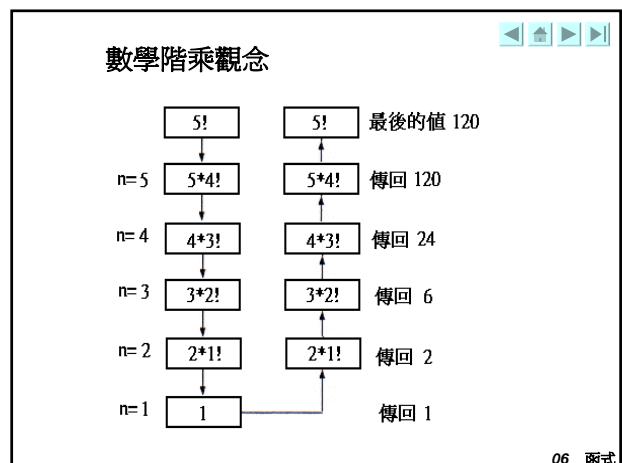
06 函式

◀ ▶ ⟲ ⟳

```
***** recur.c *****/
#include <stdio.h>
void msg(int n)
{
    if (n>0)
        { printf("您好!!\n");
          msg(n-1);
        }
}
int main()
{
    int n;
    printf("您要重複顯示信息幾次 n: ");
    scanf("%d", &n);
    msg(n);
    return 0;
}
```

recur.exe ↴
您要重複顯示信息幾次 n: 3 ↴
您好!!
您好!!
您好!!

06 函式



◀ ▶ ⟲ ⟳

```
***** factorial.c *****/
#include <stdio.h>
long factorial(int n)
{
    if (n<=1)
        return 1;
    else
        return (n* factorial(n-1));
}
int main()
{
    int i;
    for (i=1; i<=5; i++)
        printf("%d! = %ld\n", i, factorial(i));
    return 0;
}
```

factorial.exe ↴
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120

06 函式

07 陣列

- 7.1 陣列
- 7.2 宣告陣列
- 7.3 陣列長度
- 7.4 陣列範例
- 7.5 陣列與函式呼叫
- 7.6 陣列與排序
- 7.7 陣列與搜尋法
- 7.8 多維陣列



7.1 陣列

- 陣列是在電腦記憶體裡一塊連續的記憶體，用於儲存同一型態的若干個資料。
- 每一個陣列資料均使用同一個名稱(亦即**陣列名稱**)。及**位置索引值**來參考陣列中某一個位置或元素。

• 陣列的概念包括

陣列的宣告、初值的設定、陣列使用

07 陣列

陣列（續）

- 注意陣列的位置索引，又稱為**註標**，是從**0**算起的。**註標必須是整數或整數運算式**，
- n個元素的陣列其註標是從第0個至第(n-1)個。
- int k[8]={64, 38, 16, 49, 91, 73, 82, 27};
為整數陣列k，包含8個元素，第0個元素表示為k[0]，第1個元素表示為k[1]，第7個元素表示為k[7]等等。

k	64	38	16	49	91	73	82	27
	k[0]	k[1]	k[2]	k[3]	k[4]	k[5]	k[6]	k[7]

07 陣列



陣列（續）

- 陣列是由一群具有相同資料型態的元素所組成的，利用迴圈並配合陣列的註標，可輕易的處理大量的資料。
- 陣列及結構在資料結構中佔很重要的地位。
- 結構(struct)中的元素可由不同型態的資料組成。
- 陣列與未含指標成員的struct結構體均屬於**靜態結構**。
- 含指標成員的struct結構(串列、佇列、堆疊與樹等結構)則屬於**動態結構**。

07 陣列

7.2 宣告陣列

- 陣列佔一塊連續的記憶體空間，必須標明每一個元素的型態，以及元素的個數，編譯器才能決定出所需的記憶體空間。例如

```
int k[8]; /*宣告一個八個元素的整數陣列k*/
```

上述宣告告訴電腦幫您保留一個八個元素的整數陣列

```
int a[6], b[13], c[52];
```

一個宣告敘述可以同時宣告若干個陣列，以逗號隔開。

07 陣列



宣告陣列(續)

- 陣列元素在宣告時同時給予初值，如下例：

```
int k[] = {64,38,16,49,91,73,82,37};
```

宣告一個整數陣列k，含八個元素，其值分別為
k[0]=64、k[1]=38、k[2]=16、k[3]=49、
k[4]=91、k[5]=73、k[6]=82及k[7]=37。

- 注意：方括號內可以不標明陣列元素個數，編譯器會決定其個數

07 陣列

◀ ◁ ▶ ▷

宣告陣列(續)

- 字元陣列宣告

```
char c[] = {'A', 'a', '1', '+'};
```

宣告一個字元陣列c，含四個元素，其值分別為c[0]='A'、c[1]='a'、c[2]='1'及c[3]='+'。

- 當明確定義陣列大小時，若設定的陣列元素初始值不足，則編譯器會將剩餘未設定初值的元素內容設定為0（針對數值陣列而言）

```
float Temper[12]={}; // 同時將12個元素內容皆設定為0
```

07 陣列

◀ ◁ ▶ ▷

一維陣列

記憶體位置 *char name[6]*; 記憶體位置 *int score[6]*; 記憶體位置 *double rate[6]*;

圖6-1 陣列的記憶體配置

陳錦輝，C/C++初學指引，金禾書局(R501)。

◀ ◁ ▶ ▷

【程式intarray.c】

```
***** intarray.c *****/
#include <stdio.h>
int main()
{
    int k[]={64, 38, 16, 49, 91, 73, 82, 37};
    double d[]={1.2, 3.4, 5.6};
    char c[]{"A", 'a', '1', '+'};
    printf("c[1]=%c\n", c[1]);
    printf("d[2]=%lf\n", d[2]);
    printf("k[3]=%d\n", k[3]);
    system("pause");
    return 0;
}
```

07 陣列

◀ ◁ ▶ ▷

```
***** initarray.c *****/
#include <stdio.h>
int main()
{
    int k[8]={0}, m[8];
    printf("k[3]=%d\n", k[3]);
    printf("k[5]=%d\n", k[5]);
    printf("k[7]=%d\n", k[7]);
    printf("m[3]=%d\n", m[3]);
    printf("m[5]=%d\n", m[5]);
    printf("m[7]=%d\n", m[7]);
    system("pause");
    return 0;
}
```

【執行】

```
initarray.exe ←
k[3]=0
k[5]=0
k[7]=0
m[3]=-1
m[5]=2293728
m[7]=2013471392
```

07 陣列

◀ ◁ ▶ ▷

7.3 陣列長度

sizeof 運算子可以求得每一個陣列長度（元素個數以位元組計算）。

```
***** lenarray.c *****/
#include <stdio.h>
int main()
{
    int k[]={64, 38, 16, 49, 91, 73, 82, 37};
    float f[]={1.2, 3.4, 5.6};
    char c[]{"A", 'a', '1', '+'};
    printf("整數陣列k的長度(元素個數)=%d\n",
           sizeof(k)/sizeof(int));
    printf("浮點數陣列f的長度(元素個數)=%d\n",
           sizeof(f)/sizeof(float));
    printf("字元陣列c的長度(元素個數)=%d\n",
           sizeof(c)/sizeof(char));
    system("pause");
    return 0;
}
```

07 陣列

◀ ◁ ▶ ▷

7.4 陣列範例

```
***** evenarray.c *****/
#include <stdio.h>
#define SIZE 8
int main()
{
    int a[SIZE], i;
    for (i=0; i<SIZE; i++)
        a[i] = 2 * i; //使用for重複結構敘述建立整數陣列a的初值
    printf("%s%13s\n", "元素名稱", "元素值");
    for (i=0; i<SIZE; i++)
        printf(" a[%d] %12d\n", i, a[i]);
    return 0;
}
```

07 陣列

```

***** summaray.c *****/
#include <stdio.h>
int main()
{ int k[]={24, 38, 16, 49, 51, 33, 42, 37};
  int i, SIZE=sizeof(k)/sizeof(int); //使用sizeof計算整數陣列a的個數
  float sum=0.0, avg;
  for (i=0; i<SIZE; i++)
    sum = sum + k[i];
  avg = sum / SIZE;
  printf("sum=%6.2f avg=%5.2f\n", sum, avg);
  return 0;
}

```

07 陣列

```

***** vote.c *****/
#include <stdio.h>
#define STUDENT 50
#define NUM 4
int main()
{
  int vote[STUDENT]=
  {
    1,3,1,2,3,1,3,3,1,2, 3,3,1,1,1,3,3,3,3,2,
    3,3,3,1,1,3,3,1,2,2, 2,3,3,3,1,1,3,3,3,3,
    1,3,3,3,1,2,2,3,3,3
  };
  int i, count[NUM]={0};
  for (i=0; i<STUDENT; i++) count[vote[i]]++;
  printf("number count\n");
  for (i=1; i<NUM; i++)
    printf("%3d %d\n", i, count[i]);
  return 0;
}

```

07 陣列

```

***** diecount.c *****/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 7
int main()
{
  int i, point, count[SIZE]={0};
  srand(time(NULL));
  for (i=0; i<10000; i++)
  {
    point = rand() % 6 + 1;
    count[point]++;
  }
  printf("point count\n");
  for (i=1; i<SIZE; i++)
    printf("%3d %d\n", i, count[i]);
  return 0;
}

```

07 陣列

7.5 陣列與函式呼叫

- 要將陣列當函式呼叫的引數，只要標明陣列的名稱以及總共元素個數就可以了。
- 例如下列 `doublearray()` 函式，有兩個參數，第一個參數為整數陣列 `b`，第二個參數為 `b` 陣列的元素個數 `size`(整數)。

```

void doublearray(int b[], int size)
{
  int i;
  printf("\nb陣列的位址=%p\n", b);
  for (i=0; i<size; i++) b[i]=b[i]*2;
}

```

07 陣列

- 在主函式 `main()` 中呼叫

```

doublearray(a,SIZE);

```

- 呼叫成功時會將 **陣列a的位址值** 傳給第一個參數 `b`，將 `SIZE` 值傳給第二個參數 `size`。

進入函式 <code>doubleArray()</code> 時						
	<table border="1"> <tr><td>8</td></tr> <tr><td>size</td></tr> <tr><td>● → [1 2 3 4 5 6 7 8]</td></tr> <tr><td>b</td></tr> <tr><td>b[0] b[1] b[2] b[3] b[4] b[5] b[6] b[7]</td></tr> </table>	8	size	● → [1 2 3 4 5 6 7 8]	b	b[0] b[1] b[2] b[3] b[4] b[5] b[6] b[7]
8						
size						
● → [1 2 3 4 5 6 7 8]						
b						
b[0] b[1] b[2] b[3] b[4] b[5] b[6] b[7]						
離開 <code>doubleArray()</code> 函式後						
	<table border="1"> <tr><td>2 4 6 8 10 12 14 16 </td></tr> <tr><td>a</td></tr> <tr><td>a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7]</td></tr> </table>	2 4 6 8 10 12 14 16	a	a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7]		
2 4 6 8 10 12 14 16						
a						
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7]						

07 陣列

函式呼叫傳值與函式呼叫傳址

- 函式傳值呼叫：**以元素型態複製一份資料給函式始用。函式傳值時雖然在函式中改變其值，但在呼叫方仍然保持原值不變。
- 函式傳址呼叫：**以指定位址告訴函式直接就此位址的內容做修改(處理)。函式傳址時，在函式中改變該位址的內含值，同時在呼叫方，該位址的內含值也跟著改變。
- 單獨之陣列元素宜採取傳值呼叫；整個陣列內容的資料處理宜採取傳址呼叫。**

07 陣列

```

***** passarray.c *****
#include <stdio.h>
void doublearray (int b[], int size) // called by address
{
    int i;
    printf("nb陣列的位址=0x%p\n", b);
    for (i=0; i<size; i++) b[i]=b[i]*2;
}
int main()
{
    int a[]={1,2,3,4,5,6,7,8};
    int i, SIZE=sizeof(a)/sizeof(int);
    printf("a陣列的位址=0x%p\n", a);
    printf("呼叫doublearray()函式前之a陣列\n");
    for (i=0; i<SIZE; i++) printf("%4d",a[i]);
    doublearray(a, SIZE);
    printf("呼叫doublearray()函式後之a陣列\n");
    for (i=0; i<SIZE; i++) printf("%4d",a[i]);
    return 0;
}

```

07 陣列

```

***** inputarray.c *****
#include <stdio.h>
#define SIZE 8
void inputArray ( int a[], int size )
{
    //略
}
void printArray ( int a[], int size )
{
    //略
}
int main()
{
    int a[SIZE];
    inputArray(a, SIZE);
    printArray(a, SIZE);
    return 0;
}

```

07 陣列

```

***** sumup.c *****
#include <stdio.h>
int sumup( int a[], int size )
{
    //略
}
void printarray( int a[], int size )
{
    //略
}
int main()
{
    int a[]={1,2,3,4,5,6,7,8}, sum;
    int SIZE=sizeof(a)/sizeof(int);
    printf("陣列 a 的元素值:\n");
    printarray(a, SIZE);
    sum = sumup(a, SIZE);
    printf("陣列 a 的元素值之和=%d\n", sum);
    return 0;
}

```

07 陣列

```

***** charaddr.c *****
charaddr.exe ←
字元陣列ca的位址 = 0x0022FF60
字元陣列ca每個元素的位址如下:
&ca[0] = 0x0022FF60
&ca[1] = 0x0022FF61
&ca[2] = 0x0022FF62
&ca[3] = 0x0022FF63
&ca[4] = 0x0022FF64

```

	0x0022FF60	0x0022FF62	0x0022FF64		
ca	ca[0]	ca[1]	ca[2]	ca[3]	ca[4]

可以看到第0個元素ca[0]與ca的位址相同

07 陣列

7.6 陣列與排序

- 排序 (sort) 是將一組資料依指定的順序排列，可以是由小到大的遞升順序，或從大到小的遞降順序。
- 若是字元型資料排序的話要留意編碼系統是BCD碼，ASCII碼，EBCDIC碼，或是unicode？
- 一般個人電腦最常用的是ASCII碼，其字元'A'的相對應數值為65、「B」為66、「C」為67等等，'a'為97、「b」為98、「c」為99等等，'0'為48、「1」為49、「2」為50等等。

07 陣列

氣泡排序法 (bubble sort)

{24,7,36,2,65}

相鄰的兩數加以比較。
•若左邊的數比右邊的大就互換，換句話說
 若 $a(j) > a(j+1)$
 則 $a(j)$ 內含值與 $a(j+1)$ 內含值互換
•
 若左邊的數比右邊的小或相等，就不互換。

07 陣列

◀ ▶ ⟲ ⟳

氣泡排序法 (Bubble Sort)

- 『氣泡排序法』是一種非常簡單且容易的排序方法（效率普通）
- 『氣泡排序法』將相鄰的兩個資料一一互相比較，依據比較結果，決定資料是否需要對調，由於整個執行過程，有如氣泡逐漸浮上水面，因而得名，其方法及示意圖如下：
- 假設我們有{24,7,36,2,65}要做氣泡排序，最後的排序結果為{2,7,24,36,65}。

陳錦輝, C/C++初學指引, 金禾書局(R501)。

◀ ▶ ⟲ ⟳

```

for (i=1; i<SIZE; i++)
{
    for (j=0; j<SIZE-i; j++)
    {
        if ( a[j] > a[j+1] )
        {
            temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
        }
    }
    for (j=0; j<SIZE-i+1; j++) printf("%4d",a[j]);
    printf("\n");
}

```

07 陣列

◀ ▶ ⟲ ⟳

氣泡排序法 (Bubble Sort) 【補充】

```

*****
    檔名:ch6_04.cpp
    功能:陣列與排序
*****
#include <iostream>
#include <stdlib.h>

using namespace std;

int main(void)
{ int x[6]={25,10,39,40,33,12};
  int spec=11; //特別號
  int k,times,i,temp;
  *****
  k=6-1;
  while(k!=0)
  {times=0;
   for(i=0;i<k-1;i++)
   { if(x[i]>x[i+1])
     { //x[i]與x[i+1]互換
       temp=x[i]; x[i]=x[i+1]; x[i+1]=temp;
       times=i;
     }
   }
   k=times;
  }
  for(i=0;i<6;i++)
  cout << x[i] << "\t"; //印出排序後的結果
  cout << "n特別號\t" << spec << "n";
  // system("pause");
  return 0;
}

```

陳錦輝, C/C++初學指引, 金禾書局(R501)。

◀ ▶ ⟲ ⟳

```

***** qsort.c *****
#include <stdio.h>
#include <stdlib.h>
int comp(int *arg1, int *arg2)
{
    if (*arg1 < *arg2)
        return -1;
    else if (*arg1 > *arg2)
        return 1;
    else
        return 0;
}
int main()
{
    int i, a[] = {84, 78, 16, 69, 96, 33, 82, 77, 55};
    int N = sizeof(a)/sizeof(int);
    qsort(a, N, sizeof(int), comp);
    for (i=0; i<N; i++) printf("%5d",a[i]);
    return 0;
}
?
```

07 陣列

◀ ▶ ⟲ ⟳

7.7 陣列與搜尋法

- 若要知道某項是否在陣列元素中，那該怎麼辦呢？
- linear search:** 當然您可以將某項與陣列中每一個元素比較，若相等就找到了，若一直都不相等，那就找不到。若陣列有n個元素，找到平均次數為 $n/2$ 。
- binary search:** 對於已經排序的陣列，先從陣列的中央開始搜尋，若該中央記錄比某項還小，則往大的剩餘另一半搜尋；若中央記錄比某項還大，則往小的剩餘另一半搜尋；若相等，則代表找到某項。重複此步驟直到找到某項為止，或者發現要搜尋的某項不存在。在最壞的狀況下，最多的比較次數為 $\log_2 n$ 。

07 陣列

◀ ▶ ⟲ ⟳

binary search

```

#include <stdlib.h>
void *bsearch(void *item, void *array, size_t number,
              size_t size, int (*comp)(void *arg1, void *arg2) )

```

搜尋鍵item，array陣列必須已經排序過。有number個元素，每個元素占size個位元組記憶空間。

comp() 函式傳回值如下：

- 若 *arg1 < *arg2 傳回負數。
- 若 *arg1 == *arg2 傳回0。
- 若 *arg1 > *arg2 傳回正數。

07 陣列

```

***** bsearch.c *****/
#include <stdio.h>
#include <stdlib.h>
int comp (int *arg1, int *arg2) { //略 }
int main()
{ int i,*ip, item=33;
  // int a[] = {84, 78, 16, 69, 96, 33, 82, 77, 55};
  // qsort (a, N, sizeof(int), comp);
  int a[ ] = {16, 33, 55, 69, 77, 78, 82, 84, 96 };
  int N = sizeof(a)/sizeof(int);
  for (i=0; i<N; i++) printf("%d",a[i]);
  printf("\n");
  ip = (int *) bsearch (&item, a, N, 4, comp );
  if (ip==NULL)
    printf("%d找不到。%n", item);
  else
  { i = ip-&a[0];
    printf("%d在第%d位置(從0算起)找到。%n", item,i);
  }
  return 0;
}

```

?

07 陣列



7.8 多維陣列

- 在C語言中的陣列可以具有多重註標值，這種陣列稱為**多維陣列**。
- 若具有二個註標值，稱為二維陣列**，ANSI的C語言最少必須支援到12重註標值。
- 二維陣列資料表，**橫的稱為列，直的稱為行**。

07 陣列

```

int a[ROW][COL] = { {1, 2, 3}, {4, 5, 6} };

```

	行 0	行 1	行 2
列 0	a[0][0]	a[0][1]	a[0][2]
列 1	a[1][0]	a[1][1]	a[1][2]

↑ ↑ ↑
陣列名稱 行註標 列註標

a[0][0] 表示a陣列中第0列第0行的元素

	行 0	行 1	行 2
a	a[0][0]	a[0][1]	a[0][2]
列 0	1	2	3
列 1	4	5	6

↑
a[1][0] a[1][1] a[1][2]

07 陣列

```

***** table.c *****/
#include <stdio.h>
#define ROW 2
#define COL 3
void printarray ( int a [ROW] [COL] ) { /*略*/ }
int main()
{ int a[ROW][COL] = { {1, 2, 3}, {4, 5, 6} };
  int b[ROW][COL] = { {11, 12, 13}, {14, 15} };
  int c[ROW][COL] = { {21, 22}, {24} };
  printf("陣列a每列如下:\n");
  printarray(a);
  printf("陣列b每列如下:\n");
  printarray(b);
  printf("陣列c每列如下:\n");
  printarray(c);
  return 0;
}

```

07 陣列

Open question

- What's the memory allocation of two-dimensional array?
- What's the memory allocation of multiple dimensional array?
- What's the relationship between pointer and array?
- What's the relationship between string and character array?

07 陣列

08 指標



查看全圖。

- www.weilou.com/Code/Rcpt/index.htm
- [8.1 告白及初始化](#)
 - [8.2 指標運算子](#)
 - [8.3 函式呼叫](#)
 - [8.4 指標算術](#)
 - [8.5 指標陣列](#)
 - [8.6 函式指標](#)

- C語言一個最強大的功能，就是**指標**(pointer)，它也是最難理解的一個項目，指標讓設計者能夠模擬函式的**傳址呼叫**、建立並操作**動態資料結構**，例如動態的連結串列、併列、堆疊、以及樹等結構，這種結構號稱動態，意即其節點隨時可以增減。

08 指標

8.1 告白及初始化

- 指標變數事實上就是一個變數，只不過它的內含值是一個位址而已。

```
char ch='A';
char *p=&ch;
```

變數p宣告為char *型態

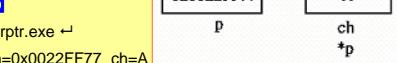
- 告白時*運算子表示所宣告的變數屬於指標變數，
- 運算子&表示取其後變數的位址。

08 指標

```
***** charptr.c *****
```

```
#include <stdio.h>
int main()
{
    char ch='A';
    char *p=&ch;
    printf("&ch=0x%p ch=%c\n", &ch, ch);
    printf("&p=0x%p p=0x%p *p=%c\n", &p, p, *p);
    return 0;
}
```

【執行】



08 指標

```
***** intptr.c *****
```

```
#include <stdio.h>
int main()
{
    int k=123;
    int *p=&k;
    printf("&k=0x%p k=%d\n", &k, k);
    printf("&p=0x%p p=0x%p *p=%d\n", &p, p, *p);
    return 0;
}
```

【執行】

```
intptr.exe ←
&k=0x0022FF74 k=123
&p=0x0022FF70 p=0x0022FF74 *p=123
```

08 指標

8.2 指標運算子

- 指標運算子&稱為位址運算子，是一個一元運算子，傳回運算元的位址。

```
int k=123;
int *p=&k; /*傳回運算元k的位址指定給p*/
```

或

```
p=&k; /*傳回運算元k的位址指定給p*/
```

- 位址運算子&的運算元必須為變數，不能為常數、運算式。
- 指標運算子*又稱為間接運算子，傳回運算元所指位址的內含值。如下例：

```
printf("%d", *p);
```

- 將p所指位址的內含值顯示出來。

08 指標

◀ ◁ ▶ ▷

8.3 函式呼叫

- 函式呼叫有兩種方式，傳值及傳址呼叫。
- 傳值呼叫的引數值為一般的數值，
- 傳址呼叫所傳的引數值為位址值。

08 指標

◀ ◁ ▶ ▷

```
***** byval.c *****
#include <stdio.h>
int cube(int n)
{
    n = n*n*n;
    return n;
}
int main()
{
    int i=4, j;
    printf("呼叫cube()函式之前 i=%d\n", i);
    j = cube(i);
    printf("呼叫cube()函式之後 i=%d j=%d\n", i,j);
    return 0;
}
```

08 指標

◀ ◁ ▶ ▷

```
***** byaddr.c *****
#include <stdio.h>
void cubeByRef ( int *p )
{
    *p = (*p) * (*p) * (*p);
}
int main()
{
    int i = 4;
    printf("呼叫cube()函式之前 i=%d\n", i);
    printf("呼叫cube()函式之前 &i=%p\n", &i);
    cubeByRef(&i);
    printf("呼叫cube()函式之後 i=%d\n", i);
    printf("呼叫cube()函式之後 &i=%p\n", &i);
    return 0;
}
```

08 指標

◀ ◁ ▶ ▷

```
***** convert.c *****
#include <stdio.h>
void conv ( char *p )
{
    while (*p!='\0')
        if (*p<='a' && *p<='z') *p=*p-'a'+'A';
        p++;
}
int main()
{
    char s[] = "How are you?";
    printf("呼叫conv()之前 s=\"%s\" &s=0x%p\n", s,&s);
    conv(s);
    printf("呼叫conv()之後 s=\"%s\" &s=0x%p\n", s,&s);
    return 0;
}
```

08 指標

◀ ◁ ▶ ▷

- 定義一個 conv() 函式，參數p為指向一個字元的指標，該指標所指位址的內含值為「*p」。
- 判斷該內含值是否為字串結束符號'\0'，若不是，則檢查是否為英文小寫字母；若是，則將它轉成英文大寫字母，然後p指到下一個字元位址，繼續檢查該內含值是否為字串結束符號'\0'，一直到該內含值'\0'才停止下來。

0x0022FF60

&s [0x0022FF60] 呼叫 conv() 開始時

0x0022FF60

*p ↑

p [0x0022FF60] 呼叫 conv() 結束

08 指標

◀ ◁ ▶ ▷

- 呼叫 conv(s) 函式，開始時p指到s字串的第0個位址，即0x0022FF60，它的內含字元為'H'，以*p表示之。
- 執行p++後p指到0x0022FF60+1，此時p*為'o'，將這個小寫字母轉成大寫字母'O'之後，
- 重複執行p++及小寫轉換為大寫，直到p指向'\0'時才結束 conv() 函式的執行，返回 main() 函式。

0x0022FF60

*p ↑

p [0x0022FF60+12] 呼叫 conv() 結束

08 指標

8.4 指標算術

- 在算術運算式、指定運算式、以及比較運算式中指標是一個正確的運算元，在其他的運算式中指標並不是正確的運算元，這一點請特別注意。
- 指標的運算也只限於有限的幾個運算子而已：
 - 指標的遞加`++`、指標的遞減`--`
 - 指標的整數加法`+或+=`、指標的整數減法`-或-=`
- 事實上，指標內容指向令一個變數的位址，要了解指標的運算最好能夠了解資料在電腦記憶體中的表示法。

08 指標

***** size.c *****

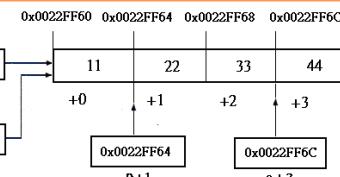
```
#include <stdio.h>
int main()
{ printf(" sizeof(char)=%2d\n",sizeof(char));
  printf(" sizeof(short)=%2d\n",sizeof(short));
  printf(" sizeof(int)=%2d\n",sizeof(int));
  printf(" sizeof(long)=%2d\n",sizeof(long));
  printf(" sizeof(float)=%2d\n",sizeof(float));
  printf(" sizeof(double)=%2d\n",sizeof(double));
  printf(" sizeof(long double)=%2d\n",sizeof(long double));
  return 0;
}
```

To know the advance step of a variable

08 指標

```
***** arrayptr.c *****
#include <stdio.h>
int main()
{
    int a[] = {11, 22, 33, 44};
    int i, j, SIZE=sizeof(a)/sizeof(int);
    int *p = a;
    printf(" 整數陣列a每一個元素的位址\n");
    for (i=0; i<SIZE; i++)
        printf(" &a[%d]=0x%p\n", i, &a[i]);
    printf(" 陣列a以指標及註標方式表示\n");
    for (i=0; i<SIZE; i++)
        printf(" p[%d]=%d\n", i, p[i]);
    return 0;
}
```

08 指標



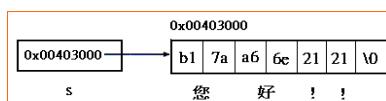
08 指標

```
***** strptr.c *****
#include <stdio.h>
int main()
{ char *s = "您好!!";
  int i;
  for (i=0; *(s+i); i++)
  { printf("i=%d &(s+i)=0x%04x *(s+i)=0x%02x\n",
         i, s+i, *(s+i) & 0x00ff);
  }
  return 0;
}

【執行】
strptr.exe ←
i=0 &(s+i)=0x00403000 *(s+i)=0xb1 ) 您
i=1 &(s+i)=0x00403001 *(s+i)=0x7a
i=2 &(s+i)=0x00403002 *(s+i)=0xa6
i=3 &(s+i)=0x00403003 *(s+i)=0x6e ) 好
i=4 &(s+i)=0x00403004 *(s+i)=0x21
i=5 &(s+i)=0x00403005 *(s+i)=0x21
```

08 指標

- 指標變數s的位址從0x00403000開始，其內含值為十六進位b1，位址從0x00403001其內含值為7a，這兩個位元組b17a就是以中文Big5碼表示的"您"。
- 位址0x00403002至0x00403003位址的內含a66e就是以中文Big5碼表示的"好"。
- 0x00403004、0x00403005位址的內含值21，表示ASCII碼的'!'字元。
- 0x00400006位址的內含值'\0'，表示字串結束。



08 指標

◀ ◁ ▶ ▷

8.5 指標陣列

- 字元指標陣列char *suit[4]包含四個元素，每個元素均為一個字串

```
char *suit[4] = {"C", "D", "H", "S"};
```

08 指標

◀ ◁ ▶ ▷

- 字元指標陣列char *point[13]包含13個元素，每個元素均為一個字串。

```
char *point[13] = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King" };
```

0	→	'A'	\0'
1	→	'2'	\0'
2	→	'3'	\0'
3	→	'4'	\0'
4	→	'5'	\0'
5	→	'6'	\0'
6	→	'7'	\0'
7	→	'8'	\0'
8	→	'9'	\0'
9	→	'1'	\0' \0'
10	→	'T'	ack \0
11	→	'Q'	ueen \0
12	→	'K'	ing \0

point

08 指標

◀ ◁ ▶ ▷

字串與二維陣列

```
char Week[7][10]={"Monday","Tuesday","Wednesday","Thursday",
"Friday","Saturday","Sunday"};
```

0x2000	M	o	n	d	a	y	\0			
0x200a	T	u	e	s	d	a	y	\0		
0x2014	W	e	d	n	e	s	d	a	y	\0
0x201e	T	h	u	r	s	d	a	y	\0	
0x2028	F	r	i	d	a	y	\0			
0x2032	S	a	t	u	r	d	a	y	\0	
0x203c	S	u	n	d	a	y	\0			

圖8-13 使用二維陣列宣告方式儲存字串群

陳錦輝，C/C++初學指引，金禾書局(R501)。

◀ ◁ ▶ ▷

字串與一維指標陣列

```
char *Week[7]={"Monday","Tuesday","Wednesday","Thursday",
"Friday","Saturday","Sunday"};
```

0x2000	M	o	n	d	a	y	\0			
0x2007	T	u	e	s	d	a	y	\0		
0x200f	W	e	d	n	e	s	d	a	y	\0
0x2019	T	h	u	r	s	d	a	y	\0	
0x2022	F	r	i	d	a	y	\0			
0x2029	S	a	t	u	r	d	a	y	\0	
0x2032	S	u	n	d	a	y	\0			

圖8-14 使用指標陣列宣告方式儲存字串群

明顯地，它節省了一些記憶體空間

陳錦輝，C/C++初學指引，金禾書局(R501)。

◀ ◁ ▶ ▷

8.6 函式指標

- 函式的指標是指該函式在記憶體中的位址。
- 就像陣列一樣，陣列名稱為在記憶體中第0個元素的位址，**函式的名稱也是在記憶體中該函式的開始位址**。
- 函式指標可以當另一個函式的引數，也就是說函式定義中其參數可為另一個函式指標。
- 函式指標也可以當函式的傳值、儲存於陣列中、或指定為其他的函式指標等等。

08 指標

◀ ◁ ▶ ▷

```
***** funcptr.c *****/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void func0( int a )
{ printf("func0()函式的參數 a=%d\n", a); }
void func1( int a )
{ printf("func1()函式的參數 a=%d\n", a); }
int main()
{ int n;
  void (*f)(int);
  srand(time(NULL));
  n=rand();
  if (n%2==0)
    f=func0;
  else
    f=func1;
  (*f)(n); // firing a function
  return 0;
}
```

08 指標

```

***** sorting.c *****
#include <stdio.h>
#define SIZE 10
void bubble(int * , int (*)(int, int)); //function pointer
int ascending(int, int);
int descending(int, int);
int main()
{ int a[SIZE] = {22, 16, 4, 38, 20, 32, 89, 68, 15, 37};
  int i, order;
  printf("請輸入遞升(1)或遞降(2)排序: ");
  scanf("%d", &order);
  printf("\n資料原來順序\n");
  for (i=0; i<SIZE; i++) printf("%4d", a[i]);
  if (order==1)
  { bubble(a, SIZE, ascending); // apply the function pointer
    printf("\n遞升排序後\n");
  }
  else
  { bubble(a, SIZE, descending); // apply the function pointer
    printf("\n遞降排序後\n");
  }
  for (i=0; i<SIZE; i++) printf("%4d", a[i]);
  return 0;
}

```

08 指標

```

***** sorting.c (cont.) *****
void bubble(int *work, int size, int (*compare)(int, int) ) //function
pointer
{ int i, j;
  void swap(int *, int *);
  for (i=0; i<size; i++)
    for (j = 0; j<size-1; j++)
      if ((*compare)(work[j], work[j+1]))
        swap(&work[j], &work[j+1]);
}

void swap(int *a, int *b)
{ int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
int ascending(int a, int b)
{ return a > b;
}
int descending(int a, int b)
{ return a < b;
}

```

Craftsmanlike
an unrivaled sight

08 指標

- 程式sorting.c執行排序工作，這個工作委託bubble()函式來做，這個函式的第一個參數為整數指標，指到要排序的數列，第二個參數為整數，指到數列的個數，第三個參數為函數的指標，所指的函式是有二個整數參數，並傳回一個整數值。

```
bubble(a, SIZE, ascending);
bubble(a, SIZE, descending);
```

- 實際呼叫bubble()函式，其中a為整數陣列，SIZE為陣列元素個數，ascending及descending為函式名稱，遞升排序時使用ascending()，遞降排序時使用descending()。

08 指標

09 字元與字串

9.1 字元處理函式庫

9.3 字串轉換函式

9.4 字元、字串輸入及輸出

9.2 字串長度*

9.5 字串操作函式

9.6 字串比較函式

9.7 字串搜尋函式

9.8 字串記憶體函式*

9.9 字串錯誤訊息

字元、字串

- **字元**是構成字串的元素。
- **字元常數**是一個以單引號括起來的字元，表示某一符號的整數值，例如字元常數‘A’。
- **字元常數'\0'**，也稱為NULL常數。
- **字串**是由含有字母、數字及數種特殊的字元所組成，以字元常數'\0'當字串結束符號。
- **字串常數**以含在兩個雙引號間的字元表示。

```
"Good Morning!"  
"Kay Lin"  
"(07)2259999"  
"您好!"  
"歡迎光臨"
```

09 字元與字串

字串及字元陣列

`char color[]={'r','e','d'};`
字元陣列color，包含3個字元元素'r', 'e', 'd'。
不能稱為**字串**

`char color[]={'r','e','d','\0'};`
字元陣列color，包含4個字元元素'r', 'e', 'd'
及'\0'；也是**字串**。

字元陣列 color

 字元指標 p

陣列名稱具有指標功能，指向為陣列的開始元素的位址。

09 字元與字串

【程式chartest.c】

```
/****** chartest.c *****/  
#include <stdio.h>  
int main()  
{  
    char color[]={'r','e','d','\0'};  
    char *p="blue";  
    int i;  
    for (i=0; i<4; i++) printf("%x ", color[i]);  
    printf("\n");  
    for (i=0; i<5; i++) printf("%x ", *(p+i));  
    printf("\n%s\n", color);  
    printf("%s\n", p);  
    return 0;  
}
```

【執行】

```
gcc chartest.c -o chartest.exe <Enter>  
chartest.exe <Enter>  
72 65 64 0 [註] r' e' 'd' '\0'  
62 6c 75 65 0 [註] b' l' u' e' '\0'  
red  
blue
```

09 字元與字串

General C++

- Pre-processor commands
- Operator Precedence
- Escape Sequences
- ASCII Chart
- Data Types
- Keywords

Standard C Library

- Standard C I/O
- Standard C String & Character
- Standard C Math
- Standard C Time & Date
- Standard C Memory
- Other standard C functions

C++ Reference

Search

09 字元與字串

9.1 字元處理函式庫

- 字元處理函式庫處理字元資料測試及操作的函式，
- 字元引數EOF表示檔案結束符號，其值為整數值-1。
- 字元處理函式定義於表頭檔ctype.h中。

int isalnum(int c)

- 若c為字母A-Z、a-z、數字0-9，則傳回非0值，否則傳回0值。

int isalpha(int c)

- 若c為字母A-Z、a-z則傳回非0值，否則傳回0值。

09 字元與字串

<http://www.cs.tcu.edu/~magi/173/char-funcs.html>

Some Useful C Character Functions

Function	Return true if
int isalpha(c);	c is a letter.
int isupper(c);	c is an upper case letter.
int islower(c);	c is a lower case letter.
int isdigit(c);	c is a digit [0-9].
int isxdigit(c);	c is a hexadecimal digit [0-9A-Fa-f].
int isalnum(c);	c is an alphanumeric character (c is a letter or a digit).
int isblank(c);	c is a SPACE, TAB, RETURN, NEWLINE, FORMFEED, or vertical tab character.
int ispace(c);	c is a punctuation character (neither control nor alphanumeric).
int isprint(c);	c is a printing character.
int iscntrl(c);	c is a delete character or ordinary control character.
int isascii(c);	c is an ASCII character, code less than 0200.

int toupper(int c); convert character c to upper case (leave it alone if not lower)
int tolower(int c); convert character c to lower case (leave it alone if not upper)

Don't forget to #include <ctype.h> to get the function prototypes.

09 字元與字串

```
***** inputchar.c *****/
#include <stdio.h>
#include <ctype.h>
int main()
{
    char ch;
    printf("請輸入一個字元: ");
    scanf("%c", &ch);
    if (isdigit(ch))
        printf(" %c 是數字\n", ch);
    else
        printf(" %c 非數字\n", ch);
    if (isalpha(ch))
        printf(" %c 是英文字母\n", ch);
    else
        printf(" %c 非英文字母\n", ch);
    return 0;
}
```

09 字元與字串

```
***** lower.c *****/
#include <stdio.h>
#include <ctype.h>
int main()
{
    char ch;
    printf("請輸入一個字元: ");
    scanf("%c", &ch);
    if (islower(ch))
    {
        printf(" %c 是小寫英文字母\n", ch);
        printf(" %c 轉成大寫字母為 %c\n", ch, toupper(ch));
    }
    else
        printf(" %c 非小寫英文字母\n", ch);
    if (isupper(ch))
    {
        printf(" %c 是大寫英文字母\n", ch);
        printf(" %c 轉成小寫字母為 %c\n", ch, tolower(ch));
    }
    else
        printf(" %c 非大寫英文字母\n", ch);
    return 0;
}
```

09 字元與字串

```
***** testctrl.c *****/
#include <stdio.h>
#include <ctype.h>
int main()
{
    printf("\n' %s\n", isspace('\n') ? "是空白字元" : "不是空白字元");
    printf("'\007' %s\n", iscntrl('\007') ? "是控制字元" : "不是控制字元");
    printf("分號 ';' %s\n", ispunct(';') ? "是標點符號字元" : "不是標點符號字元");
    printf("'A' %s\n", isprint('A') ? "是可印字元" : "不是可印字元");
    printf("空白 '' %s\n", isgraph(' ') ? "是可繪圖字元" : "不是可繪圖字元");
    return 0;
}
```

09 字元與字串

9.3 字串轉換函式

String conversion (1)

- 字串轉換函式將由數字所組成的字串轉換成整數、長整數、或浮點數。
- 字串轉換函式定義於公用函式庫stdlib.h裡

`double atof(const char *s)`

- 將s字串轉換並傳回**浮點數**。轉換不成功時傳回0值。

`int atoi(const char *s)`

- 將s字串轉換並傳回**整數值**。轉換不成功時傳回0值。

`long atol(const char *s)`

- 將s字串轉換並傳回**長整數值**。轉換不成功時傳回0值。

09 字元與字串

```
***** convert.c *****/
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int i;
    long li;
    double d;
    d = atof("123.45");
    printf("atof(\"123.45\")= %.2f\n", d);
    i = atoi("12345");
    printf("atoi(\"12345\")= %d\n", i);
    li = atol("123456789");
    printf("atol(\"123456789\")= %ld\n", li);
    return 0;
}
```

ex D:\1_2008_962\CSU2008_962\convert.exe

atof("123.45")= 123.45
atoi("12345")= 12345
atol("123456789")= 123456789
請按任意鍵繼續 . . .

09 字元與字串

◀ ◁ ▶ ▷

9.3 字串轉換函式 String conversion (2)

```

double strtod ( const char * str, char ** endptr );
Convert string to double

long int strtol ( const char * str, char ** endptr, int base );
Convert string to long integer

unsigned long int strtoul ( const char * str, char ** endptr, int base );
Convert string to unsigned long integer

字串轉換函式定義於公用函式庫<stdlib.h> <cstdlib>裡

http://www.cplusplus.com/reference/clibrary/cstdlib/strtod.html
http://www.cplusplus.com/reference/clibrary/cstdlib/strtol.html
http://www.cplusplus.com/reference/clibrary/cstdlib/strtoul.html

```

09 字元與字串

◀ ◁ ▶ ▷

```

***** todecimal.c *****
#include <stdio.h>
#include <stdlib.h>
int main()
{
    double d;
    char *s = "12.3% ABC";
    long ld;
    char *sl = "-1234567abc";
    unsigned long u;
    char *su = "1234567abc";
    d = strtod(s, &sPtr);
    printf(" s=%-12s d=%-.2f sPtr=%s\n", s, d, sPtr);
    ld = strtol(sl, &slPtr, 0);
    printf(" sl=%-12s ld=%ld slPtr=%s\n", sl, ld, slPtr);
    u = strtoul(su, &suPtr, 0);
    printf(" su=%-12s u=%ld suPtr=%s\n", su, u, suPtr);
    return 0;
}

```

09 字元與字串

◀ ◁ ▶ ▷

9.4 字元字串輸入及輸出

標準輸入及輸出函式庫stdio.h

```

int getchar()
char *gets(char *buf)

int putchar(int ch)
int puts(const char *s)

int sscanf(const char *s, const char *format [, addr...])

int sprintf(char *s, const char *format [, argument ...])

```

09 字元與字串

◀ ◁ ▶ ▷

```

***** getline.c *****
#include <stdio.h>
void reverse(char *s)
{
    int i, n;
    for (n=0; s[n]!='\0'; n++);
    for (i=n-1; i>=0; i--) putchar(s[i]);
}
int main()
{
    char s[80];
    printf("輸入一個字串: ");
    gets(s);
    printf("反轉列印: ");
    reverse(s);
    return 0;
}

```

09 字元與字串

◀ ◁ ▶ ▷

```

***** keyinchar.c *****
#include <stdio.h>
int main()
{
    char ch, s[80];
    int i = 0;
    printf("輸入一個字串: ");
    while ( (ch = getchar() )!= '\n' ) s[i++]=ch;
    s[i] = '\0';
    puts("您輸入的字串為: ");
    puts(s);
    return 0;
}

```

09 字元與字串

◀ ◁ ▶ ▷

```

***** editstr.c *****
#include <stdio.h>
int main()
{
    char s[80], si[10], sf[80];
    int x;
    float y;
    printf("輸入一個整數x及一個浮點數y: ");
    scanf("%d%f", &x, &y);
    printf("整數: %6d 浮點數: %8.2f", x, y);
    printf("輸出至字串s為:\n%s\n", s);
    sscanf(s, "%6s%6d%8s%8.2f", si,&x, sf,&y);
    printf("從字串s使用sscanf()函式讀入 : \n"
           "si=%s" x=%d sf=%s" y=%..2f\n", si,x,sf,y);
    return 0;
}

```

09 字元與字串

◀ ▶ ⟲ ⟳

String Functions 字串處理函式庫

- `strcat(s1, s2)` - It concatenates two strings. It concatenates s2 at the end of s1.
- `strcmp (s1, s2)` - It is used to compare strings. It returns 0 if they are equal and less than 0 if s1<s2 and greater than 0 if s1>s2.
- **`strlen(s1)`** - It returns the length of the string s1.
- `strcpy(s1, s2)` - It copies s2 into s1.
- `strncat(s1, s2, n)` - It appends substring to the string. It takes first n characters of string s2 and appends s1.
- `strcmp(s1,s2,n)` - It compares first n characters of string s1 with first n characters of string s2.
- `strchr(s1, ch)` - Finds character ch in string s1. It returns the pointer at the first occurrence of ch.
- `strstr(s1, s2)` - Finds substring s2 in s1. It returns a pointer at the first occurrence of s1.

Either [header file <cstring>](#) or [<string.h>](#) must be included in the program. http://cpp-tutorial.cpp4u.com/C_style_string_functions.html

09 字元與字串

◀ ▶ ⟲ ⟳

9.2 字串長度*

`size_t strlen(const char *s)`

傳回字串s之長度，即字元數，不含字串結束符號'\0'字元(即NULL常數)
`size_t`型態為長整數型態。

09 字元與字串

◀ ▶ ⟲ ⟳

```
***** strength.c *****/
#include <stdio.h>
#include <string.h>

int main()
{
    char s1 = "abcdefghijklmnopqrstuvwxyz";
    char *s2 = "China";
    char *s3 = "Taiwan";
    printf("字串 s1 的長度 = %d\n", strlen(s1));
    printf("字串 s2 的長度 = %d\n", strlen(s2));
    printf("字串 s3 的長度 = %d\n", strlen(s3));
    return 0;
}
```

09 字元與字串

◀ ▶ ⟲ ⟳

9.5 字串操作函式

`char *strcpy(char *destination, const char *source)`

`char *strncpy(char *destination, const char *source, size_t n)`

由source字串最多拷貝n個字元至destination。若source之長度超過或等於n，則destination將不會以'\0'結束。傳回destination。

`char *strcat(char *destination, const char *source)`

`char *strncat(char *destination, const char *source, size_t n);`

09 字元與字串

◀ ▶ ⟲ ⟳

```
***** copystr.c *****/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char s[] = "Hi, Good Morning!";
    char s2[25], s3[15];
    printf("0000012345678901234567890\n");
    printf("s = \"%s\"\n s2=%\"s\"\n", s, strcpy(s2,s));
    strncpy(s3, s, 8);
    s3[8] = '\0';
    printf("s3=%\"s\"\n", s3);
    system("pause");
    return 0;
}
```

D:\1_2008_962CSUV2008_962C\copyStr.exe
 0000012345678901234567890
 s = "Hi, Good Morning!"
 s2="Hi, Good Morning!"
 s3="Hi, Good"
 請按任意鍵繼續 . . .

09 字元與字串

◀ ▶ ⟲ ⟳

```
/* strncpy example */
#include <stdio.h>
#include <string.h>
int main ()
{
    char str1[] = "To be or not to be";
    char str2[6];
    strncpy (str2,str1,5);
    str2[5]='\0';
    puts (str2);
    return 0;
}
```

D:\1_2008_962CSUV2008_962C\strncpy.exe
 012345678901234567890
 To be or not to be
 To be
 請按任意鍵繼續 . . .

09 字元與字串

◀ ▶ ⟲ ⟳

```
***** concat.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
    char s1[10] = "Hi, ";
    char s2[20] = "Good Morning.";
    char s3[40] = "";
    printf("s1=\"%s\"\n s2=\"%s\"\n", s1,s2);
    printf("strcat(s1,s2)= \"%s\"\n", strcat(s1,s2));
    printf("strncat(s3,s1,3)= \"%s\"\n", strncat(s3,s1,3));
    s3[3]='\0';
    strcpy(s2,"every body.");
    printf("strcat(s3,s2)= \"%s\"\n", strcat(s3,s2));
    return 0;
}
```

09 字元與字串

9.6 字串比較函式

int strcmp(const char *s1, const char *s2)

- 若s1<s2則傳回負數。
- 若s1==s2則傳回0值。
- 若s1>s2則傳回正數。

int strncmp(const char *s1, const char *s2, size_t n);

- 最多比較n個字元。
- 若s1<s2則傳回負數。
- 若s1==s2則傳回0值。
- 若s1>s2則傳回正數。

09 字元與字串

◀ ▶ ⟲ ⟳

```
***** compstr.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
    char *s1 = "Good Morning";
    char *s2 = "Good Morning";
    char *s3 = "Good Night";
    printf("s1=\"%s\" s2=\"%s\" s3=\"%s\"\n", s1,s2,s3);
    printf("strcmp(s1,s2)= %d\n", strcmp(s1,s2));
    printf("strcmp(s1,s3)= %d\n", strcmp(s1,s3));
    printf("strcmp(s3,s1)= %d\n", strcmp(s3,s1));
    printf("strcmp(s1,s3,5)= %d\n", strncmp(s1,s3,5));
    printf("strcmp(s1,s3,6)= %d\n", strncmp(s1,s3,6));
    printf("strcmp(s3,s1,7)= %d\n", strncmp(s3,s1,7));
    return 0;
}
```

09 字元與字串

9.7 字串搜尋函式

char *strchr(const char *s, int ch)

從s字串中由前向後搜尋第一個出現ch之位置。
失敗時傳回NULL。

char *strrchr(const char *s, int ch)

從s字串中由後向前搜尋第一個出現ch之位置(亦即傳回s中最後出現ch字元之指標)，失敗時傳回NULL。

char *strstr(const char *s1, const char *s2)

傳回s1中第一次出現s2的指標。

09 字元與字串

◀ ▶ ⟲ ⟳

```
***** charSearch.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
    char *s="Happy birthday!";
    char ch1='y', ch2='z';
    if ( strchr(s,ch1) !=NULL)
        printf("\'%c\' 在右列字串中找到 \'%s'\n", ch1,s);
    else
        printf("\'%c\' 在右列字串中找不到 \'%s'\n", ch1,s);
    if ( strchr(s,ch2) !=NULL)
        printf("\'%c\' 在右列字串中找到 \'%s'\n", ch2,s);
    else
        printf("\'%c\' 在右列字串中找不到 \'%s'\n", ch2,s);
    return 0;
}
```

09 字元與字串

◀ ▶ ⟲ ⟳

```
***** lastChar.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
    char *s="Happy New Year!!";
    int ch='Y';
    printf("在 \"%s\" 中最後出現\'%c\'字元的字串是\n", s,ch);
    printf("\'%s'\n", strchr(s,ch));
    return 0;
}
```

09 字元與字串

```

***** strSearch.c *****/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{ char *s1="Good Afternoon!!";
  char *s2="noon";
  printf("0000012345678901234567890\n");
  printf("s1=%s" s2="%s\n", s1,s2);
  printf("字串s1第一次出現s2的其餘字串為\n");
  printf("%s\n", strstr(s1,s2));
  printf("%d\n", strstr(s1,s2)-s1 );
  system("pause");
  return 0;
}

```

09 字元與字串

```

***** strtok.c *****/
#include <stdio.h>
#include <string.h>
int main()
{ char s[] = "This is a sentence with 7 tokens.";
  char *p;
  printf("要被剖成句元的字串\n%s\n", s);
  printf("剖成句元如下 :\n");
  p = strtok(s, " ");
  while (p!=NULL)
  { printf("%s\n", p);
    p = strtok(NULL, " ");
  }
  return 0;
}

```

09 字元與字串

9.8 字串記憶體函式*

- 字串函式庫string.h提供許多有關字串**記憶體函式**的操作，

```
void *memchr(const void *region, int ch, size_t n)
```

- 傳回region字串前n個字元第一個與ch相同字元的指標，失敗時傳回NULL值。

```
int memcmp(const void *r1, const void *r2, size_t count)
```

- 若r1<r2則傳回負數。若r1==r2則傳回0值。
- 若r1>r2則傳回正數。r1與r2以無符號字元逐字元比較，直到比完count個字元或分出大小為止。

09 字元與字串

```
void *memcpy(void *r1, const void *r2, size_t n)
```

- 由r2字串拷貝前n個字元至r1。傳回r1。

```
void *memmove(void *r1, const void *r2, size_t count)
```

- 由r2拷貝前n個字元至r1。傳回r1。
位置重疊也可以。

```
void *memset(void *buf, int ch, size_t n)
```

- 將buf前n個字元之內含設定為ch值。傳回buf。

09 字元與字串

```

***** memcpy.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
  char s1[17], s2[]="Memory copy this string.";
  printf("s2=%s\n", s2);
  memcpy(s1, s2, 11);
  s1[11]='\0';
  printf("字串s2使用memcpy()函式拷貝11個字元至s1\n");
  printf("s1=%s\n", s1);
  return 0;
}

```

09 字元與字串

```

***** movemem.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
  char s[] = "Home Sweet Home";
  printf("呼叫memmove()函式之前的字串: \"%s\"\n", s);
  printf("呼叫memmove()函式之後的字串: \"%s\"\n",
        memmove(s, &s[5], 10);
  return 0;
}

```

09 字元與字串

```

***** memcomp.c *****/
#include <stdio.h>
#include <string.h>
main()
{
    char s1[] = "abcdefg", s2[] = "abcdEFG";
    printf("s1=%s s2=%s\n", s1,s2);
    printf("相等 memcmp(s1,s2,4)=%d\n", memcmp(s1,s2,4));
    printf("s1大 memcmp(s1,s2,7)=%d\n", memcmp(s1,s2,7));
    printf("s2小 memcmp(s2,s1,7)=%d\n", memcmp(s2,s1,7));
    return 0;
}

【執行】
memcomp.exe ←
s1=abcdefg s2=abodEFG
相等 memcmp(s1,s2,4)=0
s1大 memcmp(s1,s2,7)=1
s2小 memcmp(s2,s1,7)=-1

```

09 字元與字串

9.9 字串錯誤訊息

- 函式`strerror()`會接收一個錯誤訊息號碼，然後傳回相對應的錯誤訊息字串，其函式宣告如下所示：

```
#include <string.h>
char *strerror(int error);
```

- 傳回對應錯誤訊息號碼`error`之錯誤訊息字串指標。

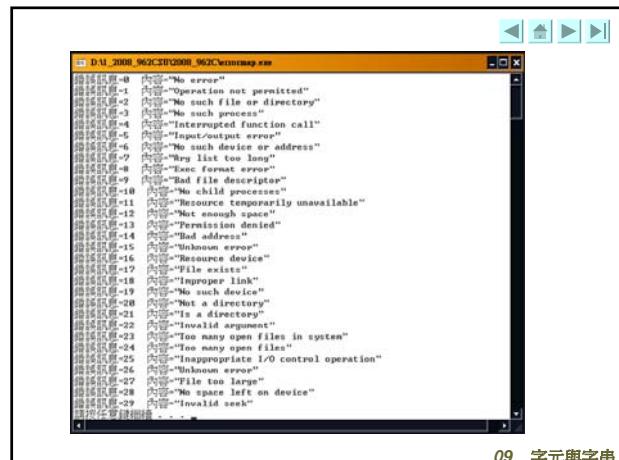
09 字元與字串

```

***** errormap.c *****/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 30
int main()
{ int i;
    for (i=0; i<SIZE; i++)
    { printf("錯誤訊息=%d 內容=%s\n", i, strerror(i));
    }
    system("pause");
    return 0;
}

```

09 字元與字串



09 字元與字串

10 格式化輸入與輸出

10.1 輸出至螢幕

10.2 從鍵盤輸入

```
#include <stdio.h>
/* function main begins program execution */

i #include <stdio.h>
i #include <math.h>
( int main(void) {
    printf(" 12345678901234567890 12345678901234567890 12345678901234567890\n");
    printf(" #工學院          \"管理學院\"      \"人文社會系\"      \n");
    printf(" 化工與材料工程系  \"國際企管系\"  \"幼兒保育系\"  ");
    printf(" 土木工程資訊系   \"企業管理系\"  \"應用外語系\"   ");
    printf(" 電機工程系       \"資訊管理系\"  \"運動健康與休閒系\" ");
    printf(" 機械工程系       \"財務金融系\"  \"共同科\"        ");
    printf(" 電競工程系       \"經營管理研究所\" \n");

    printf(" 12345678901234567890 12345678901234567890 12345678901234567890\n");
    printf(" #1-21a-21b-21c-21d-21e-21f-21g-21h-21i-21j-21k-21l-21m-21n-21o-21p-21q-21r-21s-21t-21u-21v-21w-21x-21y-21z-21a\n");
    printf(" #工學院          \"管理學院\"      \"人文社會系\"      ");
    printf(" #化學工程系      \"國際企管系\"  \"幼兒保育系\"  ");
    printf(" #土木工程資訊系 \"企業管理系\"  \"應用外語系\"   ");
    printf(" #電子工程系      \"資訊管理系\"  \"運動健康與休閒系\" ");
    printf(" #機械工程系      \"財務金融系\"  \"共同科\"        ");
    printf(" #電競工程系      \"經營管理研究所\" \n");

    printf(" 12345678901234567890 12345678901234567890 12345678901234567890\n");
    printf(" #1-21a-21b-21c-21d-21e-21f-21g-21h-21i-21j-21k-21l-21m-21n-21o-21p-21q-21r-21s-21t-21u-21v-21w-21x-21y-21z-21a\n");
    printf(" #化學工程系      \"管理學院\"      \"人文社會系\"      ");
    printf(" #土木工程資訊系 \"國際企管系\"  \"幼兒保育系\"  ");
    printf(" #電子工程系      \"企業管理系\"  \"應用外語系\"   ");
    printf(" #機械工程系      \"資訊管理系\"  \"運動健康與休閒系\" ");
    printf(" #電競工程系      \"財務金融系\"  \"共同科\"        ");
    printf(" #經營管理研究所 \"經營管理研究所\" \n");

    system("PAUSE"); // getch();
    return 0;
}
```

輸入與輸出

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

/* function main begins program execution */

int main()
{
    /* initialize variables in definitions */
    double a, b, d, e, AA, BB;
    double determinant;
    double xTemp, yTemp;
    double xResult, yResult;

    // read data
    printf(" read data from keyboard (a, b, d, e, AA, BB):");
    // scanf("%f %f %f %f %f", &a, &b, &d, &e, &AA, &BB); //wzrong
    scanf("%lf %lf %lf %lf %lf", &a, &b, &d, &e, &AA);
    printf("%lf %lf %lf %lf %lf", a, b, d, e, AA);
    // calculate determinant, xTemp, yTemp;
    determinant = (a*e - b*d);
    xTemp      = (AA * e - BB * d);
    yTemp      = (a * BB - d * AA);

    if (determinant == 0) printf( "行列式值=0 ***無解*** \n");
    else {
        printf( "x= %f\n", xTemp/determinant );
        printf( "y= %f\n", yTemp/determinant );
    }
    system("pause");
    return 0; // indicate program ended successfully */
}

/* end function main */
```

10 格式化輸入與輸出

10.1 輸出至螢幕

- `printf()` 函式可輸出精確格式化的結果
 - `printf()` 呼叫語法

```
printf(" prompt message %s... %c... %d ...", variable1, variable2,variable3);
```

格式控制字串 (format control string)

- 格式控制字串包括字串常數及轉換格式。例如：

```
printf("OK!");           /*字串常數OK*/
printf("%d", 123);      /*轉換格式%d*/
printf("整數=%d\n", 123); /*字串常數及轉換格式*/
```

10 格式化輸入與輸出

stdin, stdout, stderr

- C 言的輸入與輸出相關的函式很多，最基本的輸入函式是 `scanf()`，最基本的輸出函式是 `printf()` 函式
 - 資料流是指一連串排列的字元，ANSI C 規定資料流最少必須包含 254 個字元，包括結束的換列字元 `\n`。
 - 程式執行時會有三個資料流與程式自動結合，標準輸入資料流 (`stdin`) 與鍵盤結合，標準輸出資料流 (`stdout`) 與螢幕結合，標準錯誤資料流 (`stderr`) 與螢幕結合。

10 格式化輸入與輸出

printf() 函式格式

- 格式一：

`printf(格式控制字串);`

- 格式二：

`printf(格式控制字串, 引數列);`

10 格式化輸入與輸出

格式控制字串
format control string

% [flags] [width] [.precision] [modifiers] type

- 轉換格式的格式以百分比「%」開始，其後跟著選用的「旗標」（flags）、選用的「欄寬」（width）、選用的「精確度」（precision）、選用的「修飾字」（modifiers）及一定要標示的「轉換型態」（type）

10 格式化輸入與輸出

• 轉換格式如下（方括號[]表示選用）：

% [flags] [width] [.precision] [modifiers] type

- 最簡單的轉換格式就是「%」後跟著轉換型態
- 每一個轉換格式必須對應一個引數(argument)，引數與引數之間以逗號隔開。例如：

```
printf("%d %s", 123, "OK");
```

引數123對應轉換格式「%d」，引數 "OK" 對應轉換格式「%s」。

10 格式化輸入與輸出

10.1.1 printf() 轉換格式

type	引數型態	輸出格式
c	字元	將整數轉換為字元。
d	整數	有符號之十進位整數。
e	浮點數	格式為 [-]d.dddde[+ -]dd一個 d 表一個十進位數字。
E	浮點數	格式為 [-]d.dddddeE[+ -]dd。
f	浮點數	有符號值，格式為 [-]d.ddddd。
g	浮點數	依引數值決定輸出 f 或 e 格式。
G	浮點數	依引數值決定輸出 f 或 E 格式。
i	整數	有符號之十進位整數。
n	整數指標	輸出字元數。
o	整數	無符號之八進位整數。
p	指標	其格式依所安裝的系統而定。
s	字串指標	輸出整個字串，或指定長度之字串。
u	整數	無符號之十進位整數。
x	整數	無符號之十六進位整數，0-9,a,b,c,d,e,f。
X	整數	無符號之十六進位整數，0-9,A,B,C,D,E,F。

10 格式化輸入與輸出

printf() 轉換格式 for integer

```
***** intformats.c *****/
#include <stdio.h>
int main()
{
    printf("d轉換型態 %d\n", 365);
    printf("i轉換型態 %i\n", 365);
    printf("di轉換型態 %di\n", +365);
    printf("di轉換型態 %dn", -365);
    printf("hd轉換型態 %hd\n", 32767);
    printf("ld轉換型態 %ld\n", 1234567890);
    printf("oi轉換型態 %o\n", 365);
    printf("ui轉換型態 %u\n", 365);
    printf("ui轉換型態 %u\n", -365);
    printf("x轉換型態 %x\n", 365);
    printf("X轉換型態 %X\n", 365);
    return 0;
}
```

10 格式化輸入與輸出

printf() 轉換格式 for real number

```
***** floatformats.c *****/
#include <stdio.h>
int main()
{
    printf("e轉換型態 %e\n", 1234567.89);
    printf("e轉換型態 %e\n", 1234567.89);
    printf("e轉換型態 %e\n", -1234567.89);
    printf("E轉換型態 %E\n", 1234567.89);
    printf("f轉換型態 %f\n", 12.3456789);
    printf("g轉換型態 %g\n", 1234567.89);
    printf("G轉換型態 %G\n", 1234567.89);
    return 0;
}
```

10 格式化輸入與輸出

printf() 轉換格式 for string & character

```
***** charformat.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
    char ch = 'A';
    char *s = "Good Morning!";
    printf("這是一個'A'字元: \'%c\'\n", ch);
    printf("%s", "這是一個字串: ");
    printf("\n%s\n", s);
    return 0;
}
```

10 格式化輸入與輸出

```

/************* ptrformat.c *****/
#include <stdio.h>
int main()
{
    int *iPtr;
    int m = 12345, n;
    iPtr = &m;
    printf("整數指標變數iPtr的內含值= 0x%p\n", iPtr);
    printf("變數m的位址= 0x%p\n", &m);
    printf("本列所印出來的字元數=%n", &n);
    printf("%d\n", n);
    n = printf("abcdef");
    printf("印出 %d 個字元\n", n);
    printf("在格式控制字串中列印百分比符號 %%\n");
    return 0;
}

```

10 格式化輸入與輸出

printf() 轉換格式 :width for real number or integer

```

/************* width.c *****/
#include <stdio.h>
int main()
{
    printf("欄寬4,轉換型態d=%4d\n", 1);
    printf("欄寬4,轉換型態d=%4d\n", 123);
    printf("欄寬4,轉換型態d=%4d\n", 1234);
    printf("欄寬4,轉換型態d=%4d\n", 12345);
    printf("欄寬4,轉換型態d=%4d\n", -12345);
    printf("欄寬4,轉換型態d=%4d\n", -1234);
    printf("欄寬4,轉換型態d=%4d\n", -123);
    printf("欄寬4,轉換型態d=%4d\n", -12);
    printf("欄寬4,轉換型態d=%4d\n", -1);
    return 0;
}

```

10 格式化輸入與輸出

10.1.6 精確度之使用

- 精確度與整數轉換型態一起使用
 - 表示最少必須印出指定的位數，如果印出來的位數小於指定的精確度，則多出來的位數會補上零，整數的精確度預設為1。例如下列程式使用「%.4d」。
- 精確度與轉換型態 e、E或f一起使用
 - 表示小數點後必須印出指定的位數。
- 精確度與轉換型態g或G一起使用
 - 表示印出指定的最大有效位數。
- 精確度與轉換型態s一起使用
 - 表示最多印出指定的字元。
- 精確度可與欄寬一起使用
 - 例如 printf("%10.2f",123.456) 印出欄寬十位其中兩位小數。

10 格式化輸入與輸出

printf() 轉換格式 :precision

```

/************* precision.c *****/
#include <stdio.h>
int main()
{
    int i = 365;
    float f = 123.78567;
    char *s = "Hello";
    printf("%d 使用整數精確度\n", i);
    printf(" 精確度.4d\n%.4d 精確度.9d \t%.9d\n", i, i);
    printf("%f 使用浮點數精確度\n", f);
    printf(" 精確度.3f \t%.3f 精確度.3e \t%.3e\n", f, f);
    printf(" 精確度.3g \t%.3g\n", f);
    printf("\">%s\" 使用字串精確度\n", s);
    printf(" 精確度11s \t%.11s\n", s);
    return 0;
}

```

10 格式化輸入與輸出

10.1.7 旗標之使用

- printf() 函式也提供旗標 (flags) 以補功能的不足，旗標有「-」、「+」、「_」及「#」等四種，其用法如下表所示。

flags	說明
-	左邊對齊，右邊以空白填入。
	預設為右邊對齊。
+	數字必須印出加號 +，或減號 -。
空白	正數或零之符號位輸出空白，負數之符號位輸出負號。
#	與數值轉換型態 (type) 一起使用。

10 格式化輸入與輸出

【程式flags.c】

```

/************* flags.c *****/
#include <stdio.h>
int main()
{
    printf("     1   2   3   4 \n");
    printf("12345678901234567890123456789012\n");
    printf("-----\n");
    printf("%10s%10d%10c%10f\n", "string", 456, 'C', 1.23);
    printf("%-10s%-10d%-10c%-10f\n", "string", 456, 'C', 1.23);
    return 0;
}

```

【執行】

```

gcc flags.c -o flags.exe <Enter>
flags.exe <Enter>
     1   2   3   4
12345678901234567890123456789012
-----
string 456      C  1.230000
string 456      C  1.230000

```

10 格式化輸入與輸出

【程式plusflag.c】

```
***** plusflag.c *****/
#include <stdio.h>
int main()
{
    printf("沒用+旗標d \t%ld\t%ld\n", 987, -987);
    printf("使用+旗標+d \t%+ld\t%+ld\n", 987, -987);
    return 0;
}
```

【執行】

```
gcc plusflag.c -o plusflag.exe <Enter>
plusflag.exe <Enter>
沒用+旗標d 987 -987
使用+旗標+d +987 -987
```

10 格式化輸入與輸出

printf() 轉換格式 : number system for integer

【程式alternate.c】

```
***** alternate.c *****
#include <stdio.h>
int main()
{
    int i = 365;
    float f = 365.25;
    printf(" 365使用#旗標#o %#o\n", i);
    printf(" 365使用#旗標#x %#x\n", i);
    printf(" 365使用#旗標#X %#X\n", i);
    printf("365.25沒用#旗標g %g\n", f);
    printf("365.25使用#旗標#g %#g\n", f);
    return 0;
}
```

【執行】
alternate.exe <Enter>

```
365使用#旗標#o 0 0555
365使用#旗標#x 0x16d
365使用#旗標#X 0X16D
365.25沒用#旗標g 365.25
365.25使用#旗標#g 365.250
```

10 格式化輸入與輸出

printf() 轉換格式 : leading zero

【程式zeroflag.c】

```
***** zeroflag.c *****/
#include <stdio.h>
int main()
{
    printf("123使用0旗標+09d\n%+09d\n", 123);
    printf("123使用0旗標09d\n%09d", 123);
    return 0;
}
```

【執行】

```
gcc zeroflag.c -o zeroflag.exe <Enter>
zeroflag.exe <Enter>
123使用0旗標+09d
+00000123
123使用0旗標09d
00000123
```

10 格式化輸入與輸出

10.1.8 變動的欄寬與精確度

- 使用星號「*」來指定變動的欄寬及精確度。
- Program assignment during execution: 相對應的引數值先被計算出來，然後取代相對應的星號。若計算出來的欄寬值是負數，表示左邊對齊，但精確度計值必須是正整數。例如：

```
printf("%*.1f", 12, 3, 123.456);
```

- 表示欄寬12，小數3位，左邊對齊，印出“123.456”。

```
printf("%*.1f", 12, 3, -987.654);
```

- 表示欄寬12，小數3位，右邊對齊，印出“ -987.654”。

10 格式化輸入與輸出

***** asterisk.c *****

```
#include <stdio.h>
int main()
{
    int w, n=12345;
    float f=123.456;
    for (w=7; w<=9; w++)
    {
        printf("欄寬w=%-6d", w);
        printf(" 整數n=%*d ", w, n);
        printf("\t浮點數f=%.*f\n", w, f);
    }
    return 0;
}
```

【執行】

```
gcc asterisk.c -o asterisk.exe <Enter>
asterisk.exe <Enter>
欄寬w=7 整數n= 12345 浮點數f=123.456
欄寬w=8 整數n= 12345 浮點數f= 123.456
欄寬w=9 整數n= 12345 浮點數f= 123.456
```

10 格式化輸入與輸出

10.2 從鍵盤輸入

精確的格式化輸入呼叫函式scanf() 函式的語法

```
scanf("%s %c %d", &variable1, &variable2,&variable3);
```

格式控制字串 (format control string)

- 格式控制字串說明輸入資料的格式，引數列由逗號隔開的引數所組成。
- 每一個引數均為指向變數的指標(該記憶體位址用來儲存輸入的資料)。
- 每一個引數均對應格式控制字串中一個轉換型態。
- 沒有prompt message**

10 格式化輸入與輸出

◀ ▶ ⟲ ⟳

10.2.1 scanf 轉換格式

- scanf() 函式的格式控制字串

% [*] [width] [modifiers] type

- 星號「*」表示要取消下一輸入欄之轉換格式，
width（欄寬）表示輸入最大之字元數，
modifiers（修飾字）包含「h」、「l」及「L」，
h表引數的資料型態為「short int」，
l表引數資料型態為「long int」或double，
L表引數的資料型態為「long double」。

10 格式化輸入與輸出

◀ ▶ ⟲ ⟳

表 10.5 scanf 轉換型態與引數資料型態表

type	輸入資料型態	引數資料型態
c	字元	字元指標。
d	十進位整數	整數指標。
e	浮點數	浮點數指標。
E	浮點數	浮點數指標。
f	浮點數	浮點數指標。
g	浮點數	浮點數指標。
G	浮點數	浮點數指標。
i	十、八、十六進位整數	整數指標。
n	整數指標。	
o	八進位整數	整數指標。
p	十六進位格式	依記憶體模式轉換為指標格式。
s	字串	字元陣列指標。
u	無號十進位整數	無號整數指標。
x	十六進位整數	整數指標。
X	十六進位整數	整數指標。
%	% 字元	跳過 % 字元。
I	字元集合	只讀入方括號內指定之字元。

10 格式化輸入與輸出

◀ ▶ ⟲ ⟳

Read integer data

```
***** intscan.c *****/
#include <stdio.h>
int main()
{
    int a, b, c, d, e, f, g;
    printf("請以%%d格式輸入一個整數a: ");
    scanf("%d", &a);
    printf("請以%%o格式輸入一個整數b: ");
    scanf("%o", &b);
    printf("請以%%x格式輸入一個整數c: ");
    scanf("%x", &c);
    printf("請以%%u格式輸入一個整數d: ");
    scanf("%u", &d);
    printf("您所輸入的整數 a,b,c,d : \n");
    printf("a=%d b=%d c=%d d=%d ", a, b, c, d);
    return 0;
}
```

10 格式化輸入與輸出

◀ ▶ ⟲ ⟳

Read real data

```
***** floatscan.c *****/
#include <stdio.h>
int main()
{
    float a, b, c;
    printf("請以%%e格式輸入一個浮點數a: ");
    scanf("%e", &a);
    printf("請以%%f格式輸入一個浮點數b: ");
    scanf("%f", &b);
    printf("請以%%g格式輸入一個浮點數c: ");
    scanf("%g", &c);
    printf("您所輸入的浮點數 a,b,c : \n");
    printf("a=%f b=%f c=%f\n", a, b, c);
    return 0;
}
```

10 格式化輸入與輸出

◀ ▶ ⟲ ⟳

Read character or string

```
***** charscan.c *****/
#include <stdio.h>
int main()
{
    char ch, s[80];
    printf("以%%c格式輸入一個字元ch: ");
    scanf("%c", &ch);
    printf("以%%s格式輸入一個字串 s: ");
    scanf("%s", s);
    printf("您所輸入的字元=%c\n", ch);
    printf("您所輸入的字串=%s\n", s);
    return 0;
}
```

10 格式化輸入與輸出

◀ ▶ ⟲ ⟳

Read a special string (1)

【程式 scanset.c】

```
***** scanset.c *****/
#include <stdio.h>
int main()
{
    char s[9];
    printf("請依%%[aeiou]格式輸入一個字串s: ");
    scanf( "%[aeiou]", s);
    printf("您所輸入的字串=%s\n", s);
    return 0;
}
```

【執行】

```
gcc scanset.c -o scanset.exe <Enter>
scanset.exe <Enter>
請依%%[aeiou]格式輸入一個字串s: aiov! <Enter> 您所輸入
的字串="aiov"
```

10 格式化輸入與輸出

Read a special string (2)

```
【程式iscanset.c】
/*****iscanset.c *****/
#include <stdio.h>
int main()
{
    char s[9];
    printf("請依%[^aeiou]格式輸入一個字符串: ");
    scanf( "%[^aeiou]", s);
    printf("您所輸入的字符串=%s\n", s);
    return 0;
}
```

【執行】

```
gcc iscanset.c -o iscanset.exe <Enter>
iscanset.exe <Enter>
請依%[^aeiou]格式輸入一個字符串: string <Enter>
您所輸入的字符串="str"
```

10 格式化輸入與輸出

Be careful in data entry (1)

```
【程式widthscan.c】
/*****widthscan.c *****/
#include <stdio.h>
int main()
{
    int a, b;
    printf("請輸入七個阿拉伯數字: ");
    scanf("%3d%4d", &a, &b);
    printf("整數變數a=%d 整數變數b=%d\n", a,b);
    return 0;
}
```

【執行】

```
gcc widthscan.c -o widthscan.exe <Enter>
widthscan.exe <Enter>
請輸入七個阿拉伯數字: 1234567 <Enter>
整數變數a=123 整數變數b=4567
```

10 格式化輸入與輸出

Be careful in data entry (2)

```
***** suppress.c *****/
#include <stdio.h>
int main()
{
    int mm1, dd1, yy1, mm2, dd2, yy2;
    printf("請輸入以mm-dd-yy格式的日期: ");
    scanf("%d%c%d%c%d", &mm1, &dd1, &yy1);
    printf("月=%d 日=%d 年=%d\n", mm1, dd1, yy1);
    printf("請輸入以mm/dd/yy格式的日期: ");
    scanf("%d%c%d%c%d", &mm2, &dd2, &yy2);
    printf("月=%d 日=%d 年=%d\n", mm2, dd2, yy2);
    return 0;
}
```

```
suppress.exe <Enter>
請輸入以mm-dd-yy格式的日期: 1-26-2007 <Enter>
月=1 日=26 年=2007
請輸入以mm/dd/yy格式的日期: 1/26/2007 <Enter>
月=1 日=26 年=2007
```

10 格式化輸入與輸出



11 結構

- [11.1 結構定義](#)
- [11.2 顯示結構成員](#)
- [11.3 從鍵盤輸入至結構成員](#)
- [11.4 結構變數輸出至檔案](#)
- [11.5 串列 list](#)
- [11.6 同位 union](#)
- [11.7 位元欄位](#)
- [11.8 列舉型態 enum](#)
- [11.9 型態定義 typedef](#)

www.caneis.com.tw

結構

- 結構在高階的物件導向的程式中，是一個極其重要的工具。
- 傳統的資料型態僅適用於個別的單純資料。
- 傳統的資料型態結合陣列只是將相同型態的資料結合起來。
- **結構 (struct)** 可將不同資料型態及相同型態的資料結合起來。
- 結構可以用來定義儲存在檔案裡的記錄，建構資料庫系統；結構配合指標可建立複雜的資料結構，如連結串列、佇列、堆疊、及樹結構等。

11 結構

11.1 結構定義

- 結構是一種衍生的資料型態，以關鍵字 **struct** 開始，接著為結構的標記(也就是結構的名稱)，其後以大括號括起來的是結構的成員，每一個成員皆為指定型態的變數。

```
    結構的標記
struct person
{ char name[12];      結構的成員1
  int age;             結構的成員2
};
```

• 宣告以**person**結構體為組成的結構變數**man1, man2**

```
struct person man1, man2;
```

11 結構

struct結構體

圖 10-1 一筆資料可能需要使用多種資料型態

學號	數學	英文	計概	平均
"59103501"	89	84	75	82.67
"59103502"	77	69	87	77.67
"59103503"	65	68	77	70.00

排序	學號	計概	數學	英文	平均	排序	學號	計概	數學	英文	平均
	"59103501"	"65"	"84"	"75"	"82.67"		"59103503"	"65"	"68"	"77"	"70.00"
	"59103502"	"77"	"69"	"87"	"77.67"		"59103502"	"77"	"69"	"87"	"77.67"
	"59103503"	"89"	"68"	"77"	"70.00"		"59103501"	"89"	"84"	"75"	"82.67"

陳錦輝，C/C++初學指引，金禾書局(R501)。

struct結構體：定義

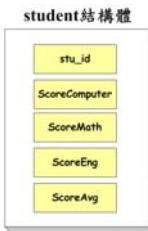
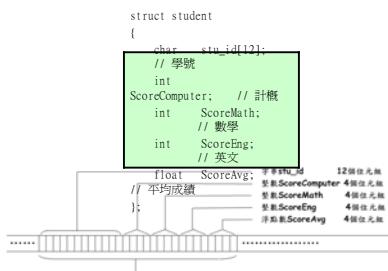


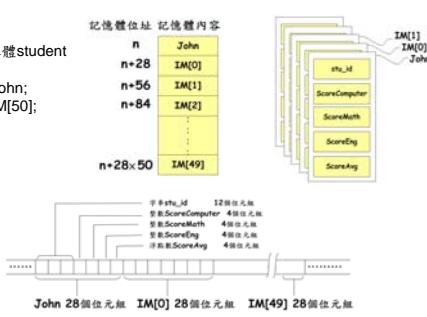
圖 10-3 結構體示意圖

陳錦輝，C/C++初學指引，金禾書局(R501)。

struct結構體

延續前面的結構體student

```
struct student John;
struct student IM[50];
```



John 28 個位元組 IM[0] 28 個位元組 IM[49] 28 個位元組

圖 10-5 結構體變數與陣列在記憶體中的狀況

陳錦輝，C/C++初學指引，金禾書局(R501)。

宣告結構變數同時賦予初值

- 宣告結構變數的同時也可以賦予初值的，例如：

```
struct person man = { "KayLin", 6 };
```

- 以句點(.)稱呼結構中的成員，其格式如下：

```
結構變數名稱 . 成員變數名稱
```

- **此時**，結構變數名稱為man的成員變數name及age的內容分別為

```
man.name = "KayLin";
man.age = 6;
```

11 結構

透過結構指標變數取得成員變數

- 透過結構指標變數以取得成員變數，其格式如下：

```
結構指標變數 -> 成員變數名稱
```

- 如下例：

```
struct person *p = &man;
```

- 相當於

```
p->name = "KayLin";
p->age = 6;
```

11 結構

結構體範例

```
***** person.c *****
#include <stdio.h>
struct person
{
    char name[12];
    int age;
};
int main()
{
    struct person man = {"KayLin", 6};
    struct person *p;
    printf("person結構變數man表示法\n");
    printf(" man.name=%s\n", man.name);
    printf(" man.age=%d\n", man.age);
    p = &man;
    printf("person結構指標*p表示法\n");
    printf(" p->name=%s\n", p->name);
    printf(" p->age=%d\n", p->age);
    return 0;
}
```

11 結構

person 結構成員

- person結構，包含兩個成員，
 - 第一個為字元陣列name，含12個字元，
 - 第二個成員為整數變數age。
- 宣告一個person結構型態的變數man，它的成員初值以大括號表示，name成員的初值為“Kay”，age成員的初值為6。age佔四個位元組，加上name佔十二個位元組，所以man變數在記憶體裡佔十六個位元組。

11 結構

person 結構成員(續)

- 程式中宣告一個person結構的指標變數p。使用變數man中成員的表示法，即man.name及man.age，並將它們的值顯示出來。將person結構指標變數p的內含值設定為man的位址，如下圖所示。

11 結構

11.3 從鍵盤輸入至結構成員

```
***** keyinperson.c *****
#include <stdio.h>
struct person
{
    char name[12];
    int age;
};
int main()
{
    struct person man, *p;
    printf("請輸入姓名：");
    scanf("%s", man.name);
    printf("請輸入年齡：");
    scanf("%d", &man.age);
    p = &man;
    printf("person結構指標*p表示法\n");
    printf(" 輸入姓名=%s\n", p->name);
    printf(" 輸入年齡=%d\n", p->age);
    return 0;
}
```

How about if **data type struct** is applied upon a group data **like array**?

11 結構

11.4 結構變數輸出至檔案(1)

```
// file name: outperson.c
struct person
{
    char name[12];
    int age;
};

int main()
{
    char buf[256];
    struct person man, *p;
    FILE *f = fopen("person.dat", "w");
    printf("請逐筆輸入姓名、年齡(直接按Ctrl-Z結束): \n");
    while (gets(buf)!=NULL)
    {
        memset(man.name, '\0', sizeof(man.name));
        sscanf(buf, "%s %d", &man.name, &man.age);
        fseek(f, 0L, SEEK_END);
        fwrite(&man, sizeof(man), 1, f);
    }
    fclose(f);
    return 0;
}
```

1 結構

11.4 結構變數輸出至檔案(2)

- 程式outperson.c將從鍵盤輸入的man值輸出至檔案person.dat。宣告一個FILE檔案指標f，呼叫fopen()函式開啓一個person.dat輸出檔，開啓失敗時傳回NULL值，開啓成功時傳回一個該檔案的指標f。
- 將man結構變數中的name成員值清除為'\0'，這一列不執行其實沒有關係的，執行的話資料中字串剩餘的空間就不會有其他的雜值。

11 結構

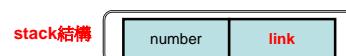
結構變數輸出至檔案(3)

- 使用gets(buf)重複輸入姓名及年齡至記憶體緩衝器buf，直到您直接按Enter鍵才停止輸入。這時buf字串的長度為零。
- 找到檔案尾端，將輸入後的man結構變數值輸出至person.dat檔案的尾端，姓名若輸入"-1"則結束輸入，接著將person.dat檔案關閉而結束程式的執行。
- 本例題重點在介紹結構之應用，將整個結構變數(記錄)輸出至檔案。

11 結構

11.5 串列(list)

- 結構裡的成員若是屬於結構的指標，這種指標指向另一個結構，可構成一個連結串列。
- 連結串列中的每一個節點都是同一種結構。



- stack結構中第一個為整數成員變數number，第二個為結構stack的指標變數link。成員link的內含值可為NULL表示沒有連結至其他的結構，若不為NULL則表示連結至其他的結構。

11 結構

結構連結串列(續)

```
struct stack
{
    int number;
    struct stack *link;
} node, *p, *top;
```

- node為stack結構的變數，node.number參考到number成員，node.link參考到link成員。p及top均為stack結構的指標變數，這三個變數node、p及top均沒給予初值。

11 結構

```
***** stack.c *****
```

```
#include <stdio.h>
#include <stdlib.h>
struct stack
{
    int number;
    struct stack *link;
} node, *p, *top;

int main()
{
    int i;
    top = malloc(sizeof(node));
    top->link = NULL;
    for (i=0; i<3; i++)
    {
        p = malloc(sizeof(node));
        printf("請輸入一個整數: ");
        scanf("%d", &p->number);
        p->link = top->link;
        top->link = p;
    }
    p = top->link;
    while (p != NULL)
    {
        printf("%d ", p->number);
        p = p->link;
    }
    return 0;
}
```

11 結構

• 程式stack.c呼叫 malloc() 函數取得一塊node大小的記憶體，其位址存入stack結構的指標變數top中，設定top所指結構的成員link的內含值為NULL，表示top節點（node）沒有連結至其他的節點。

11 結構

• 呼叫 malloc() 函數取得一塊node大小的記憶體，其位址存入stack結構的指標變數p中，輸入一個整數111至p所指結構的成員number，設定p所指結構的成員link的內含值為top所指結構成員link的內含值，將top指標移到p所指的節點。

11 結構

- 輸入222至p所指結構的成員number，插入p節點至top節點的後面，如下圖所示。
- 輸入333至p所指結構的成員number，插入p節點至top節點的後面，如下圖所示。

11 結構

11.6 同位 union

- 同位（union）也是種衍生的資料型態
- 同位的成員佔相同的記憶體空間
 - 程式中有些變數互相之間沒有關聯，有排他性，因此讓它們共用記憶體空間，為了節省記憶體，
 - 當記憶體已經很便宜，記憶體擴充很方便，節省記憶體已經不是那麼重要，
 - 不過在單晶片程式設計，仍然有需求，我們還是要了解同位的原理，不然別人設計的程式您可能就看不懂了。

11 結構

11.6.4 union宣告

```
union uTag
{
    int i;
    float f;
} u;
```

宣告一個同位變數 u。成員u. i(整數變數)及成員u. F(浮點數變數)佔共同的記憶體空間。

11 結構

```
int main()
{
    union uTag u;
    u.i = 365;
    printf("將整數365存入union uTag型態同位變數u中\n");
    printf("整數u.i的值=%d 浮點數u.f的值=%f\n", u.i, u.f);
    u.f = 365.0;
    printf("將365.0存入union uTag型態同位變數u中\n");
    printf("整數u.i的值=%d 浮點數u.f的值=%f\n", u.i, u.f);
    return 0;
}
```

int	31 30 23 22	0
	± 數值	

float	± 指數	尾數
	63 62 52 51	0

double	± 指數	尾數
	63 62 52 51	0

11 結構

【執行】
uniontest.exe <Enter>
將整數365存入union uTag型態同位變數u中
整數u.i的值=365 浮點數u.f的值=0.000000
將365.0存入union uTag型態同位變數u中
整數u.i的值=1136033792 浮點數u.f的值=365.000000

將 u.f 設定為 365.0，整數變數i佔共同的記憶體空間，因此將下列資料：
43 B6 80 00
以整數的儲存格式加於解釋，您將它以「%d」格式列印，結果就是1136033792。

11 結構

11.7 位元欄位 bit field

- ANSI C 語言提供位元欄位（bit field）的功能，這種功能讓您可以在結構struct及同位union的unsigned int或int型態成員指定位元個數。
- 位元欄位（bit field）的功能除了節省記憶體之外，也提高了執行效率，**位元欄位的成員必須宣告為int或unsigned**。

11 結構

```
***** bitfield.c *****
#include <stdio.h>
struct cardtag
{ unsigned suit : 2;
  unsigned point : 4;
};

*Suit(牌組)有四種：
黑桃(0)、紅心(1)、方塊(2)、梅花(3)：占2位元。
*Point(牌點)共有13種牌點占4個位元。

當牌組為 3，牌點為12，
```

結果列印 "梅花K"。

11 結構

11.8 列舉型態 enum

- enum列舉型態是ANSI C 提供給使用者自定資料內容(識別標籤;tag)的合法使用集合

enum etag { North, East, South, West } e1, e2;
North, East, South, West值分別為0至3，而變數e1, e2 e的值只能為0, 1, 2, 3中的一個。

enum { North=3, East, South=7, West } e;
North、East、South、West的值分別為3,4,7,8，而變數e的值只能為3,4,7,8中的一個。

11 結構

【程式enumtest.c】

```
***** enumtest.c *****
#include <stdio.h>
enum etag { North, East, South, West } e2;
int main()
{
    char enames[][3] = { "北", "東", "南", "西" };
    for (e2=North; e2<=West; e2++)
        printf("%2d%11s\n", e2, enames[e2]);
    return 0;
}
```

【執行】
enumtest.exe <Enter>

0	北
1	東
2	南
3	西

11 結構

【程式enumweek.c】

```
***** enumweek.c *****
#include <stdio.h>
enum enumWeek {Mon=1, Tue, Wen, Thu, Fri, Sat, Sun};
int main()
{
    enum enumWeek week;
    char weekName[][10] =
    {
        "", "Monday", "Tuesday", "Wendesday",
        "Thursday", "Friday", "Saturday", "Sunday"
    };
    for (week=Mon; week<=Sun; week++)
        printf("%2d%11s\n", week, weekName[week]);
    return 0;
}
```

【執行】
enumweek.exe ←

1	Monday
2	Tuesday
3	Wendesday
4	Thursday
5	Friday
6	Saturday
7	Sunday

11 結構

11.9 型態定義 `typedef`

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

typedef int INTEGER ;
typedef float AVERAGE ;

int main()
{ INTEGER count,total=0;
  AVERAGE number;
  for (total=count=0; count<=100; count++)
    total+=count;
  count--;
  number= (AVERAGE ) total /count;
  printf("total= %d; average= %.4f\n", total, number);
  system("pause");
  return 0;
}
```

12 檔案處理

12.1 循序檔之結構

12.2 開啓循序檔

12.3 關閉循序檔

12.4 feof()函式

12.5 資料輸出至循序檔

12.6 資料附加至循序檔

12.7 資料由循序檔讀取

12.8 二進位檔

12.9 隨機檔



http://en.wikipedia.org/wiki/Image:Hard_disk_platter_reflection.jpg



- 儲存在電腦記憶體中的變數或陣列裡的資料是暫時的，當程式執行完成或電腦關機後，這些資料就消失無蹤了。
- 檔存於磁片、磁碟、光碟、隨身碟的檔案可用來永久保存資料。
- 在ANSI C語言中，檔案區分成
 - (A)循序檔或隨機存取檔
 - (B)本文檔或二進位檔

12 檔案處理

12.1 循序檔之結構

- ANSI C將每個檔案看成一個由位元組所組成的循序資料流。每一個檔案均以一個檔案結束符號結尾，這個符號簡稱為EOF。
- 不同的作業系統有不同的EOF符號，如下表所示。

表 12.1 作業系統與EOF符號表

作業系統	按鍵組合
UNIX/Linux 系統	<return><ctrl>d
Windows 系統	<ctrl>z
Macintosh 系統	<ctrl>d
VAX (VMS)	<ctrl>z

12 檔案處理

資料流(stream)

- 當一個檔案開啓後，便有一個資料流與此檔案結合在一起。
- 開啓一個檔案會傳回一個指向FILE結構的指標，這個FILE結構定義於表頭檔stdio.h裡，FILE結構其成員依C語言編譯器的不同而異。
- 當程式開始執行時，便有三個檔案及其相結合的資料流會自動的被開啓，這三個分別是標準輸入資料流（stdin）、標準輸出資料流（stdout）及標準錯誤資料流（stderr）。
- 資料流提供檔案與程式間的通訊管道。例如標準輸入資料流讓程式能從鍵盤讀取資料，標準輸出資料流及標準錯誤資料流能將資料顯示在螢幕上。

12 檔案處理

【程式file2.c】

```
***** filetest.c *****  
#include <stdio.h>  
#include <string.h>  
int main()  
{  
    char buf[80], ptr[80];  
    FILE *f=fopen("kay.txt", "r");  
    fgets(buf,sizeof(buf),f);  
    printf("讀入kay.txt檔案第一筆資料 : \n%s",buf);  
    strcpy(ptr,f->_ptr,4);  
    ptr[4]='\0';  
    printf("檔案指標指到下一筆資料 _ptr=%s\n", ptr);  
    printf("所使用緩衝器大小為 _bufsiz=%d\n", f->_bufsiz);  
    fclose(f);  
    return 0;  
}
```

12 檔案處理

【檔案kay.txt】

```
1001 FayChen 10000  
1002 T.C.Wang 20000  
1003 KayLin 3000
```

【執行】

```
gcc filetest.c -o filetest.exe <Enter>  
filetest.exe <Enter>  
讀入kay.txt檔案第一筆資料：  
1001 Jemmy 10000  
檔案指標指到下一筆資料 _ptr="1002"  
所使用緩衝器大小為 _bufsiz=4096
```

12 檔案處理

12.2 開啓循序檔

- **表 12.2 開啓檔案模式mode**

mode	說明
r	開啓檔案為唯讀。
w	開啓檔案為建立輸出檔。
a	開啓檔案為附加檔。
r+	開啓檔案為更新，讀及寫。
w+	開啓檔案為建立新檔並更新。
a+	開啓檔案為附加更新。

- 模式mode之後附加“t”表示本文檔，附加“b”表示二進位檔。

12 檔案處理

1. fopen("fileWrt.dat", "w");

開啓一循序檔fileWrt.dat供輸出之用，若該檔不存在，則產生一個新的資料檔，並將檔案指標移至新資料檔fileWrt.dat的起始位置。若該檔為既有的檔案，則將檔案指標移至原檔案的起始位置，並將原檔案的內容拋棄，使用時要特別注意。

2. fopen("fileApndx.dat", "a");

開啓一循序檔fileApndx.dat供附加之用，若該檔不存在，則產生一個新的資料檔，並將檔案指標移至新資料檔fileApndx.dat的起始位置。若該檔為既有的檔案，則將檔案指標移至原檔案之結束位置（最後一筆資料的後面），以備新資料逐筆附加。注意，原檔案資料仍然保留著，只是新增資料附加其後而已。

3. fopen("fileRD.dat", "r");

開啓一循序檔fileRD.dat供輸入之用，若該檔不存在，則執行時會產生錯誤。若該檔存在，則檔案指標移至fileRD.dat的起始位置。

12 檔案處理

12.3 關閉循序檔

fclose()函式可用來關閉循序檔
fclose()定義在stdio.h表頭檔中

fclose() 語法如下：

```
int fclose(FILE *stream);
```

關閉stream所指之檔案。
成功時傳回0值，失敗時傳回EOF值。

12 檔案處理

12.4 feof()函式

feof()函式用來檢查檔案是否已經結束。
feof()函式定義在stdio.h表頭檔中

feof() 語法如下：

```
int feof(FILE *stream);
```

檢查stream所指檔案是否已到檔案尾。
是檔案尾則傳回非0，否則傳回0值。

12 檔案處理

12.5 資料輸出至循序檔(new)

資料輸出至循序檔的四個基本過程：

1. 宣告一個 FILE 結構的指標變數
`FILE *fPtr;`
2. 以 fopen() 函式開啓指定的檔案，如 outfile.txt 為輸出。
`fPtr=fopen("outfile.txt", "w");`
3. 以 fprintf() 函式將資料輸出至指定之檔案。
`fprintf(fPtr, 格式字串, 資料);`
4. 以 fclose() 關閉指定之檔案，如下：
`fclose(fPtr);`

12 檔案處理

```
***** account.c *****/
#include <stdio.h>
int main()
{
    char line[80];
    int account;
    char name[12];
    float amount;
    FILE *f = fopen("account.txt", "w");
    printf("請輸入帳號、姓名、存款\n");
    while (1)
    {
        if(gets(line)==NULL) break;
        sscanf(line,"%d %s %f", &account, &name, &amount);
        fprintf(f, "%d %s %.2f\n", account, name, amount);
    }
    fclose(f);
    return 0;
}
```

12 檔案處理

```

***** fixout.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
    char rec[25];
    FILE *f=fopen("fixout.txt", "w");
    printf("請輸入定長整筆記錄,Ctrl+Z結束 :\n");
    printf("123456789012345678901234\n");
    while (1)
    {
        if ( gets(rec)==NULL ) break;
        fprintf(f, "%s\n", rec);
    }
    fclose(f);
    return 0;
}

```

12 檔案處理



12.6 資料附加至循序檔

資料附加至循序檔的四個基本過程：

1. 壓告一個 FILE 結構的指標變數，如下：

FILE *fPtr;

2. 以 fopen() 函式開啓檔案為附加，如下：

fPtr=fopen("account3.dat", "a");

3. 以 fprintf() 函式將資料輸出至指定之檔案，如下：

fprintf(fPtr, "%s\n", rec);

4. 以 fclose() 函式關閉指定之檔案，如下：

fclose(fPtr);

12 檔案處理

```

***** append.c *****/
#include <stdio.h>
#include <string.h>
int main()
{
    char rec[25];
    FILE *f=fopen("fixout.txt", "a");
    printf("請輸入定長整筆記錄,Ctrl+Z結束 :\n");
    printf("123456789012345678901234\n");
    while (1)
    {
        if ( gets(rec)==NULL ) break;
        fprintf(f, "%s\n", rec);
    }
    fclose(f);
    return 0;
}

```

12 檔案處理



12.7 資料由循序檔讀取

資料由循序檔讀取的四個基本過程

1. 壓告一個 FILE 結構的指標變數，如下：

FILE *fPtr;

2. 以 fopen() 函式開啓檔案為輸入，如下：

fPtr=fopen("fixout.txt", "r");

3. 以 fscanf() 函式將資料由指定之檔案輸入，如下例：

fscanf(fPtr, "%d %s %f", &account, name, &amount);

4. 以 fclose() 函式關閉指定之檔案，如下：

fclose(fPtr);

12 檔案處理

```

***** readline.c *****/
#include <stdio.h>
int main()
{
    int account;
    char name[12];
    float amount;
    char line[80];
    FILE *f=fopen("fixout.txt", "r");
    while ( fgets(line,sizeof(line), f)!=NULL )
    {
        sscanf(line,"%d %s %f",&account,&name,&amount);
        printf("%d %s %.2f\n", account, name, amount);
    }
    fclose(f);
    return 0;
}

```

12 檔案處理

```

***** fixin.c *****/
#include <stdio.h>
int main()
{
    int account;
    char name[12];
    float amount;
    FILE *f=fopen("fixout.txt", "r");
    while (1)
    {
        fscanf(f, "%4d%12s%f",&account,&name,&amount);
        if ( feof(f) ) break;
        printf("%d %s %.2f\n", account, name, amount);
    }
    fclose(f);
    return 0;
}

```

12 檔案處理

◀ ▶ ⟲ ⟳

12.8 二進位檔

- 二進位檔與本文檔不同。**
- Windows作業系統的本文檔每一記錄之後均有返回字元（Carriage Return 簡稱CR）及換列字元（Line Feed 簡稱LF），CR以十六進位表示為0x0d，LF以十六進位表示為0x0a。
- C程式輸入本文檔時將CR/LF轉換為只有LF字元，輸出本文檔時又將LF轉換為CR/LF兩個字元。**
- 二進位檔之輸入及輸出都沒有這種轉換的動作。**

12 檔案處理

◀ ▶ ⟲ ⟳

12.8 二進位檔mode

- 二進位檔使用 fopen()開檔時，模式mode後附加“b”

表 12.3 開啓二進位檔案模式mode表

mode	說明
rb	開啓二進位檔為唯讀。
wb	開啓二進位檔為建立輸出檔。
ab	開啓二進位檔為附加。
rb+	開啓二進位檔為更新、讀入及寫出。
wb+	開啓二進位檔為建立新檔並更新。
ab+	開啓二進位檔為附加更新。

12 檔案處理

◀ ▶ ⟲ ⟳

12.8 二進位檔存取

- fread()函式從二進位檔讀入資料至變數**

```
size_t fread(void *ptr, size_t size, size_t n,
FILE *stream)
```

由指定檔stream輸入n項資料，每一項為size個位元組，輸入至ptr指標所指之記憶體位址。成功時傳回n值，失敗或讀至檔尾時傳回0值。

- fwrite()函式將變數的內容寫入至二進位檔案**

```
size_t fwrite(const void *ptr, size_t size, size_t n,
FILE *stream);
```

將ptr指標所指處之n項資料，每項size個位元組，共n*size個位元組寫入stream所指的檔案。成功時傳回寫入項數，失敗時傳回短缺之項數。

12 檔案處理

◀ ▶ ⟲ ⟳

12.8 二進位檔的資料特性

- 二進位檔可看成一個位元組循序的集合體，
- 記錄是二進位檔基本儲存資料單元，
- 記錄本身嚴格來說可以是一個位元組，也可以是多個位元組，
- 二進位存取不僅可讀寫ASCII碼檔，也可讀寫可執行檔(.exe檔)或目的碼檔(.obj檔)，**
- 二進位存取對一個已開啓之檔可往前或往後讀寫，**

12 檔案處理

◀ ▶ ⟲ ⟳

Position the file pointer

```
Int fseek(FILE *stream, long offset, int whence)
```

將指定檔案stream的檔案指標移動至偏離檔案指標whence處offset個位元組之位置。定位成功時傳回0值，失敗時傳回非0值。

whence值	whence識別字常數	檔案指標位置
0	SEEK_SET	檔案開頭
1	SEEK_CUR	目前檔案位置
2	SEEK_END	檔案尾

- 檔案指標位置若為檔開頭位置，則開頭之位元組編號為0，下一個位元組編號為1，再下一個位元組編號為2，等等。檔案指標位置若為目前檔案位置，則目前檔案位置位元組編號為「檔案指標位置+0」，下一個位元組編號為「檔案指標位置+1」，等等。

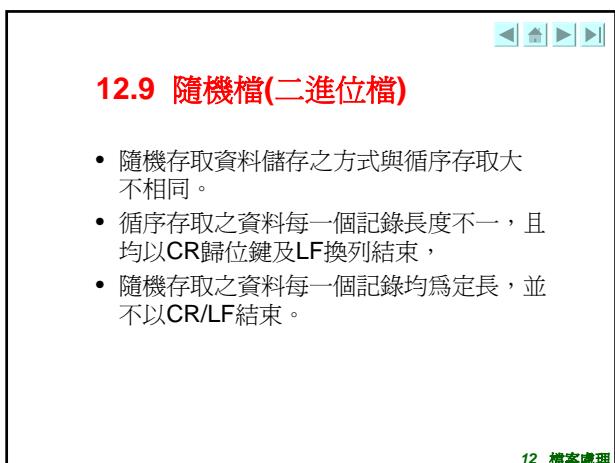
12 檔案處理

◀ ▶ ⟲ ⟳

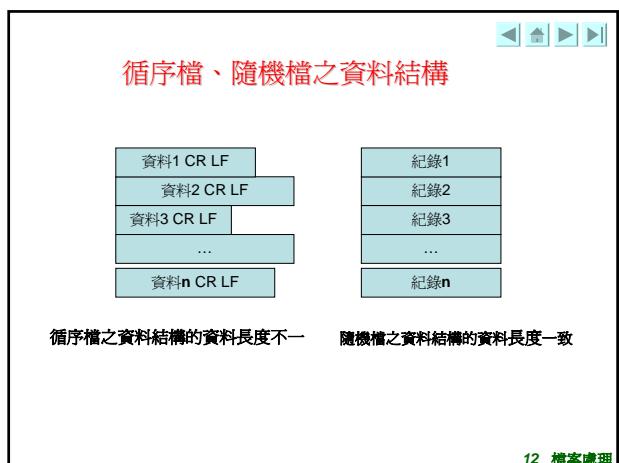
```
***** 【程式readwrite.c】 *****/
#include <stdio.h>
#include <string.h>
int main()
{
    char s[80];
    unsigned char uc;
    FILE f=fopen("readwrite.dat", "wb");
    printf("請輸入一個字符串: ");
    gets(s);
    fwrite(s, strlen(s), 1, f);
    fclose(f);
    printf("\n傾印readwrite.dat檔\n");
    f=fopen("readwrite.dat", "rb");
    while(1)
    {
        fread(&uc, 1, 1, f);
        if (feof(f)) break;
        printf("%02X ", uc);
    }
    fclose(f);
    return 0;
}
```

【執行】
 gcc readwrite.c -o readwrite.exe ↴
 readwrite.exe ↴
 請輸入一個字符串: 123 ABC ↴
 傾印readwrite.dat檔
 31 32 33 20 41 42 43

12 檔案處理



12 檔案處理



12 檔案處理

```

struct accttag acctnode;
FILE *acctfile;
long totalbytes, totalnodes;
void texttobase(void)
{
    char line[80];
    FILE *f=fopen("acct.txt", "rt");
    acctfile=fopen("acct.dat", "wb+");
    while ( fgets(line,sizeof(line),f) !=NULL)
    {
        printf("%s", line);
        sscanf(line,"%4d%12s%10f", &acctnode.account,
               &acctnode.name, &acctnode.amount);
        fwrite(&acctnode, sizeof(acctnode), 1, acctfile);
    }
    fclose(f);
    fclose(acctfile);
}

```

12 檔案處理

```

int acctshow(void)
{
    long n, offset;
    acctfile=fopen("acct.dat", "r+b");
    fseek(acctfile, 0L, SEEK_END);
    totalbytes = ftell(acctfile);
    totalnodes = totalbytes/sizeof(acctnode);
    printf("\n 輸入記錄編號 : ");
    scanf("%ld", &n);
    n = n % totalnodes;
    offset = n * sizeof(acctnode);
    fseek(acctfile, offset, SEEK_SET);
    fread(&acctnode, sizeof(acctnode), 1, acctfile);
    printf(" 客戶編號=%-7d\n", acctnode.account);
    printf(" 客戶姓名=%-20s\n", acctnode.name);
    printf(" 客戶存款=%-12.2f\n", acctnode.amount);
    fclose(acctfile);
    return acctnode.account;
}

```

12 檔案處理

```

FILE *acctfile;
int texttobase(void)
{
    char line[80];
    struct accttag acctnode;
    int n, count=0;
    FILE *f=fopen("acct.txt", "rt");
    while (1)
    {
        fgets(line, sizeof(line), f);
        if (feof(f)) break;
        sscanf(line,"%4d%12s%10f", &acctnode.account,
               &acctnode.name, &acctnode.amount);
        n = acctnode.account;
        fseek(acctfile, (n-1)*sizeof(acctnode), SEEK_SET);
        fwrite(&acctnode, sizeof(acctnode), 1, acctfile);
        count++;
    }
    fclose(f);
    return count;
}

```

12 檔案處理

```

int main()
{
    int choice=1, acct1=1, acct2=N-1;
    FILE *f = fopen("acctbase.dat", "r+");
    while (choice != 0)
    {
        printf("輸入選項 /1加入/2更新/3刪除/4至本文檔/5顯示/0結束: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1: newrecord(f); break;
            case 2: updaterecord(f); break;
            case 3: deleterecord(f); break;
            case 4: totextfile(f); break;
            case 5:
                printf("請輸入起迄客戶編號(%d-%d): ", 1,(N-1));
                scanf("%d %d", &acct1, &acct2);
                showrecord(f, acct1, acct2);
                break;
        }
    }
    fclose(f);
    return 0;
}

```

12 檔案處理

```
【執行】
acctbase.exe <Enter>
輸入選項 /1加入/2更新/3刪除/4至本文檔/5顯示/0結束: 1 <Enter>
輸入要新增的客戶編號(1-9999): 33 <Enter>
輸入姓名(字符串)及存款(浮點數): SmithWang 333.3 <Enter>
輸入選項 /1加入/2更新/3刪除/4至本文檔/5顯示/0結束: 5 <Enter>
請輸入起迄客戶編號: 33 33 <Enter>
編號 客戶姓名      存款
 33 SmithWang    333.30
輸入選項 /1加入/2更新/3刪除/4至本文檔/5顯示/0結束: 2 <Enter>
輸入要更新的客戶編號(1-9999): 33 <Enter>
 33 SmithWang    333.30
輸入調整金額(+)或(-): -33 <Enter>
 33 SmithWang    300.30
輸入選項 /1加入/2更新/3刪除/4至本文檔/5顯示/0結束: 3 <Enter>
輸入要刪除的客戶編號(1-9999): 33 <Enter>
輸入選項 /1加入/2更新/3刪除/4至本文檔/5顯示/0結束: 0 <Enter>
```

12 檔案處理