Chapter

3

變數與常數

本章大綱

本章是介紹程式設計的一些基本觀念，例如：程式保留字、識別字、資料型態、變數與常數的宣告與有效範圍、變數資料型態的轉換。以上內容或許有點瑣碎，也有點枯燥，但卻又很重要，就像練功前的蹲馬步，所以請讀者們耐心讀完。

3-1　保留字與識別字

保留字(Keywords)

保留字（又稱關鍵字）是任一程式語言已事先賦予某一識別字（可識別的文字或字串，稱爲識別字）一個特別意義，所以程式設計者不得再重複賦予不同的用途。例如 if 已賦予決策敘述，程式設計者當然不得再定義 if 爲另外的用途，以下是 C 語言的保留字，請於 https://en.cppreference.com/w 點選『C Keywords』。

auto	float	signed	_Alignas (since C11)
break	for	sizeof	_Alignof (since C11)
case	goto	static	_Atomic (since C11)
char	if	struct	_Bool (since C99)
const	inline (since C99)	switch	_Complex (since C99)
continue	int	typedef	_Generic (since C11)
default	long	union	_Imaginary (since C99)
do	register	unsigned	_Noreturn (since C11)
double	restrict (since C99)	void	_Static_assert (since C11)
else	return	volatile	_Thread_local (since C11)
enum	short	while	
extern			

識別字(Identifiers)

眞實的世界裏，每個人、事及物都有一個名稱，程式設計亦不例外，於程式設計時我們必須爲每一個變數、常數及函式命名，以上所有變數、常數及函式等名稱，統稱爲程式語言的識別字。C 語言的識別字命名規則如下：

1. 識別字必須是以字母（大小寫的 A 至 Z）或底線 (_) 開頭。例如，以下是一些合法的識別字。

```
a
i
sum
_sum
Income
```

以下是一些非法的識別字。

```
7eleven     /*  不能由數字開頭  */
%as         /*  不能由符號開頭  */
```

2. 識別字由字母開頭後，僅可由字母、底線及數字組合而成，但不得包含空白。例如，以下是一些合法的識別字。

```
a123
a123b
_a_b
```

以下是一些非法的識別字。

```
A=          /*  不能含有 = 號 */
sum!        /*  不能含有 ! 號 */
Age#3       /*  不能含有 # 號 */
a c         /*  不能含空白    */
c+3         /*  不得含有加號  */
```

3. 識別字的長度不可以超過 32 個字元。但是，如果做爲外部函式名稱用時 , 則不可超過 6 個字元，例如，下列的二個外部函式呼叫 , 會呼叫到同一個外部函式 :

```
displayAmount();
displayBalance();
```

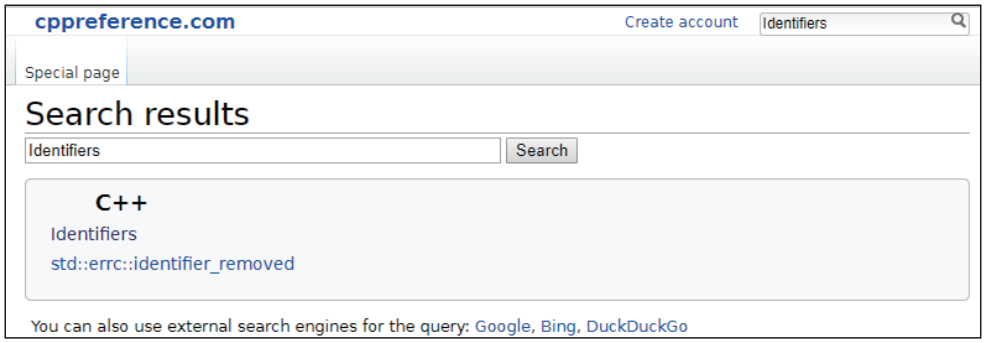
4. 識別字的大小寫均視爲不同，例如 Score、score 及 SCORE 皆代表不同的識別字。

5. 識別字不得使用保留字，如 if、for 等。但是，if1、fora 等則可使用。

6. 識別字要用有意義的單字，例如 StudentNumber 或 AverageIncome。除非有效範圍很小（或稱生命週期極短）的變數才用 x、i 或 a 等當識別字，也千萬不要用 k23erp 等這種沒意義又難記的識別字。

7. 識別字有多個單字時，中間可以加上底線（_），例如上例的 StudentNumber 可寫成 Student_Number，若擔心打字不靈光亦可寫成 Stu_Num、stu_num、StuNum 或 stumum，其中 StuNum 又稱駝峰表示法，因爲大寫字母看起來像駱駝駝峰一樣，可以避免鍵入底線的困擾、且提昇閱讀效率。

8. 關於識別字的詳細說明，請輸入『Identifiers』，如下圖：



請於上圖點選『Identifiers』，或於 https://en.cppreference.com/w/c 先點選『Basic concepts』，再點選『Identifiers』。

3-2 資料種類與資料型態

資料種類

程式設計的主要工作是處理人類量化的資料，C 語言所能處理的資料種類分別有數值 (含整數、長整數及浮點數)、字元與字串等。

整數常數(Integers)

C 語言可以處理的整數常數有三種進位方式，分別是十進位 (Decimal)、十六進位 (Hexadecimal) 及八進位 (Octal)。其中十進位則以我們平常書寫數字的方式即可。十六進位應以 0X 或 0x 開頭，例如 0xA 或 0XB 均爲十六進制，分別代表十進位的 10 與 11；八進位則應以 0 開頭，例如，072 則爲八進位，等於十進位的 58。

浮點常數(Floating-point literals)

數字中含有小數點或指數的稱爲浮點數或實數。以指數爲例，E 或 e 表示 10 的次方，例如 0.0023、2.3E-3 及 2.3e-3 都是表

示相同的浮點數；又例如 2.3E+2 則代表 230。浮點數可使用標準寫法或科學符號法表示，例如 321.123 即爲標準寫法，1.23e+4 即爲科學符號表示法。

字元常數(Character literals)

使用單引號『'』圍住的單一字元，稱爲字元常數，例如 'A' 或 'a' 等。C 語言使用 8 bits 表示一個字元，所以可表示 256 個字元，包含大小寫英文字母、數字、標點符號及其它特殊符號。下表是 ASCII 表，僅用七位元即可表示所有電腦可用字元。

https://en.cppreference.com/w/cpp/language/ascii

ASCII Chart

The following chart contains all 128 ASCII decimal **(dec)**, octal **(oct)**, hexadecimal **(hex)** and character **(ch)** codes.

dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch
0	0	00	**NUL** (null)	32	40	20	(space)	64	100	40	**@**	96	140	60	`
1	1	01	**SOH** (start of header)	33	41	21	**!**	65	101	41	**A**	97	141	61	**a**
2	2	02	**STX** (start of text)	34	42	22	**"**	66	102	42	**B**	98	142	62	**b**
3	3	03	**ETX** (end of text)	35	43	23	**#**	67	103	43	**C**	99	143	63	**c**
4	4	04	**EOT** (end of transmission)	36	44	24	**$**	68	104	44	**D**	100	144	64	**d**
5	5	05	**ENQ** (enquiry)	37	45	25	**%**	69	105	45	**E**	101	145	65	**e**
6	6	06	**ACK** (acknowledge)	38	46	26	**&**	70	106	46	**F**	102	146	66	**f**
7	7	07	**BEL** (bell)	39	47	27	**'**	71	107	47	**G**	103	147	67	**g**
8	10	08	**BS** (backspace)	40	50	28	**(**	72	110	48	**H**	104	150	68	**h**
9	11	09	**HT** (horizontal tab)	41	51	29	**)**	73	111	49	**I**	105	151	69	**i**
10	12	0a	**LF** (line feed - new line)	42	52	2a	*****	74	112	4a	**J**	106	152	6a	**j**
11	13	0b	**VT** (vertical tab)	43	53	2b	**+**	75	113	4b	**K**	107	153	6b	**k**
12	14	0c	**FF** (form feed - new page)	44	54	2c	**,**	76	114	4c	**L**	108	154	6c	**l**
13	15	0d	**CR** (carriage return)	45	55	2d	**-**	77	115	4d	**M**	109	155	6d	**m**
14	16	0e	**SO** (shift out)	46	56	2e	**.**	78	116	4e	**N**	110	156	6e	**n**
15	17	0f	**SI** (shift in)	47	57	2f	**/**	79	117	4f	**O**	111	157	6f	**o**
16	20	10	**DLE** (data link escape)	48	60	30	**0**	80	120	50	**P**	112	160	70	**p**
17	21	11	**DC1** (device control 1)	49	61	31	**1**	81	121	51	**Q**	113	161	71	**q**
18	22	12	**DC2** (device control 2)	50	62	32	**2**	82	122	52	**R**	114	162	72	**r**
19	23	13	**DC3** (device control 3)	51	63	33	**3**	83	123	53	**S**	115	163	73	**s**
20	24	14	**DC4** (device control 4)	52	64	34	**4**	84	124	54	**T**	116	164	74	**t**
21	25	15	**NAK** (negative acknowledge)	53	65	35	**5**	85	125	55	**U**	117	165	75	**u**
22	26	16	**SYN** (synchronous idle)	54	66	36	**6**	86	126	56	**V**	118	166	76	**v**
23	27	17	**ETB** (end of transmission block)	55	67	37	**7**	87	127	57	**W**	119	167	77	**w**
24	30	18	**CAN** (cancel)	56	70	38	**8**	88	130	58	**X**	120	170	78	**x**
25	31	19	**EM** (end of medium)	57	71	39	**9**	89	131	59	**Y**	121	171	79	**y**
26	32	1a	**SUB** (substitute)	58	72	3a	**:**	90	132	5a	**Z**	122	172	7a	**z**
27	33	1b	**ESC** (escape)	59	73	3b	**;**	91	133	5b	**[**	123	173	7b	**{**
28	34	1c	**FS** (file separator)	60	74	3c	**<**	92	134	5c	****	124	174	7c	**|**
29	35	1d	**GS** (group separator)	61	75	3d	**=**	93	135	5d	**]**	125	175	7d	**}**
30	36	1e	**RS** (record separator)	62	76	3e	**>**	94	136	5e	**^**	126	176	7e	**~**
31	37	1f	**US** (unit separator)	63	77	3f	**?**	95	137	5f	**_**	127	177	7f	**DEL** (delete)

字串

C 語言中，字串以雙引號『" "』所圍住，例如 ,"Gwosheng"," 台灣 " 等。

布林值

C 語言使用 1 代表布林的 true，0 代表布林的 false。

資料型態 (Data Type)

前面已介紹人類經常使用的一些資料常數，每一種資料常數均需要不同大小的記憶體儲存，電腦爲了有效率的儲存與處理這些資料，所以有資料型態的規劃，也就是大的資料用大盒子裝，小的資料用小盒子裝，如此即可節省記憶體，並加快處理效率。反過來說，若不分資料大小，通通用大盒子裝資料，那將會非常浪費記憶體，也拖慢執行效率。

C 語言的資料型態首先分爲內建資料型態及複合資料型態，內建資料型態再分爲數值 (int、long、long long、float、double、long double) 、字元 (char) 資料型態及函式專用型態 (void)，複合資料型態再分爲陣列 (array)、結構 (struct) 等型態。

數值資料型態

下表是 C 語言的數值資料型態，unsigned 表無號整數，即非負數整數：

數值資料型態	佔用記憶體的大小(位元)	所能代表的數值的範圍
short	16	-32,768 ~ +32,767
int	16	-32,768 ~ +32,767
long	32	-2,147,483,648 ~ +2,147,483,647
unsigned short	16	0 ~ 65,535
unsigned int	16	0 ~ 65,535
unsigned long	32	0 ~ 4,294,967,295
float	32	$3.4x10^{-38}$ ~ $3.4x10^{38}$
double	64	$1.7x10^{-308}$ ~ $1.7x10^{308}$
long double	80	$3.4x10^{-4932}$ ~ $3.4x10^{4932}$

上表有點複雜，所以取常用資料型態如下：本書就用這三種就好。

數值資料型態	佔用記憶體的大小(位元)	所能代表的數值的範圍
int	16	-32,768 ~ +32,767
long	32	-2,147,483,648 ~ +2,147,483,647
double	64	$1.7x10^{-308}$ ~ $1.7x10^{308}$（負數亦同）

整數

我們可以使用 signed(有號) 與 unsigned(無號) 的 short、int 及 long 等六種型態來儲存整數，但眞是太複雜了，本書整數一律使用整數預設型態的 int 型態，所能代表的數值的範圍是 -32,768 ~ +32,767。若超出此範圍則用 long。例如，一天的秒數是 86400，已經超過整數 int 的範圍，那就要用 long 表示。long 型態的範圍就大了，可儲存的範圍是 -2,147,483,648 ~ +2,147,483,647，但也要付出代價，就是要佔用 4 個位元組，32bits 的空間，而且佔的空間多，搬運也較費時，所以選用適當大小的資料型態就很重要了。

浮點數

我們可以使用 float、double 及 long double 等三種型態來儲存浮點常數。浮點數的預設值是 double，但若要強制使用 float，則應於數值後面加上英文字母的 F 或 f，例如，3.1415926F 或 3.1415926f。但若要使用 long double，則於數值後面加上 L 或 l，例如，3.1415926L 或 3.1415926l。本書採用折衷的方式，浮點數一律使用 double。(因爲 C 語言大部分數學函式傳回 double 型態)

字元資料型態

C 語言使用 char 資料型態來儲存一個 ASCII 字元，因此，C 語言中的 char 型態是 8 位元長。

布林型態

C 語言設計之初並沒有布林型態，後來才新增 _Bool 布林型態。

字串資料型態

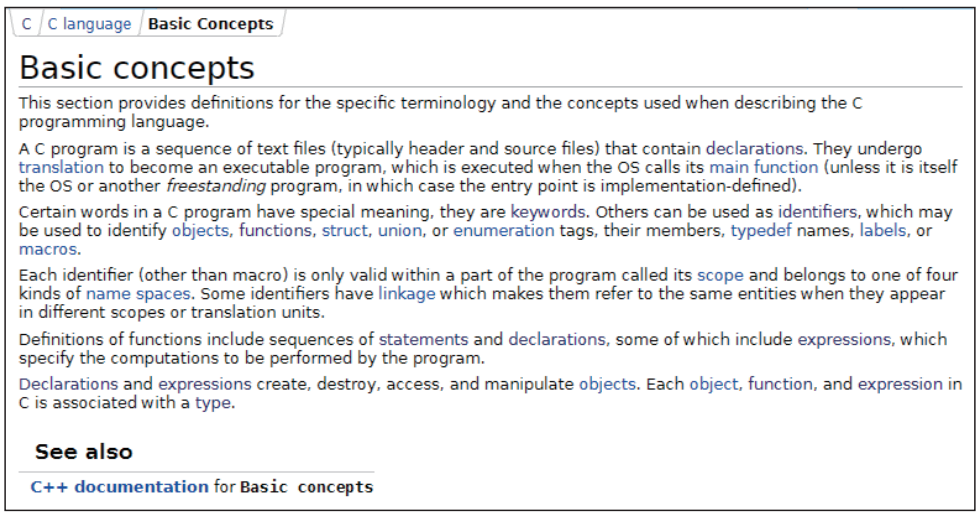
C 語言並無字串資料型態，而是使用字元陣列，例如，

```
chara[10]="gwosheng";
```

關於字元陣列，請看 7_3 節。

自我練習

1. 請 於 https://en.cppreference.com/w 點 選『Basic concepts』，畫面如下圖：


C / C language / Basic Concepts

Basic concepts

This section provides definitions for the specific terminology and the concepts used when describing the C programming language.

A C program is a sequence of text files (typically header and source files) that contain declarations. They undergo translation to become an executable program, which is executed when the OS calls its main function (unless it is itself the OS or another *freestanding* program, in which case the entry point is implementation-defined).

Certain words in a C program have special meaning, they are keywords. Others can be used as identifiers, which may be used to identify objects, functions, struct, union, or enumeration tags, their members, typedef names, labels, or macros.

Each identifier (other than macro) is only valid within a part of the program called its scope and belongs to one of four kinds of name spaces. Some identifiers have linkage which makes them refer to the same entities when they appear in different scopes or translation units.

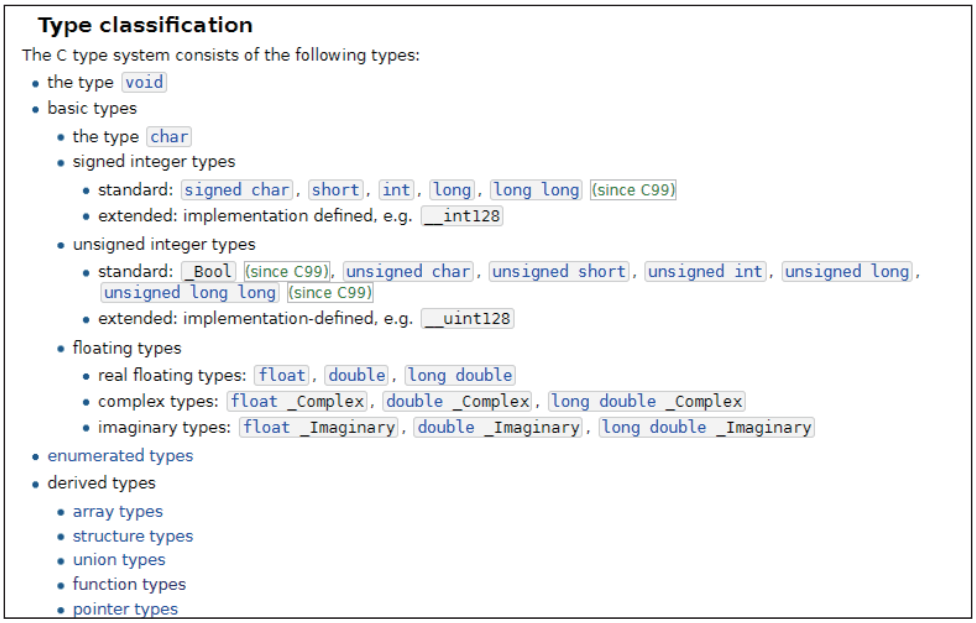
Definitions of functions include sequences of statements and declarations, some of which include expressions, which specify the computations to be performed by the program.

Declarations and expressions create, destroy, access, and manipulate objects. Each object, function, and expression in C is associated with a type.

See also

C++ documentation for Basic concepts

繼續於上圖點選『Type』，即可得所有資料型態的詳細資料，如下圖，往下捲動，還有所有資料型態的表示範圍。



Type classification

The C type system consists of the following types:

- the type `void`
- basic types
  - the type `char`
  - signed integer types
    - standard: `signed char`, `short`, `int`, `long`, `long long` (since C99)
    - extended: implementation defined, e.g. `__int128`
  - unsigned integer types
    - standard: `_Bool` (since C99), `unsigned char`, `unsigned short`, `unsigned int`, `unsigned long`, `unsigned long long` (since C99)
    - extended: implementation-defined, e.g. `__uint128`
  - floating types
    - real floating types: `float`, `double`, `long double`
    - complex types: `float _Complex`, `double _Complex`, `long double _Complex`
    - imaginary types: `float _Imaginary`, `double _Imaginary`, `long double _Imaginary`
- enumerated types
- derived types
  - array types
  - structure types
  - union types
  - function types
  - pointer types

跳脫字元序列

字元中的單引號（'）、雙引號（"）及反斜線（\）在 ASCII 均已有定義其功能，若您一定要使用這些字元，則應於使用前先加一個反斜線（\），以跳脫其原先所定義的功能，此稱爲跳脫字元 (Escape Sequence)。下表是一些常用的跳脫字元：

字元	跳脫字元序列
單引號	\'
雙引號	\"
反斜線	\\

例如：

```
char a[]="Gwosheng";
printf("%s\n",a);
```

結果是 Gwosheng。又例如，

```
char b[]="\"Gwosheng\"";
printf("%s\n",b);
```

結果是 "Gwosheng"。

自我練習

鍵入以下程式，並觀察執行結果。

題號	程式	結果

1.	`#include <stdio.h>`
	`#include <stdlib.h>`
	`int main(int argc, char *argv[]) {`
	`  char a[]="Gwosheng";`
	`  printf("%s\n",a);`
	`  char b[]="\"Gwosheng\"";`
	`  printf("%s\n",b);`
	`  return 0;`
	`}`	

3-3 變數和常數的宣告

變數宣告

變數的功能是用來輸入、處理及儲存外界的資料，而變數在使用以前均要事先宣告才可使用。在一些舊式的 Basic 語言中，變數使用前並不需要事先宣告，卻也帶來極大的困擾。以下敘述即爲變數未宣告的結果，編譯器便無法回應使用者在拼字上的錯誤，而造成除錯上的困難。

```
student=studend+1;
```

上式若事先宣告 student 如下：

```
int student;
```

則編譯器遇到 studend 時，便會出現 studend 未宣告的錯誤訊息，提醒使用者補宣告或注意拼字錯誤。其次，變數宣告的優點是可配置恰當的記憶體而提高資料的處理效率。例如，有些變數的值域僅爲整數，則不用宣告爲 float 或 double。此即爲小東西用小箱子裝，大東西用大箱子裝，才能有效運用空間與提升搬運效率。C 語言的變數宣告語法如下：

```
資料型態 變數名稱[=初值];
```

例如，

```
int a;
```

即是宣告變數 a 爲 int 型態。又例如，

```
char b,c;
```

則是宣告變數 b,c 爲 char 型態。變數的宣告亦可連同初值一起設定，如下所示：

```
double d = 30.2;
```

宣告 d 是 double 型態，並設其初值是 30.2。以下程式，同時宣告兩個整數，且指派其初值。

```
int f1=3,f2=4;
```

以下程式可宣告變數爲字元陣列，字元陣列就能代表字串。

```
char g[]="Gwosheng";//不用宣告其大小也可
```

以下程式就不行，因爲同一函式範圍內，不能重複宣告同一變數。

```
int a;
char a;
```

變數經過宣告之後，編譯器即會根據該變數的資料型態配置適當的記憶體儲存此變數，所以若要提高程式的執行效率，則應儘量依照資料性質，選擇佔用記憶體較小的資料型態。

自我練習

請鍵入以下程式，編譯、除錯與執行程式。

題號	程式	結果

1.	`#include <stdio.h>`
	`#include <stdlib.h>`
	`int main(int argc, char *argv[]) {`
	`  int 7eleven;  /*  不能由數字開頭  */`
	`  int %as;  /*  不能由符號開頭  */`
	`  int _a;`
	`  int A=;  /*  不能含有 = 號 */`
	`  int sum! ;  /*  不能含有! 號 */`
	`  int Age#3; /*  不能含有 # 號 */`
	`  int a c ;  /*  不能含空白  */`
	`  int c+3 ;  /*  不得含有加號  */`
	`  int if;`
	`  return 0;`
	`}`	

2.	`#include <stdio.h>`
	`#include <stdlib.h>`
	`int main(int argc, char *argv[]) {`
	`  int student=0;`
	`  student=studend+1;`
	`  printf("%d",student);`
	`  int a=3.4;`
	`  char b,c;`
	`  b="aa"`
	`  c="a";`
	`  int d='a';`
	`  double e=3;`
	`  return 0;`
	`}`	

常數符號的宣告

跟變數一樣，常數符號亦需要記憶體儲存位置，與變數不同的是，常數符號正如名稱所示，常數符號在整個程式中都不會，也不能改變其值。程式設計有兩種表示常數的方式，一種是文字式 (Literal)，例如直接以 15 或 3.14159 表示某一常數；另一種是本單元的常數符號式 (Symbolic)，因爲有些數字在程式中會不斷的重複出現，爲了增加程式的可讀性及減少程式鍵入的麻煩，此時即可用一個有意義的符號代替，我們可以利用 const 來定義常數，則該符號的值將永遠保持在你所宣告的符號，程式中任何位置均不可能改變其值，此稱爲常數符號，簡稱常數。const 宣告常數，語法如下：

```
const 型態 常數名稱=常數值;
```

例如，以下程式令 PI=3.14，常數符號通常均用大寫表示。

```
const double PI=3.14;
```

則每次要使用 3.14 時，只要填入 PI 即可。例如，以下程式可計算圓面積。

```
a=3*3*PI；
```

又例如，於銀行利息的計算，利率亦應以常數符號表示，程式中任何位置若需使用此利率時，均應以此常數符號代替，當要調整此利率時，只要在程式的最前面修改此常數符號的值即可。若未使用常數符號統一此值，則因此常數散落程式各地，而必須到處修改，萬一有些地方漏掉未修改，則無法確保此值的一致性。

自我練習

1. 請鍵入以下程式，並觀察執行結果。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
   const double PI=3.14;
   double a;
   a=3*3*PI;
   printf("%f\n",a);
   const int b=4;
   b=b+1;//常數不能改變其值
   printf("%d",b);
   return 0;
}
```

變數與常數的有效範圍

在一個大型程式裏，一個程式是由一或數個函式組合而成。這些函式通常是由許多人合力完成，爲了避免彼此變數互相干擾，例如張三設 stunum=15，李四又設 stunum=20，則必然造成無法預期的錯誤，所以爲了防止變數互相干擾，才有變數的有效範圍（又稱爲變數生命週期），茲將變數的有效範圍敘述如下：

任一變數的宣告，若無特殊聲明，均屬於區域變數，其有效範圍僅止於該變數所在的程式區塊。例如，若宣告於函式中，則它的有效範圍僅止於該函式，其它的函式均無法取得該區域變數的值；若宣告在 main() 的前面，才是全域變數。例如，以下程式，全域的 a 有效範圍是 main( )、setup( )，c 的有效範圍則是 main( )。其次，main() 與 setup() 內雖然都有變數 b，但此 b 在不同函式只是同名而已，它們各自佔有不同的記憶體，且其值不會互相干擾。就如同不同家庭會有相同的人名，但他們一定可配到不同身份證號碼，兩者不會互相干擾。

```
#include <stdio.h>
#include <stdlib.h>
int a=6;
void setup(){
   int b=1;
}
int main(int argc, char *argv[]) {
   int b=2;
   int c=3;
   printf("%d\n",a);//使用全域
   return 0;
}
```

以下程式，變數 a 是全域變數，但又於 setup() 內重複宣告 int a；那也合理，那表示此 a 在 setup() 範圍內是 setup() 專用，此 a 的生命，僅止於 setup()，不會影響全域變數的 a。

```
#include <stdio.h>
#include <stdlib.h>
int a=6;
void setup(){
   int a=3;
   printf("b=%d\n",a);//使用區域
}
int main(int argc, char *argv[]) {
   printf("a=%d\n",a);//使用全域
   setup();
   printf("a=%d\n",a);     //使用全域
   return 0;
}
```

自我練習

請鍵入以下程式，並觀察執行結果。

程式	執行結果

#include <stdio.h>
#include <stdlib.h>
int a=6;
void setup(){
  int b=1;
  a=3;
  printf("b=%d\n",b);
}
int main(int argc, char *argv[]) {
  int b=2;
  int c=3;
  printf("a=%d\n",a);//使用全域
  printf("b=%d\n",b);
  setup();
  printf("b=%d\n",b);
  printf("a=%d\n",a);
  return 0;
}	

3-4 資料型態轉換

變數的資料型態轉換

每一個變數宣告之後，即有屬於自己的型態，往後此變數均只能指派給相同或相近型態的變數儲存，若執行階段欲指定給相近型態的變數儲存，則稱此爲型態轉換。

例如，

```
int a=3,b=4;
b=a;
```

變數型態相同，可以。但是

```
int a;char d[10];//the type is string
d=a;
```

變數型態不同，不行。但是，

```
int a;double b=3.4;
a=b;
```

這樣變數型態相近，可以，這樣會型態轉換。

在C語言中型態轉換又可分爲隱含轉換（Implicit Conversion）與強制轉換（Explicit Conversion），分別說明如下：

隱含轉換（Implicit Conversion）

將值域小的型態轉爲值域大的型態，稱爲自動轉換或轉型（Convert）。此種轉換，系統可自動處理並確保資料不會流失。例如，將 int 轉爲 long，則因後者的值域比前者大，所以可順利的轉換。以下敘述可將型態爲 int 的變數 a 指派給型態爲 long 的變數 b，且原值不會改變。

```
int a=23;
long b;
b=a;
printf("%d",b);
```

結果是 23。

強制轉換（Emplicite Conversion）

將值域大的轉爲值域小的型態（如 long 轉爲 int），則稱此爲強制轉換或稱爲鑄型（cast）。強制轉換的語法如下：

```
變數1  =  (變數1的型態)  變數2 ;
```

例如，以下敘述可將型態是 long 的 c 變數指定給型態是 int 的 d 變數。

```
long c=23;
int d;
d=(int) c;
printf("%d\n",d);
```

結果是 23。強制轉換的風險比較大，有可能資料流失或溢位。例如，以下敘述將 double 型態強制轉換爲 int 型態，將造成小數部份的流失。

```
double e=3.4f;
int f;
f=(int) e;
printf("%d\n",f);//3
```

結果 3。其次，關於數值與字串互轉，則要看第 8 章的字串函式。

自我練習

請鍵入以上程式，並觀察執行結果。

3-5 APCS觀念題

1. 程式執行過程中，若變數發生溢位情形，其主要原因爲何？ 106/01/24

(A) 以有限數目的位元儲存變數值

(B) 電壓不穩定

(C) 作業系統與程式不甚相容

(D) 變數過多導致編譯器無法完全處理

參考解答

（A）

自我練習

本章的另一重點是變數的有效範圍，請鍵入以下程式，並觀察執行結果。

題號	題目	執行結果

1.	const double PI=3.14159;//全域常數
	int a;//全域a
	int main(int argc, char** argv) {
	int b, c;
	a=5;//全域a
	c=5;//main 內 c
	for (int a =2 ;a<=3;a++){//好習慣
	//for 內a,僅for內有效,不會改變全域的a
	//反過來說,若未宣告a，那就慘了，會影響全域a
	}
	if (c>2){
	int a;
	a=1;//if 內a,僅if內有效
	}
	return 0;
	}
	void aa(){
	int c, d;//c,d 僅aa內有效
	c=3;//aa內的c,僅aa內有效
	}
	void bb(){
	a=8;//使用全域a
	}	

2.	#include <stdio.h>
	#include <stdlib.h>
	int a=1;
	void aa(){
	printf("1.a=%d\n",a);
	int a=2;
	printf("2.a=%d\n",a);
	}
	void bb(){
	printf("4.a=%d\n",a);
	a=3;
	}
	int main(int argc, char** argv) {
	aa();
	printf("3.a=%d\n",a);
	bb();
	printf("5.a=%d\n",a);
	return 0;
	}	

3.

```
#include <stdio.h>
#include <stdlib.h>
const double PI=3.14159;//全域常數
int a;//全域a
void aa()
{
int c, d;
  c=3;//aa內的c
}
void bb()
{
  a=8;//全域a
}
int main(int argc, char** argv) {
   int b, c;
   a=5;//全域a
   c=5;//main 內 c
   aa();
   bb();
   for (int a =2 ;a<=3;a++)
      printf("4.for 內的 a=%d\n",a);
   if (c>2){
      int a;
      a=1;
      printf("8.if 內a=%d\n",a);
   }
   return 0;
}
```

4.

```
#include <stdio.h>
#include <stdlib.h>
const double PI=3.14159;//全域常數
int a;//全域a
void aa()
{
int c, d;
  c=3;//aa內的c
printf("1.aa內的c=%d\n",c);
}
void bb()
{
  a=8;//全域a
}
int main(int argc, char** argv) {
   void aa();
```

5.

```
   void bb();
   int b, c;
   a=5;//全域a
   c=5;//main 內 c
   aa();
   printf("2.main 的c=%d\n",c);
   bb();
   printf("3.全域a=%d\n",a);
   for (int a =2 ;a<=3;a++)
     printf("4.for 內的 a=%d\n",a);
   printf("5.全域a=%d\n",a);
   for ( a =2 ;a<=3;a++)//誤用
     printf("6.for 內的 a=%d\n",a);
   printf("7.全域a=%d\n",a);
   if (c>2){
     int a;
     a=1;
     printf("8.if 內a=%d\n",a);
   }
   printf("9.全域a=%d\n",a);
   return 0;
}
```

MEMO