

CHAPTER 1



Introduction to C Programming (ch1)(ch2)

• 人類與電腦溝通所用的語言

1. 高階語言 (High-Level Language)

- 採取人類日常使用的語言，以敘述句 (Statement) 的方式，規則化、公式化編寫程式，經過特定軟體編譯後，產生很多 0、1 訊號組成的指令以供電腦執行
- Ex: C/C++, Java
- 相關課程: 資料結構, 演算法, 軟體工程, 應用程式設計

2. 組合語言 (Assembly Language)

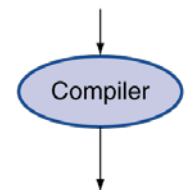
- 使用縮寫代碼 (Mnemonic Code) 的符號或字彙來取代 0、1 的指令、資料與記憶位址，再組譯成機器語言

3. 機器語言 (Machine Language)

- Binary digits (bits) 所有指令、資料與記憶位址均需用 0 與 1 來編寫
- 相關課程: 計算機組織

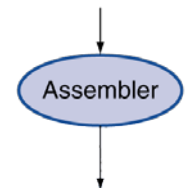
High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```



Assembly
language
program
(for MIPS)

```
swap:
    muli $2, $5, 4
    add $2, $4, $2
    lw $15, 0($2)
    lw $16, 4($2)
    sw $16, 0($2)
    sw $15, 4($2)
    jr $31
```

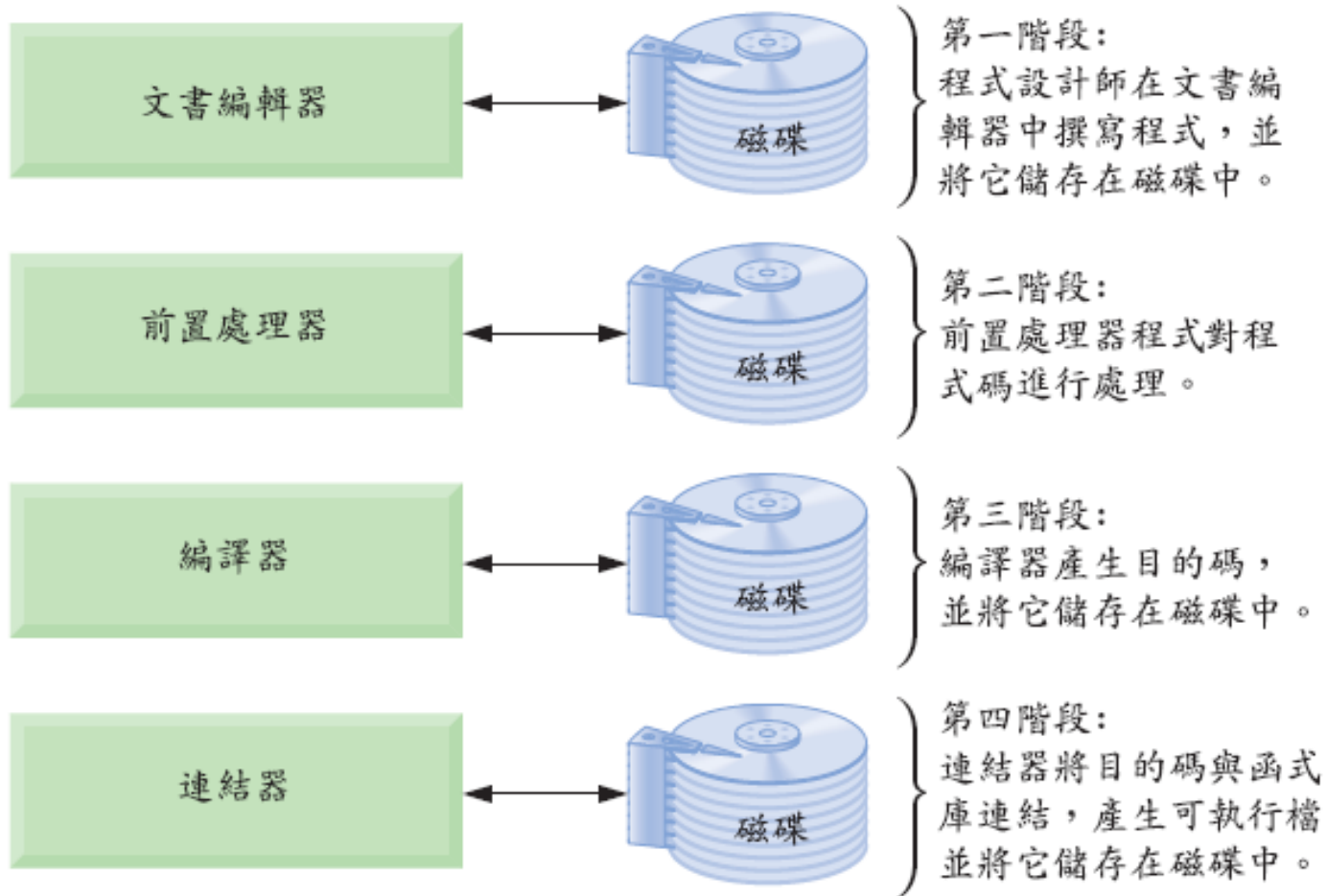


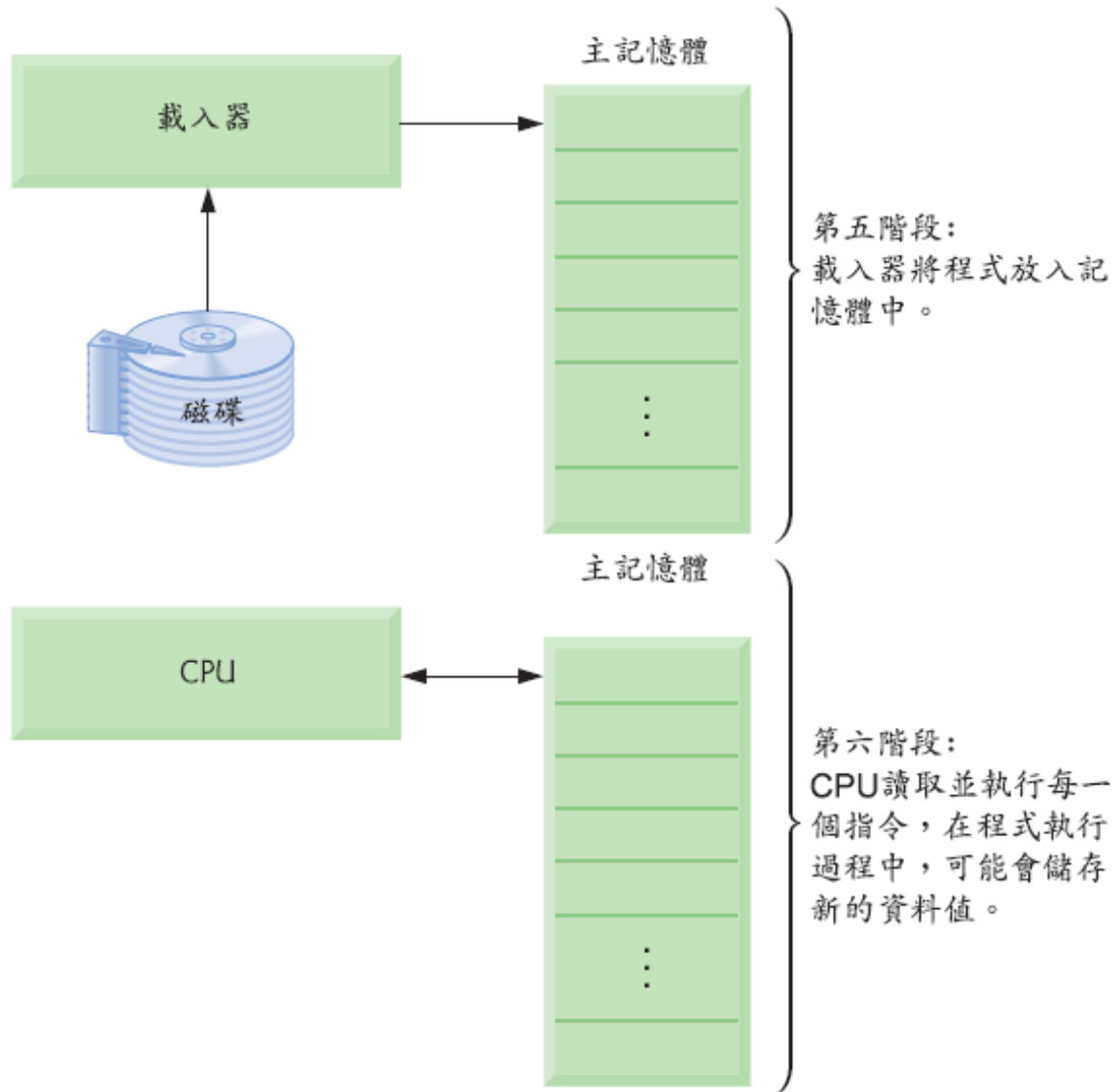
Binary machine
language
program
(for MIPS)

```
000000001010000100000000000011000
000000000000110000001100000100001
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

- C的演進
 - 1972年由Dennis Ritchie與Ken Thompson在貝爾實驗室 (Bell Laboratory)所發展出來的。
 - 1980 Bjarne Stroustrup將C擴充成C++，加入「物件導向」的觀念，與「封裝」、「繼承」、「多型」等功能，使程式設計者多了不少好用的工具。
- C的版本很多
 - Dev C/C++ , Visual C/C++ , Microsoft Visual C/C++ 2010 ,
 - Turbo C/C++ , Borland C/C++ , Lattice C/C++ ,
 - Quick C/C++ , Linux C/C++ 等等。

- 效率高
 - 贏過 PASCAL , COBOL , FORTRAN等編譯器。
- 中高階語言
 - 兼具高(適於人類)、低階特色(適於機器)。
- 結構化控制
 - 標準化：sequence，selection，iteration三大類。
- 可攜性佳
 - 適用於各型與不同廠牌之電腦。
- 語法精簡：例如
 - (1) `i ++ ; i = i+1 ;`
 - (2) `i+=j ; i = i+j ;`
 - (3) `(a>b? a=y : a=n ;) if(a>b) a=y ;else a=n;`
- 擴充性高
 - 豐富的程式庫。(#include)
 - 模組化





```
#include <stdio.h>
#include <stdlib.h>
//...
void sub_prog(void) ;
char ans[3] ;
//...
int main(void)
{
    int i , j ;
    i = 10 ; j = 20 ;
    printf("這是主程式的輸出 = %d\n", i + j) ;
    sub_prog( ) ; gets(ans) ;
}
void sub_prog(void)
{
    int m = 123 ;
    printf("這是副程式的輸出 = %d\n" , m) ;
}
```

1. 前置處理敘述區
(Pre-processor
directive)

前置處理敘述區

外部宣告區

2. 外部宣告區
(External Declaration)

主程式區

3. 主程式區
(Main program)

副程式區

4. 副程式區
(Subprogram)

```
1  /* Fig. 2.1: fig02_01.c
2     A first program in C */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     printf( "Welcome to C!\n" );
9
10     return 0; /* indicate that program ended successfully */
11 } /* end function main */
```

Welcome to C!

Fig. 2.1 | A first program in C.

Lab1: 環境建置、開新專案與撰寫程式


```
#include <stdio.h>
#include <stdlib.h>
//...
void sub_prog(void);
char ans[3];
//...
int main(void)
{
    int i, j;
    i = 10; j = 20;
    printf("這是主程式的輸出 = %d\n", i + j);
    sub_prog(); gets(ans);
}
void sub_prog(void)
{
    int m = 123;
    printf("這是副程式的輸出 = %d\n", m);
}
```

前置處理敘述區

外部宣告區

主程式區

副程式區

- 前置處理敘述指令

- 一般放在C語言程式的最前端
- 前置處理大部份是以「#」符號為開頭
- EX: #include, #define, #ifdef, #endif 等。

- 所謂的「前置處理」

- 這些指令在程式編譯之前就會先處理
- 再將處理後的程式碼交給編譯程
- 它會合併主、副程式一起翻譯成機器碼

```
#include <stdio.h>
#include <stdlib.h>
//...
void sub_prog(void);
char ans[3];
//...
int main(void)
{
    int i, j;
    i = 10; j = 20;
    printf("這是主程式的輸出 = %d\n", i + j);
    sub_prog(); gets(ans);
}
void sub_prog(void)
{
    int m = 123;
    printf("這是副程式的輸出 = %d\n", m);
}
```

前置處理敘述區

外部宣告區

主程式區

副程式區

- `#include<library_header_name>`
 - `Library_header_name`是指內建庫存函式的標頭檔名
- `#include"user_defined_header_name"`
 - `User_defined_header_name`是指程式設計者自訂的標頭檔名
 - 此標頭檔必須放在與主程式檔相同的資料夾中，否則就要標示全部的檔名路徑(Full Pathname)。
 - 建議程式架構
 - 資料夾include, 放置*.h檔(自訂標頭檔)
 - 資料夾source, 放置*.c檔(C程式)
 - 資料夾temp, 放置輸入與輸出檔案

Lab2: include User_defined (lab1延伸)

```
#include <stdio.h>
#include <stdlib.h>
//...
void sub_prog(void);
char ans[3];
//...
int main(void)
{
    int i, j;
    i = 10; j = 20;
    printf("這是主程式的輸出 = %d\n", i + j);
    sub_prog(); gets(ans);
}
void sub_prog(void)
{
    int m = 123;
    printf("這是副程式的輸出 = %d\n", m);
}
```

前置處理敘述區

外部宣告區

主程式區

副程式區

- 外部宣告區

- 全域變數 (Global variables)
- 函式原型 (Prototype of Functions)
- 函式定義 (Definition of Functions)
- 複合資料型態 (Combined Data Type)
- 外來識別字 (External Identifiers)

```
#include <stdio.h>
#include <stdlib.h>
//...
void sub_prog(void);
char ans[3];
//...
int main(void)
{
    int i, j;
    i = 10; j = 20;
    printf("這是主程式的輸出 = %d\n", i + j);
    sub_prog(); gets(ans);
}
void sub_prog(void)
{
    int m = 123;
    printf("這是副程式的輸出 = %d\n", m);
}
```

前置處理敘述區

外部宣告區

主程式區

副程式區

• 主程式區

- 每個C程式都是從main函式開始執行，main標題

1. 簡單型：void main(void)

2. 傳回型：int main(void)

- 位於main左邊的關鍵字int，表示main函式會「回傳」一個整數值。

3. 傳入型：int main(int argc , char *argv[])

- argc 是指命令行輸入參數的個數，argv存儲了所有的命令行參數

- 宣告區域變數

- 程式主體

```
#include <stdio.h>
#include <stdlib.h>
//...
void sub_prog(void);
char ans[3];
//...
int main(void)
{
    int i, j;
    i = 10; j = 20;
    printf("這是主程式的輸出 = %d\n", i + j);
    sub_prog(); gets(ans);
}
void sub_prog(void)
{
    int m = 123;
    printf("這是副程式的輸出 = %d\n", m);
}
```

前置處理敘述區

外部宣告區

主程式區

副程式區

• 副程式區

1. 「副程式」與「主程式」規格相同，包括三部份
 - 「副程式標題」
 - 「宣告區域變數」
 - 「副程式主體」
2. 副程式需要在「外部宣告區」宣告。

撰寫程式碼及註解

編譯程式

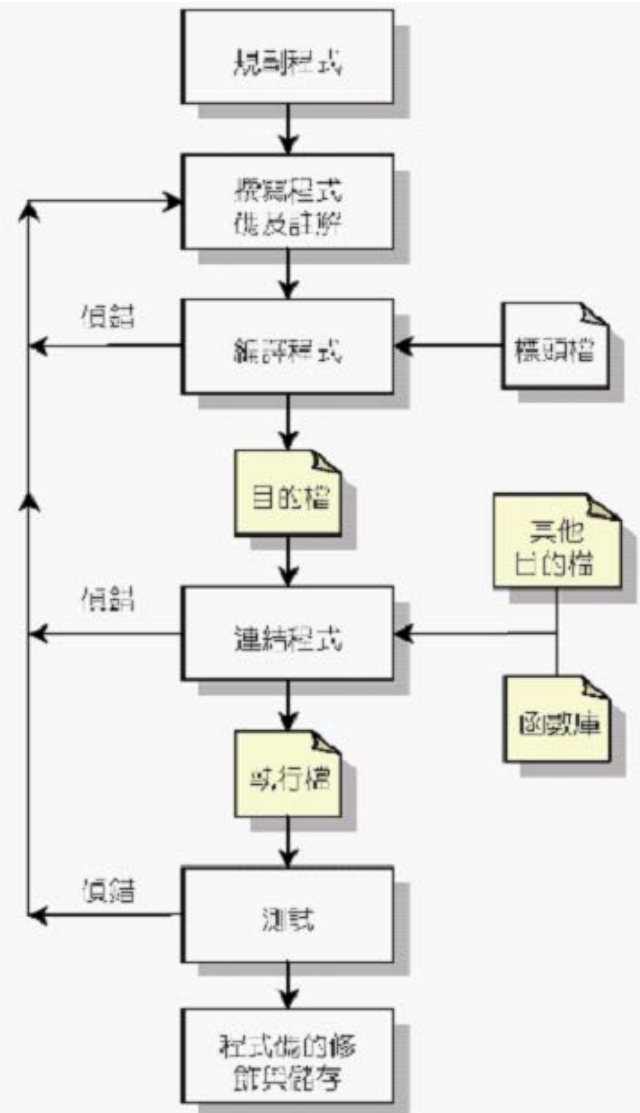
連結程式

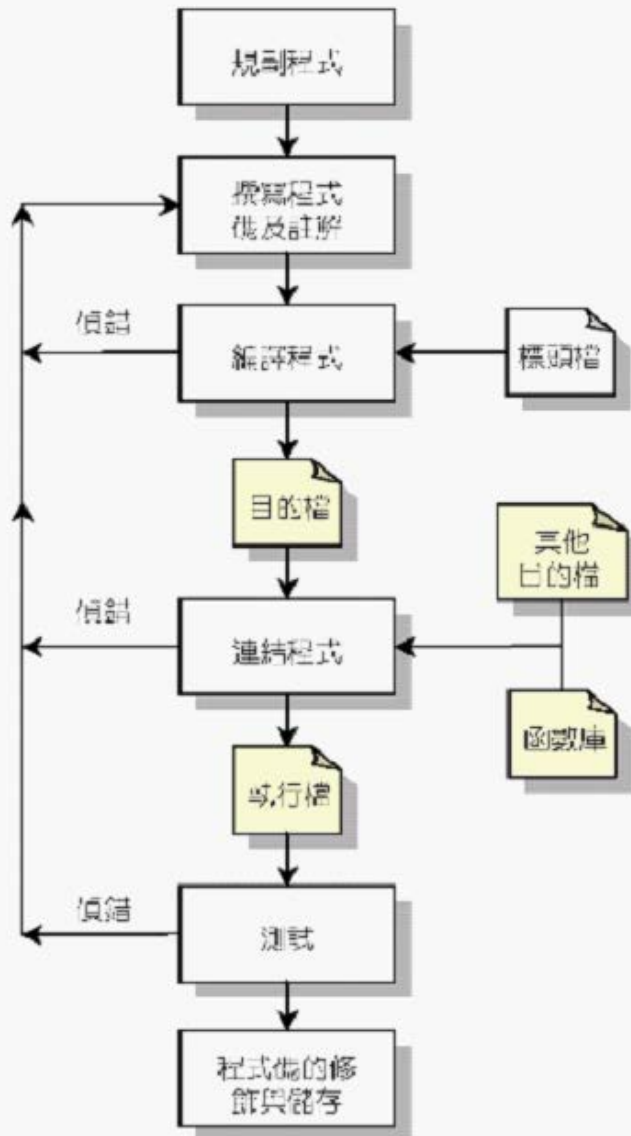
偵錯與測試

語意錯誤 (semantic error)

語法錯誤 (syntax error)

程式碼的修飾與儲存





- 撰寫程式時，應該將欲傳達給閱讀者的訊息寫成簡潔的註解放在原始程式內。
 - `/*多行註解*/`
 - `//單行註解`
- 好用的「固定格式」
 - 善用「Tab鍵」對齊

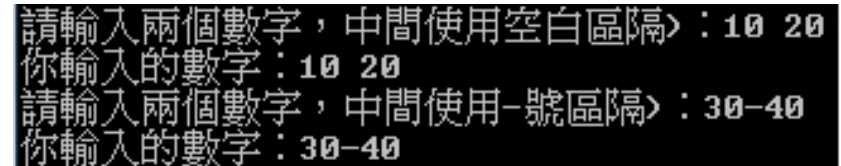
```
#include <stdio.h>
int main(void)
{
    (Tab鍵) printf("%d", 15 + 37);
    (Tab鍵) return 0 ;
}
```

```
#include <stdio.h>
int main (void)
{
    int number1,number2;

    printf("請輸入兩個數字，中間使用空白區隔):");
    scanf("%d %d", &number1, &number2);
    printf("你輸入的數字: %d %d\n", number1, number2);

    printf("請輸入兩個數字，中間使用-號區隔):");
    scanf("%d-%d", &number1, &number2);
    printf("你輸入的數字: %d-%d\n", number1, number2);

    return 0;
}
```



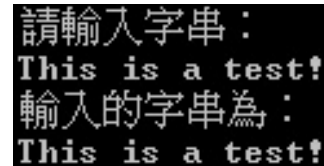
```
請輸入兩個數字，中間使用空白區隔> :10 20
你輸入的數字:10 20
請輸入兩個數字，中間使用-號區隔> :30-40
你輸入的數字:30-40
```

- **printf** 函式負責輸出
 - ex: `printf ("%d %d\n", 10 + 22, 10-2);`
 - **printf** 表示顯示，() 中則是引數。引數有兩個以上的話，必須用逗號「**,**」來區隔，**%d** 用以指定「以十進位表示後面的引數值」即**32**和**8**
 - **\n** 代表顯示後換行
 - 每段最後都要有「**;**」才算完整的敘述，相當於中文文法中的句點「**。**」。
 - 在**C**中標準輸入輸出是由**stdio** 提供，即**#include <stdio.h>**
- **scanf** 函式負責輸入用來讀取鍵盤輸入的字串
 - ex: `scanf("%d", &number);`
 - **& number**, 在變數名稱(**number**)前一定要加上**&**記號
 - 在**C**中標準輸入輸出是由**stdio** 提供，即**#include <stdio.h>**


```
#include <stdio.h>
int main(void)
{
    char str[20];

    puts("請輸入字串：");
    gets(str);
    puts("輸入的字串為：");
    puts(str);

    return 0;
}
```



```
請輸入字串：
This is a test!
輸入的字串為：
This is a test!
```

- **puts**顯示函式
 - **puts**函式只能接受一個引數。
 - **puts**函式用以在顯示引數後的字串後，予以換行。
 - **put("ABC")** 等於 **printf("ABC\n")**
- **gets**輸入函式
 - **gets**函式只能接受一個引數。
 - **gets**函式用以在輸入引數後的字串後，予以換行。

- 資料型態(Data type)
 1. 整數 (Integer)
 - 有short, int和long 型態的長度越長，表示可表示的整數值範圍越大
 - short - 2 bytes
 - int - 4 bytes
 - long - 4 bytes
 2. 浮點數 (Float)
 - 用來表示小數值，有float、double與long double
 - float - 4 bytes
 - double - 8 bytes
 - long double - 8 bytes
 3. 字元 (Character)
 - 用來儲存字元，長度為1個位元組，其字元編碼主要依ASCII表而來
 - char - 1 bytes

- 資料型態(Data type)

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    printf("型態\t\t大小(bytes)\n");
    printf("short\t\t%d\n", sizeof(short));
    printf("int\t\t%d\n", sizeof(int));
    printf("long\t\t%d\n", sizeof(long));
    printf("float\t\t%d\n", sizeof(float));
    printf("double\t\t%d\n", sizeof(double));
    printf("long double\t\t%d\n", sizeof(long double));
    printf("char\t\t%d\n", sizeof(char));
    return 0;
}
```

型態	大小(bytes)
short	2
int	4
long	4
float	4
double	8
long double	8
char	1

資料型別	printf 的轉換指定詞	scanf 的轉換指定詞
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
unsigned short	%hu	%hu
short	%hd	%hd
char	%c	%c

- 資料型態(Data type) 和變數(Variable)

```
#include <stdio.h>
int main(void)
{
    int ageOfStudent;
    double scoreOfStudent;
    char levelOfStudent;

    printf("\n年級\t得分\t等級\n");
    printf("%d\t%d\t%d\n", ageOfStudent, scoreOfStudent, levelOfStudent);

    ageOfStudent = 5;
    scoreOfStudent = 80.0;
    levelOfStudent = 'B';

    printf("\n年級\t得分\t等級\n");
    printf("%d\t%.2f\t%c\n", ageOfStudent, scoreOfStudent, levelOfStudent);

    return 0;
}
```

年級	得分	等級
9383976	9383976	9383972
年級	得分	等級
5	80.00	B

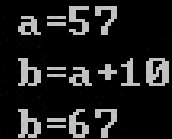
- 因為首先宣告變數，但並沒有初始其值，所以Visual Studio 的編譯程式會出現錯誤訊息
- 在指定變數的值之後，顯示變數值時就會出現指定的資料
- 在宣告變數時使用const關鍵字來限定，如果程式中有其它程式碼試圖改變這個變數，編譯器會先檢查出這個錯誤
 - 例如：**const int PI = 3.14; //圓周率部會改變**

- 程式語言的運算邏輯
 - 「=」記號代表以右邊的值代入左邊變數的位置

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int a,b;

    a=57;
    printf("a=%d\n",a);
    b=a+10;
    printf("b=a+10\n");
    printf("b=%d\n",b);

    system("pause");
    return 0;
}
```



```
a=57
b=a+10
b=67
```

- `a = 57;` `//a`被指定成57
- `b = a+10;` `//`取代`a`的值加上10來代入到`b`
- `//b`的值就變成67

- 程式範例：將兩個數相加

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void)
4 {
5     int integer1;
6     int integer2;
7     int sum;
8
9     printf("Enter first integer\n");
10    scanf("%d", &integer1);
11
12    printf("Enter second integer\n");
13    scanf("%d", &integer2);
14
15    sum=integer1+integer2;
16
17    printf("Sum is %d\n",sum);
18
19    system("pause");
20    return 0;
21 }
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

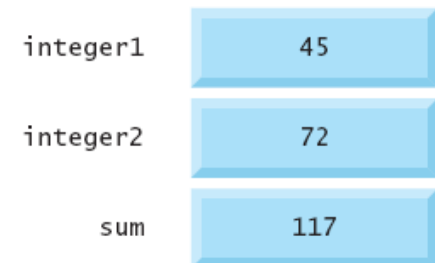


圖 2.8 在計算執行之後的記憶體情形

- C的算術運算子

C 的運算	算術運算子 (arithmetic operator)	代數運算式	C 運算式
加法	+	$f + 7$	<code>f + 7</code>
減法	-	$p - c$	<code>p - c</code>
乘法	*	bm	<code>b * m</code>
除法	/	x / y or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
模數除法	%	$r \bmod s$	<code>r % s</code>

圖 2.9 算術運算子

- C的運算子執行的順序

步驟 1 $y = 2 * 5 * 5 + 3 * 5 + 7;$ (最左邊的乘法先運算)

$2 * 5$ is 10

步驟 2 $y = 10 * 5 + 3 * 5 + 7;$ (最左邊的乘法先運算)

$10 * 5$ is 50

步驟 3 $y = 50 + 3 * 5 + 7;$ (乘法在加法之前)

$3 * 5$ is 15

步驟 4 $y = 50 + 15 + 7;$ (最左邊的加法)

$50 + 15$ is 65

步驟 5 $y = 65 + 7;$ (最後一個加法)

$65 + 7$ is 72

步驟 6 $y = 72$ (最後一個運算—把72放入y中)

- 程式範例：整數與浮點數的資料型態

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void)
4 {
5     int nx;
6     double dx;
7
8     nx=9.99;
9     dx=9.99;
10
11     printf("int 型別變數nx之值： %d\n",nx);
12     printf("int 型別變數nx/2之值： %d\n",nx/2);
13
14     printf("double 型別變數dx之值： %f\n",dx);
15     printf("double 型別變數dx/2.0之值： %f\n",dx/2.0);
16
17     system("pause");
18     return 0;
19 }
```

```
int 型別變數nx之值：9
int 型別變數nx/2之值：4
double 型別變數dx之值：9.990000
double 型別變數dx/2.0之值：4.995000
```

HW1: 練習設計計算機

```
c:\users\administrator\documents\visual st
input A: 20
input B: 33
function: 1:+, 2:-, 3:*, 4:/ 1
answer is : 53
請按任意鍵繼續 . . .
```

- 程式範例：整數與浮點數的資料型態

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void)
4 {
5     int n1,n2,n3,n4;
6     double d1,d2,d3,d4;
7
8     n1=5/2;
9     n2=5.0/2.0;
10    n3=5.0/2;
11    n4=5/2.0;
12
13    d1=5/2;
14    d2=5.0/2.0;
15    d3=5.0/2;
16    d4=5/2.0;
17
18    printf("n1=%d\n",n1);
19    printf("n2=%d\n",n2);
20    printf("n3=%d\n",n3);
21    printf("n4=%d\n",n4);
22    printf("d1=%f\n",d1);
23    printf("d2=%f\n",d2);
24    printf("d3=%f\n",d3);
25    printf("d4=%f\n",d4);
26
27    system("pause");
28    return 0;
29 }
```

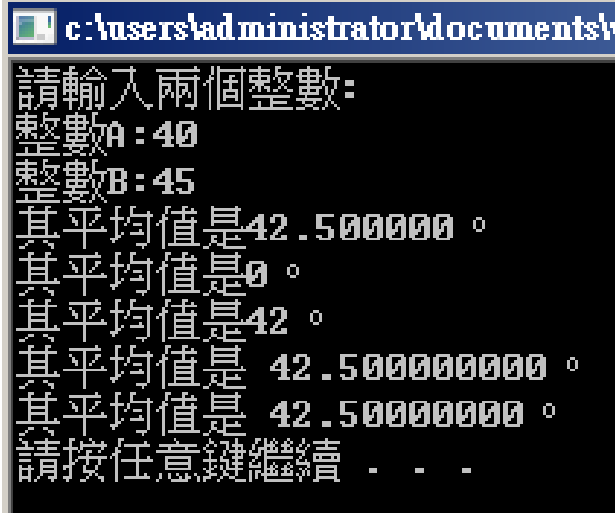
```
n1=2
n2=2
n3=2
n4=2
d1=2.000000
d2=2.500000
d3=2.500000
d4=2.500000
```

- 程式範例: 資料型態(Data type) 的轉換

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void){
5     int na, nb;
6     puts("請輸入兩個整數:");
7     printf("整數A:"); scanf("%d", &na);
8     printf("整數B:"); scanf("%d", &nb);
9
10    printf("其平均值是%f。\\n", (double)(na + nb)/2 );
11    printf("其平均值是%d。\\n", (double)(na + nb)/2 );
12    printf("其平均值是%d。\\n", (int)(na + nb)/2 );
13    printf("其平均值是% 9.9f。\\n", (double)(na + nb)/2 );
14    printf("其平均值是% 07.8f。\\n", (double)(na + nb)/2 );
15
16    system("pause");
17 }

```



```

c:\users\administrator\documents\w
請輸入兩個整數:
整數A:40
整數B:45
其平均值是42.500000。
其平均值是42。
其平均值是42。
其平均值是 42.500000000。
其平均值是 42.50000000。
請按任意鍵繼續 - - -

```

- printf("其平均值是%f。\\n", (double)(na + nb) / 2);
 - 若假設整數A為40，整數B為45，則執行結果已轉換成double型態為42.500000
- printf("其平均值是%09.9f。\\n", (double)(na + nb) / 2);
 - 0 flag: 若為0，則數值前面有空格時，以0填滿。若沒寫0，則該數值前面留白
 - 小數點前: 若為9，則輸出時至少必須有9位數。如不特別指定或實際數值超過指定位數時，則只輸出足以表示該數值的必要位數。
 - 小數點後: 若為8，精確度指定輸出8位數。如不指定，整數型態視同指定為1、浮點數型態視同指定為6。
- 轉換指令字:
 - d 以十進位輸出int型態的整數
 - f 以十進位輸出double型態的浮點數
 - printf("%.2f\\n", 3.446); /* prints 3.45 */
 - printf("%.1f\\n", 3.446); /* prints 3.4 */

- C的條件運算

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     int num1;
7     int num2;
8
9     printf("Enter two integers, and I will tell you\n");
10    printf("the relationships they satisfy: ");
11
12    scanf("%d%d", &num1, &num2);
13
14    if(num1==num2)
15    {
16        printf("%d is equal to %d\n", num1, num2);
17    }
18
19    if(num1 != num2)
20    {
21        printf("%d is not equal to %d\n", num1, num2);
22    }
23
24    if(num1 < num2)
25    {
26        printf("%d is less than %d\n", num1, num2);
27    }

```

```

28
29     if(num1 > num2)
30     {
31         printf("%d is greater than %d\n", num1, num2);
32     }
33
34     if(num1 <= num2)
35     {
36         printf("%d is less than or equal to %d\n", num1, num2);
37     }
38
39     if(num1 >= num2)
40     {
41         printf("%d is greater than or equal to %d\n", num1, num2);
42     }
43
44     system("pause");
45     return 0;
46 }

```

```

c:\users\administrator\documents\visual studio 2010\Project
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
請按任意鍵繼續 . . .

```

```

c:\users\administrator\documents\visual studio 2010\Project
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
請按任意鍵繼續 . . .

```

```

c:\users\administrator\documents\visual studio 2010\Project
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
請按任意鍵繼續 . . .

```

- C的條件運算

標準代數的等號或關係運算子	C 的等號或關係運算子	C 的條件式範例	C 條件式的意義
等號運算子			
=	==	x == y	x 等於 y
≠	!=	x != y	x 不等於 y
關係運算子			
>	>	x > y	x 大於 y
<	<	x < y	x 小於 y
≥	>=	x >= y	x 大於或等於 y
≤	<=	x <= y	x 小於或等於 y

圖 2.12 等號運算子和關係運算子