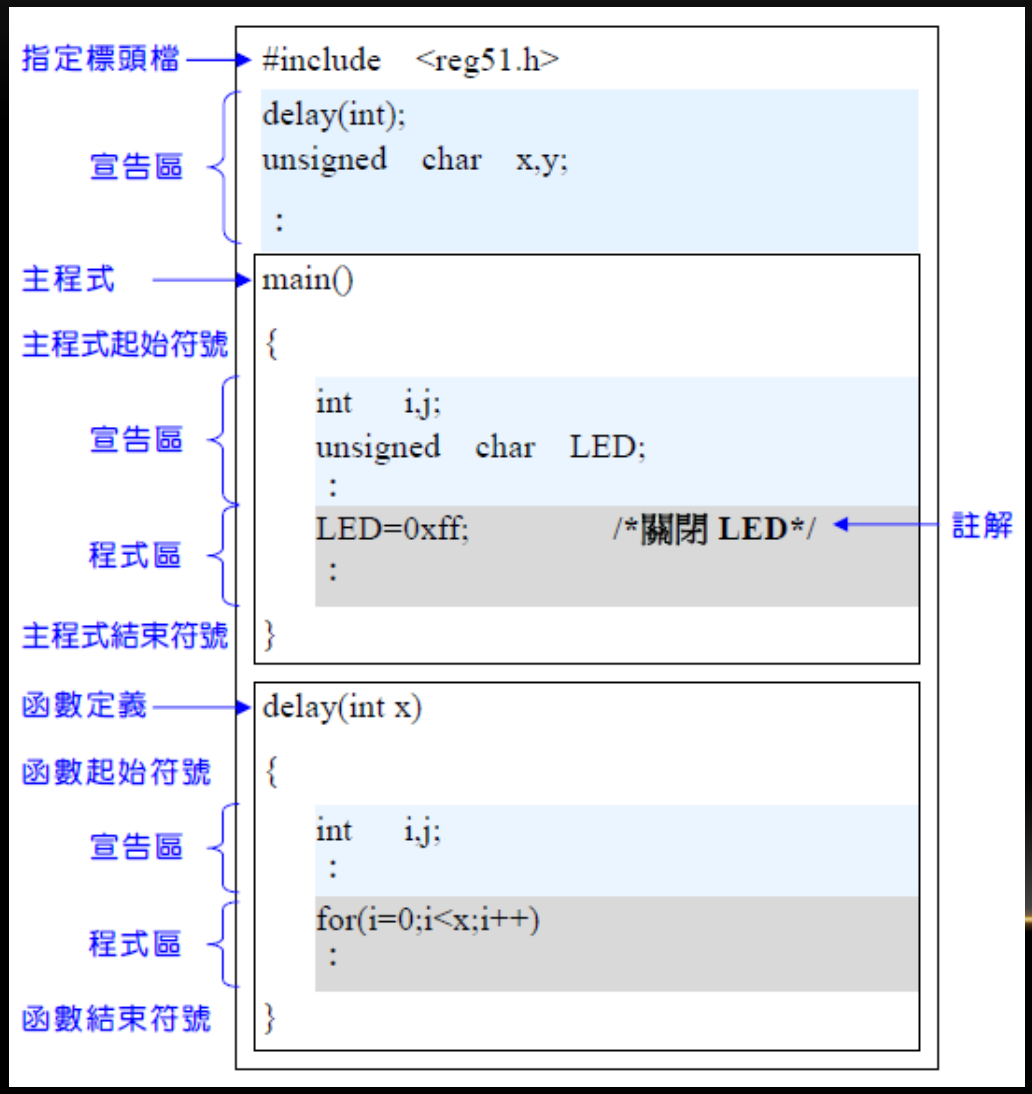
單晶片控制實習--8051 C語言

CH3 Keil C基本架構

KEIL C基本架構

- 由許多獨立的函數（Function）所組成
- 基本架構包括：
  - 指定標頭檔：將預先定義的8x51內部暫存器位址的資料放入程式中（例如學校的平面配置圖）
  - 宣告區：宣告的變數、常數、函數擴及整個程式（例如勤401教室為電二智教室、OOO為校長）
  - 主程式： 分為宣告區、程式區（例如OOO擔任電機研習社指導老師，並負責該社團所有活動）
  - 函數（副程式）： 分為宣告區、程式區（例如OOO擔任電機研習社社長，並負責紀錄社員出缺席狀況）

KEIL C基本架構




```
01 //test.c - LED高低位元交互閃爍程式
02
03 //==宣告區=============================================
04 #include    <reg51.h>    //定義8051暫存器之標頭檔
05 #define     LED P2       //定義LED接至P2
06 void delay(int);        //宣告延遲函數，範圍擴及整個程式
07
08 //==主程式=============================================
09 main()                  //主程式開始
10 {
11     LED=0x0f;           //初值=00001111，左邊四個LED亮，
12     while(1)            //無窮迴圈，程式一直跑
13     {
14         delay(200); //呼叫延遲函數
15         LED=~LED;       //LED反相輸出
16
17     }                   //while迴圈結束
18
19 }                       //主程式結束
20
21 //==延遲函數=============================================
22 void delay(int x)           //延遲函數開始，傳入x數值，不
23 {
24     int i,j;                //宣告整數變數i,j
25     for(i=0;i<x;i++)        //計數x次，延遲x*5mS
26         for(j=0;j<1730;j++);    //計數1730次，延遲5mS
27 }                           //延遲函數結束
28
```

KEIL C基本架構-指定標頭檔

```
/*--------------------------------------------------------------------------
REG51.H
Header file for generic 80C51 and 80C31 microcontroller.
Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
--------------------------------------------------------------------------*/


#ifndef __REG51_H__
#define __REG51_H__

/*    BYTE Register    */
sfr P0      = 0x80;                 /* Port 0 */
sfr P1      = 0x90;                 /* Port 1 */
sfr P2      = 0xA0;                 /* Port 2 */
```

```
/*    BIT Register    */
/*    PSW      */
sbit CY     = 0xD7;                 /*  進位位元  */
sbit AC     = 0xD6;                 /*  輔助進位位元  */
sbit F0     = 0xD5;                 /*  使用者旗標  */
```

詳見課本P.2-16～P.2-18

KEIL C基本架構-宣告區

- 函數宣告：

傳回引數之資料型態　　函數名稱(傳入引數之資料型態)

- Int　My_func(char); //分號（;）代表結束符號
- void　delay(int);　or　delay(int);　用於延遲函數
- Int　My_func(void);
- void　debounce(void);　用於防彈跳函數

KEIL C基本架構-宣告區

- 變數、常數宣告：

資料型態	變數、常數名稱

- Int x,y,z; //同時宣告多個變數，以逗號「,」區隔
- int x=50;

在宣告區的變數、常數，其範圍擴及整個程式。
亦稱為整體變數

KEIL C基本架構-宣告區

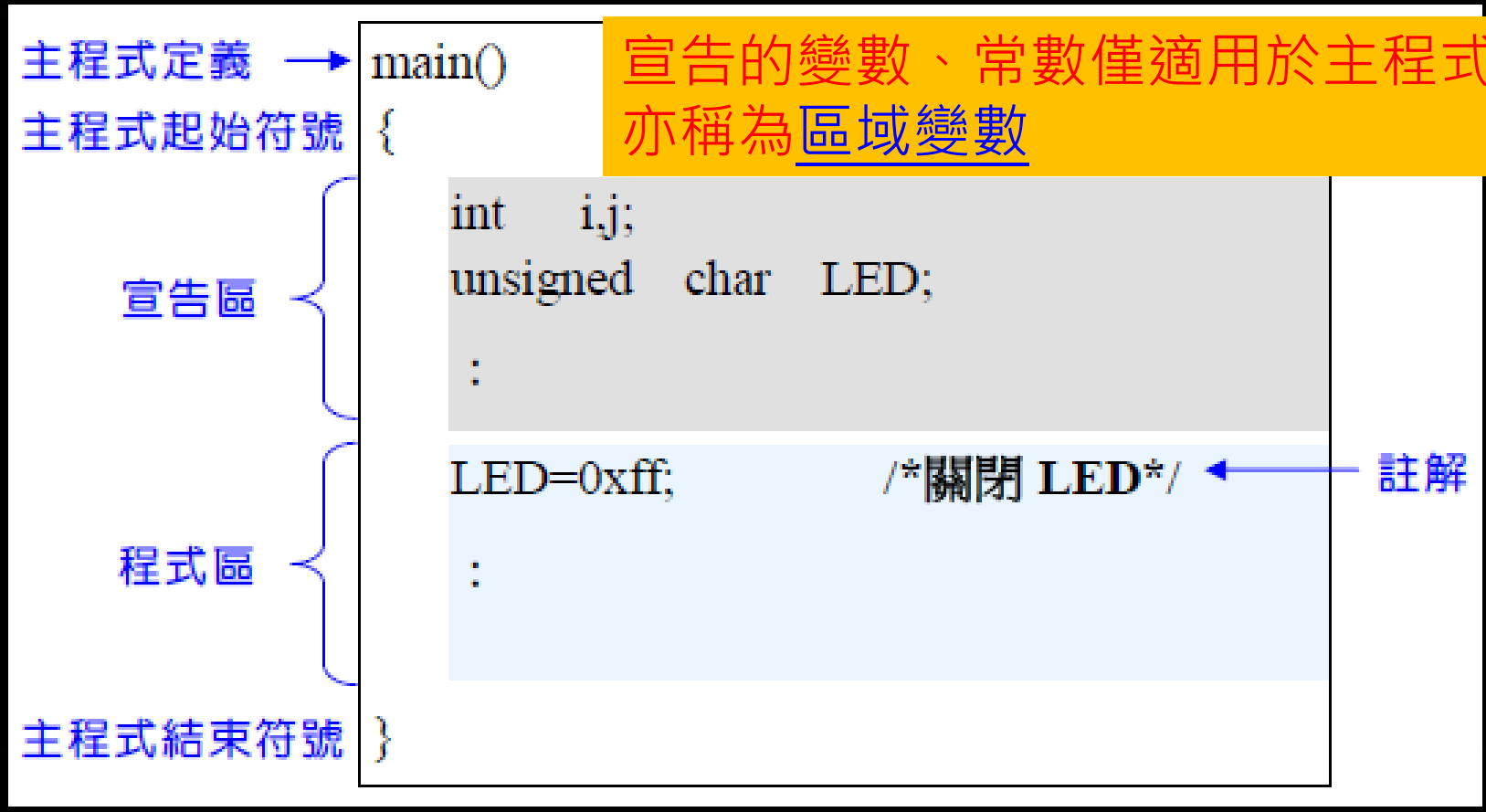
- 前置命令：
  - 包含命令：將指定的定義或宣告檔案，放入程式中

```
#include <reg51.h>      #include “myio.h”
```

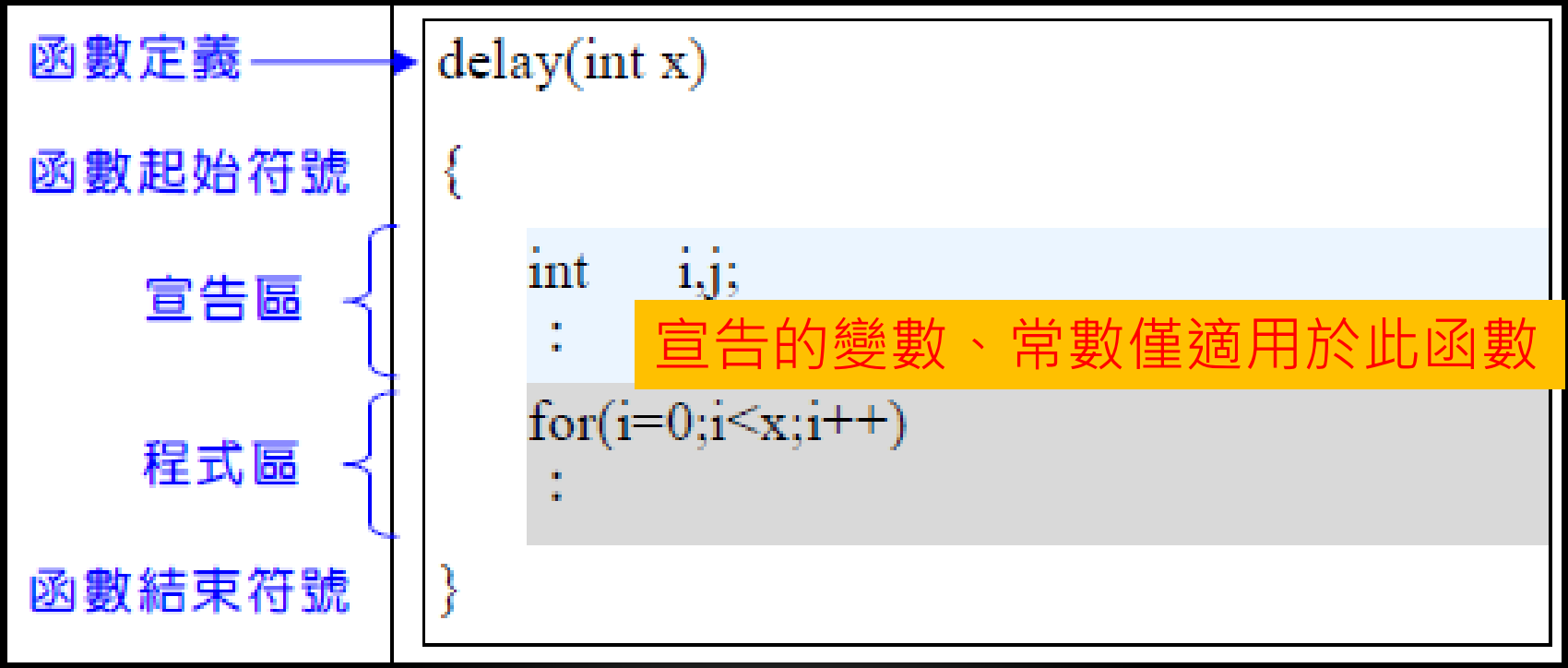
  - 定義命令：指定常數、字串、巨集函數的代名詞

```
#define count 5000      #define LED P2
#define TH_M1  (65536-count)/256
```

KEIL C基本架構-主程式



KEIL C基本架構-函數



KEIL C基本架構-函數

- 函數定義：

傳回引數之資料型態　　函數名稱(傳入引數之資料型態)

- Int　My_func(char x);
- void　delay(int x);　or　delay(int x);
- Int　My_func(void);
- void　debounce(void);

與宣告區中的函數宣告有何不同？

void delay(int)

變數與常數的資料型態

- 變數：在指定的記憶體位置中，放置一可變的資料
- 常數：在指定的記憶體位置中，放置一固定不變的資料

- 那到底要指定多大的記憶體位置給變數或常數呢？

變數與常數的資料型態-通用資料型態

$-2^{n-1} \sim 2^{n-1}-1$

$0 \sim 2^{n}-1$

型 態	名 稱	位元數	範 圍
char	字元	8	−128～+127
unsigned char	無號數字元	8	0～255
enum	列舉	8/16	−128～+127 / −32768～+32767
short	短整數	16	−32768～+32767
unsigned short	無號整數	16	0～65535
int	整數	16	−32768～+32767
unsigned int	無號整數	16	0～65535
long	長整數	32	$-2^{31} \sim +2^{31}-1$
unsigned long	無號長整數	32	$0 \sim 2^{32}-1$
float	浮點數	32	$\pm 1.175494 \times 10^{-38} \sim 3.402823 \times 10^{38}$
double	雙倍精度浮點數	64	$\pm 1.7 \times 10^{308}$
void	無	0	無

- Int x=10000 int x,y,z
- Unsigned char x=10000 →→錯誤（數值已超出範圍）

變數與常數的資料型態-8X51專屬資料型態

名 稱	位元數	範 圍
bit	1	0、1
sbit	1	0、1
sfr	8	0～255
sfr16	16	0～65535

- bit：在資料記憶體20H～2FH之間，定義1位元的變數
- sbit：在20H～2FH 或SFR（80H～FFH）之間，定義1位元的變數
- sfr：在SFR（80H～FFH）之間，定義8位元的變數
- sfr16：在SFR（80H～FFH）之間，定義16位元的變數

由於指定標頭檔（reg51.h）已宣告SFR，因此不必再次宣告

變數與常數的名稱

- 可使用大小寫字母、數字、底線
- 第一個字元不可為數字
- 不可使用保留字

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

表 3　ANSI C 與傳統 C 之保留字

_at_	_priority_	_task_	alien	bdata	bit
code	compact	data	far	idata	interrupt
large	pdata	reentrant	sbit	sfr	sfr16
small	using	xdata			

表 4　Keil C 保留字

記憶體形式

記憶體形式	說　明	適用範圍
code	程式記憶體	0x0000～0xffff(64k)
data	直接定址的內部資料記憶體	0x00～0x7f(128)
idata	間接定址的內部資料記憶體	0x80～0xff(128)
bdata	位元定址的內部資料記憶體	0x20～0x2f(16)
xdata	以 DPTR 定址的外部資料記憶體	64k bytes 之內
pdata	以 R0、R1 定址的外部資料記憶體	256 bytes 之內
far	擴充的 ROM 或 RAM 外部記憶體，僅適用於少數的晶片，如 Philips 80C51MX、Dallas 390 等。	最大可達 16M bytes

- 程式記憶體是一種唯讀記憶體，通常是用來存放程式碼。除此之外，也可利用陣列的方式來存放固定的資料

```
char   code   SEG[10]={   0x03, 0x9f, 0x25, 0x0d, 0x99,
                          0x49, 0xc1, 0x1f, 0x01, 0x19   };
```

KEIL C指令集

- 運算指令
  - 算術、關係、邏輯、布林、指定、遞增、遞減
- 流程控制指令
  - 迴圈、選擇、跳躍
- 陣列及指標

KEIL C運算指令-算術運算

符 號	功 能	範 例	說 明
+	加	A=x+y	將 x 與 y 變數的值相加，其和放入 A 變數
–	減	B=x-y	將 x 變數的值減去 y 變數的值，其差放入 B 變數
*	乘	C=x*y	將 x 與 y 變數的值相乘，其積放入 C 變數
/	除	D=x/y	將 x 變數的值除以 y 變數的值，其商數放入 D 變數
%	取餘數	E=x%y	將 x 變數的值除以 y 變數的值，其餘數放入 E 變數

KEIL C運算指令-關係運算

符 號	功 能	範 例	說 明
==	相等	x==y	比較 x 與 y 變數的值是否相等，相等則其結果爲 1、不相等則爲 0
!=	不相等	x!=y	比較 x 與 y 變數的值是否相等，不相等則其結果爲 1、相等則爲 0
>	大於	x>y	若 x 變數的值大於 y 變數的值，其結果爲 1、否則爲 0
<	小於	x<y	若 x 變數的值小於 y 變數的值，其結果爲 1、否則爲 0
>=	大等於	x>=y	若 x 變數的值大於或等於 y 變數的值，其結果爲 1、否則爲 0
<=	小等於	x<=y	若 x 變數的值小於或等於 y 變數的值，其結果爲 1、否則爲 0

用來輔助流程控制指令　for(i=0;i<100;i++)　//當i<100時，其結果為1，進入迴圈執行程式，i>=100時，其結果為0，離開迴圈

KEIL C運算指令-邏輯運算

符 號	功 能	範 例	說 明
&&	及運算	(x>y)&&(y>z)	若 x 變數的值大於 y 變數的值，且 y 變數的值也大於 z 變數的值，其結果爲 1，否則爲 0
||	或運算	(x>y)||(y>z)	若 x 變數的值大於 y 變數的值，或 y 變數的值也大於 z 變數的值，其結果爲 1，否則爲 0
!	反相運算	!(x>y)	若 x 變數的值大於 y 變數的值，則其結果爲 0，否則爲 1

用來輔助流程控制指令　if ((x>y)&&(y>z))　　//當x>y且y>z時，進入迴圈執行程式

KEIL C運算指令-布林運算

符 號	功 能	範 例	說 明
&	及運算	A=x&y	將 x 與 y 變數的每個位元，進行 AND 運算，其結果放入 A 變數
|	或運算	B=x|y	將 x 與 y 變數的每個位元，進行 OR 運算，其結果放入 B 變數
^	互斥或	C=x^y	將 x 與 y 變數的每個位元，進行 XOR 運算，其結果放入 C 變數
~	取 1's 補數	D=~x	將 x 變數的值，進行 NOT 運算，其結果放入 D 變數
<<	左移	E=x<<*n*	將 x 變數的值左移 *n* 位，其結果放入 E 變數
>>	右移	F=x>>*n*	將 x 變數的值左移 *n* 位，其結果放入 F 變數

KEIL C運算指令-指定運算

符 號	功 能	範 例	說 明
=	指定	A=x	將 x 變數的值，放入 A 變數
+=	加入	B+=x	將 B 變數的值與 x 變數的值相加，其和放入 B 變數，與 B=B+x 相同
−=	減去	C −=x	將 C 變數的值減去 x 變數的值，其差放入 C 變數，與 C=C −x 相同
*=	乘入	D*=x	將 D 變數的值與 x 變數的值相乘，其積放入 D 變數，與 D=D*x 相同
/=	除	E/=x	將 E 變數的值除以 x 變數的值，其商放入 E 變數，與 E=E/x 相同
%=	取餘數	F%=x	將 F 變數的值除以 x 變數的值，其餘數放入 F 變數，與 F=F%x 相同
&=	及運算	G&=x	將 G 變數的值與 x 變數的值進行 AND 運算，其結果放入 G 變數，與 G=G&x 相同
|=	或運算	H|=x	將 H 變數的值與 x 變數的值進行 OR 運算，其結果放入 H 變數，與 H=H|x 相同
^=	互斥或	I^=x	將 I 變數的值與 x 變數的值進行 XOR 運算，其結果放入 I 變數，與 I=I^x 相同
<<=	左移	J<<=*n*	將 J 變數的值左移 *n* 位，與 J=J<<*n* 相同
>>=	右移	K>>=*n*	將 K 變數的值右移 *n* 位，與 K=K>>*n* 相同

KEIL C運算指令-遞增/遞減運算

符 號	功 能	範 例	說 明
++	加 1	x++	執行運算後，再將 x 變數的值加 1
– –	減 1	x--	執行運算後，再將 x 變數的值減 1

KEIL C運算指令-運算的優先順序

優先順序	運算子或操作符號	說 明
1	(、)	小括號
2	~、!	補數、反相運算
3	++、 −−	遞增、遞減
4	*、/、%	乘、除、取餘數
5	+、 −	加、減
6	<<、>>	左移、右移
7	<、>、<=、>=、==、!=	關係運算子
8	&	布林運算－AND
9	^	布林運算－XOR
10	|	布林運算－OR
11	&&	邏輯運算－AND
12	||	邏輯運算－OR
13	=、*=、/=、%=、+=、 −=、<<=、>>=、&=、^=、|=	指定運算子

KEIL C流程控制指令-迴圈指令1

- for迴圈：計數迴圈

```
for(運算式 1; 運算式 2; 運算式 3)
{
        指令;
        [break;]
         :
 }
```

運算式1：初始值
運算式2：判斷條件
運算式3：條件運算方式

KEIL C流程控制指令-迴圈指令1

- for迴圈：計數迴圈

使用範例	說明
`for(i=0;i<8;i++)`	迴圈執行8次 i=0,1,2,3,4,5,6,7
`for(x=100;x>0;x--)`	迴圈執行100次
`for(;;)`	無窮迴圈
`for(num=0;num<99;num+=5)`	迴圈執行20次 Num=0,5,10……95

KEIL C流程控制指令-迴圈指令1

- for迴圈：計數迴圈

使用範例	說明
`for(i =0;i<10;i++)` `P2=table[i];`	單行指令可省略刮號

`for(i =0;i<100;i++)`
`{`
`:`
`if (sw1==0) break;`
`:`
`}`
	如果sw1=0時，強制跳出迴圈

KEIL C流程控制指令-迴圈指令2

- while迴圈：前條件迴圈

```
while(運算式)
{
        指令;
        [break;]
        :
}
```

先判斷再執行

KEIL C流程控制指令-迴圈指令2

- while迴圈：前條件迴圈

使用範例	說明

while(i !=0)
{ :
指令;
:
}
	i不等於0時，才開始執行迴圈

while(1)
{ :
指令;
:
}
	無窮迴圈

KEIL C流程控制指令-迴圈指令2

- while迴圈：前條件迴圈

使用範例	說明
`while(i !=0)` `  i--;`	單行指令可省略刮號

`while(1)`
`{  :`
`  if (sw1==0) break;`
`  :`
`}`
	如果sw1=0時，強制跳出迴圈

KEIL C流程控制指令-迴圈指令3

- do-while迴圈：後條件迴圈

```
do      {
        指令;
        [break;]
         :
        } while(運算式);
```

先執行再判斷

會先執行一次迴圈，再判斷運算式是否成立？
1. 若成立，則繼續執行迴圈
2. 若不成立，則跳出迴圈

KEIL C流程控制指令-迴圈指令3

- do-while迴圈：後條件迴圈

使用範例	說明
do { 指令; : } while(i !=0);	i不等於0時，才繼續執行迴圈
do { 指令; : } while(1);	無窮迴圈

KEIL C流程控制指令-迴圈指令3

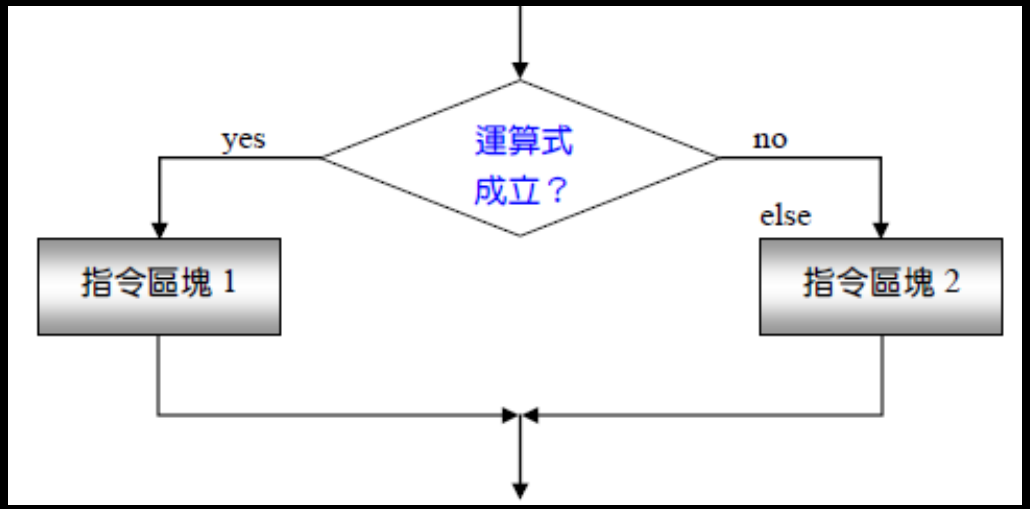
- do-while迴圈：後條件迴圈

使用範例	說明
`do i--; while(i !=0);`	單行指令可省略刮號
`do { if (sw1==0) break; : } while(1);`	如果sw1=0時，強制跳出迴圈

KEIL C流程控制指令-選擇指令1

- if-else選擇：條件選擇

```
if (運算式)
{
        指令區塊 1;
          :
 }
 else
{
        指令區塊 2;
          :
 }
```

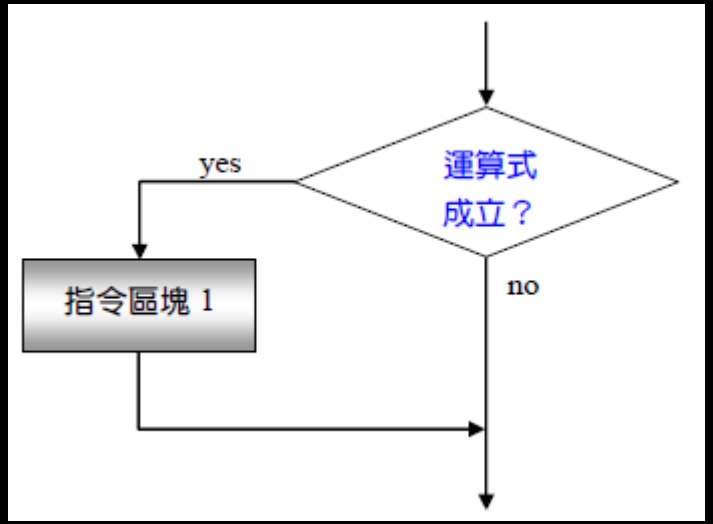


先判斷運算式是否成立？
1. 若成立，則執行指令區塊1
2. 若不成立，則執行指令區塊2

KEIL C流程控制指令-選擇指令1

- if-else選擇：條件選擇

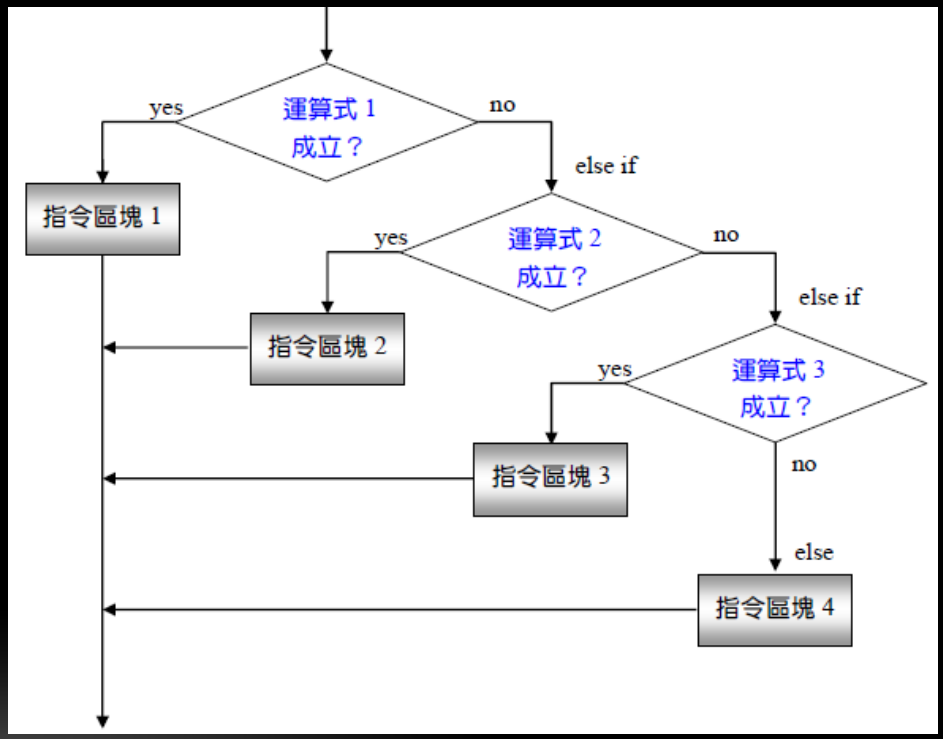
```
if (運算式)   { 指令區塊 1;}
  其它指令;
```



KEIL C流程控制指令-選擇指令1

- if-else if選擇：條件選擇

```
if (運算式 1)
        { 指令區塊 1; }
  else if (運算式 2)
        { 指令區塊 2; }
  else if (運算式 3)
        { 指令區塊 3; }
  else  { 指令區塊 4; }
        :
```



多重條件判斷，優先等級分別為1到4

KEIL C流程控制指令-選擇指令1

- if-else與if-else if之差別：

```
14    while(1)
15    {
16        if(PB1==0)            //當PB1按下時
17        {
18            debounce();
19            alter(3);
20            flash(3);
21        }
22
23        if(PB2==0)            //當PB2按下時
24        {
25            debounce();
26            left(3);
27            flash(3);
28        }
```

PB1與PB2同時按下時，有可能先執行PB1再執行PB2，或者先執行PB2再執行PB1

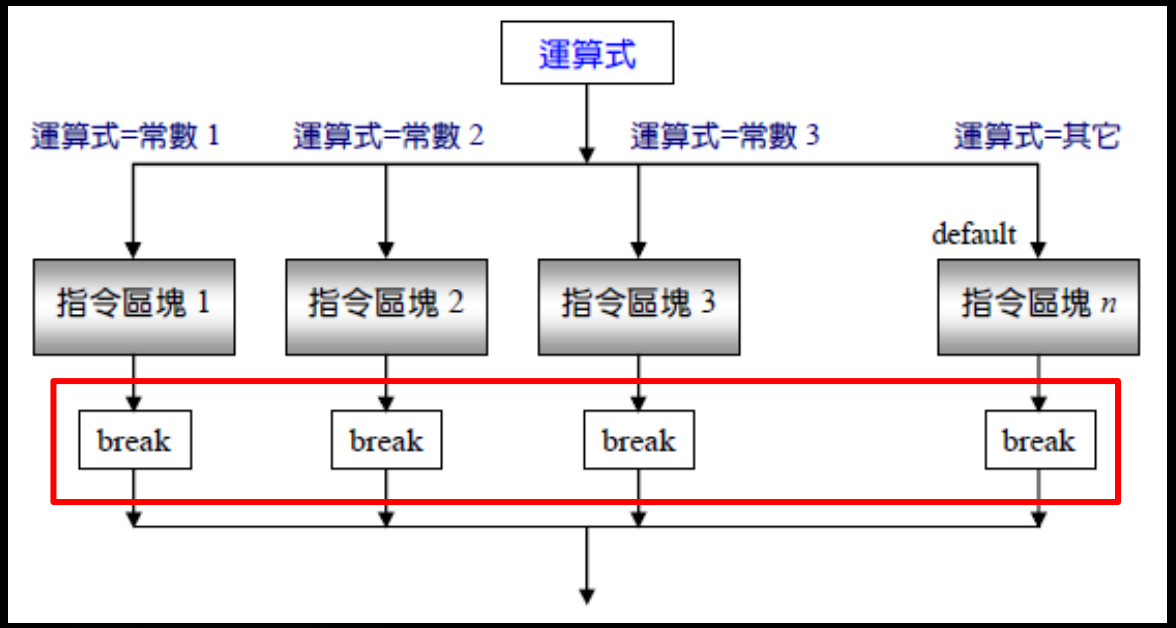
```
14    while(1)
15    {
16        if(PB1==0)            //當PB1按下時
17        {
18            debounce();
19            alter(3);
20            flash(3);
21        }
22
23        else if(PB2==0)       //當PB2按下時
24        {
25            debounce();
26            left(3);
27            flash(3);
28        }
```

PB1與PB2同時按下時，會優先執行PB1

KEIL C流程控制指令-選擇指令2

- switch-case選擇：開關式選擇

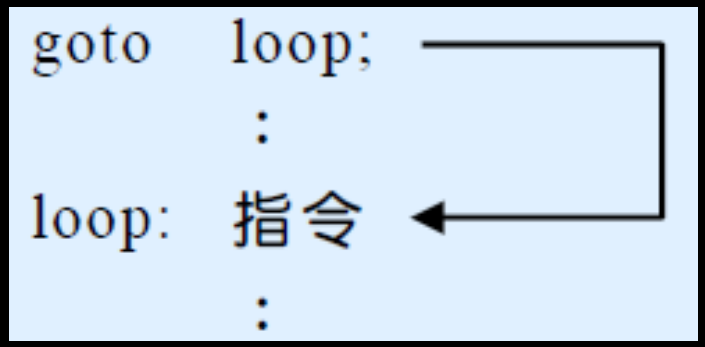
```
switch (運算式)
{  case (常數 1):
          { 指令區塊 1; }
          break;
   case (常數 2):
          { 指令區塊 2; }
          break;
           :
   default:
          { 指令區塊 n; }
          break;
}
```



無優先等級，每一個指令區塊結束時，必須下一個「break」

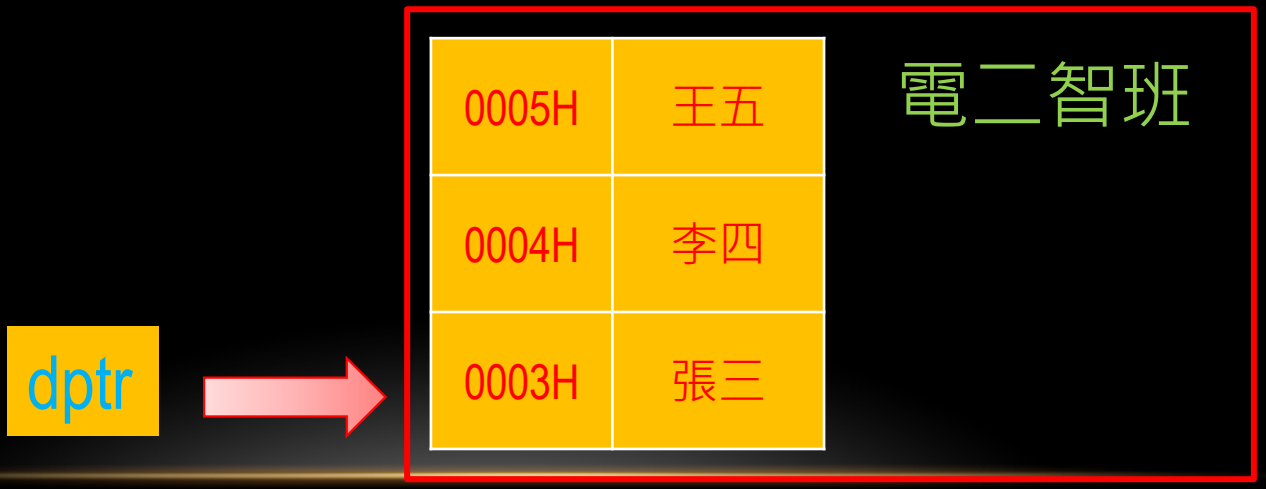
KEIL C流程控制指令-跳躍指令

- goto：無條件跳躍



KEIL C陣列及指標

- 陣列：將許多同資料型態的變數做一集合管理，並賦予一個變數名稱來表示。例如：03張三、04李四、05王五被編班到電二智班
- 指標：用來存放記憶體位址的變數。



P2 = *dptr+2 = 0003H+2 = 0005H =王五

KEIL C陣列及指標- 1維陣列

```
資料型態    陣列名稱[陣列大小];
```

```
char    LCM[9];
```

```
char    LCM[9]="Testing.";
```

01234567 \0

```
char    string1[]="Welcome to Taiwan.";
```

```
int    Num[6]={30, 21, 1, 45, 26, 37};
```

0 1 2 3 4 5

數量為6，但是起始數值從0開始　Num[3]=45

```
char    code    SEG[10]={   0x03, 0x9f, 0x25, 0x0d, 0x99,
                            0x49, 0xc1, 0x1f, 0x01, 0x19   };
```

KEIL C陣列及指標- 2維陣列

```
資料型態    陣列名稱[陣列大小 1] [陣列大小 2]...[陣列大小 n];
```

```
int    Num[3][2]={{10,11}, {12,13}, {14,15}};
```

[列][行]

Row \ Col	0	1
0	10	11
1	12	13
2	14	15

Num[0][0]=10
Num[1][1]=13
Num[2][1]=15

KEIL C陣列及指標-指標

資料型態 *變數名稱;

通常指標皆採整數資料型態

利用陣列及指標

```
int      *dptr;
dptr=&Num[0][0];        //&取得Num[0][0]的位址
P2=*dptr;
P2=*(dptr+3);
```

利用指定運算

```
P2=Num[0][0];
P2=Num[1][1];
```