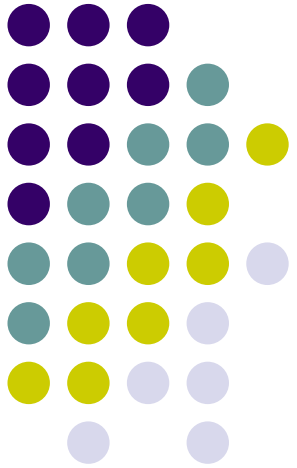
第三章
基本資料型態



常數 vs. 變數

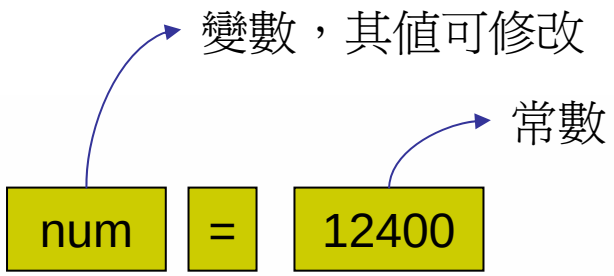
C 語言提供的資料型態

溢位的發生

資料型態間的轉換

變數與常數 (1/2)

- 下面是變數使用的範例：




```
01  /* prog3_1, 變數的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      int num1=12400;       /* 宣告num1為整數變數，並設值為12400 */
07      double num2=5.234;    /* 宣告num2為倍精度浮點數變數,並設值為5.234 */
08
09      printf("%d is an integer\n",num1);   /* 呼叫printf()函數 */
10      printf("%f is a double\n",num2);     /* 呼叫printf()函數 */
11
12      system("pause");
13      return 0;
14  }
```

```
/* prog3_1 OUTPUT---

12400 is an integer
5.234000 is a double
-----------------------*/
```

變數與常數 (2/2)

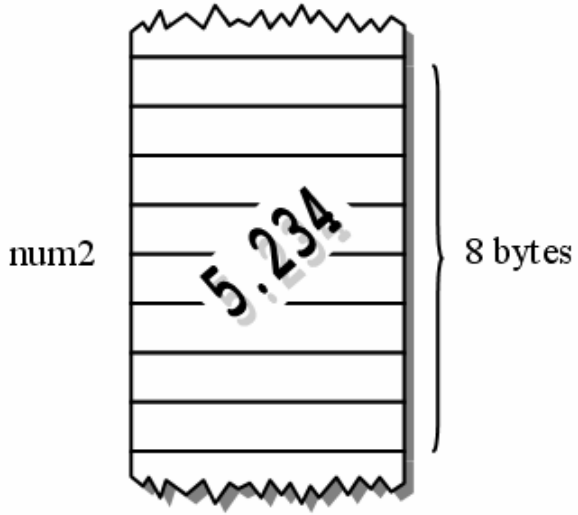
- 變數的宣告與記憶空間的配置

```
int num1=12400;
```

宣告整數變數 num1，
並設值為 12400


```
double num2=5.234;
```

宣告 double 型態的變數
num2，並設值為 5.234

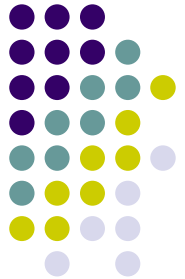


基本資料型態

- 各種基本資料型態變數所佔的記憶體空間及可以表達的資料範圍：

表 3.2.1　C 語言所提供的基本資料型態

資料型態		型態說明	位元組	表示範圍
整數類型	long int	長整數	4	-2147483648 到 2147483647
	int	整數	4	-2147483648 到 2147483647
	short int	短整數	2	-32768 到 32767
	char	字元	1	0 到 255 (256 個字元) -128 到 127
浮點數類型	float	浮點數	4	±1.2e-38 到 ±3.4e38
	double	倍精度浮點數	8	±2.2e-308 到 ±1.8e308



整數型態 int

- 整數型態可分爲
  - 長整數 (long int)
  - 整數 (int)
  - 短整數 (short int)
- 下面爲整數型態宣告的範例：

```
int num=15;           /* 宣告 num 爲整數，並設值爲 15 */
long int num=1240L;   /* 宣告 num 爲長整數，並設值爲 1240L */
short int sum;        /* 宣告 sum 爲短整數 */
```

預設 15 爲 int 常數, L,l 代表long int，LL,ll代表long long，U,u 代表無號，UL…
預設 12.3 爲 double常數，F,f代表浮點數

整數的內部表示方法

- n 位元的二進位系統使用 0, 1 來表示數字

d		a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
0	→	0	0	0	0	0	0	0	0
1	→	0	0	0	0	0	0	0	1
2	→	0	0	0	0	0	0	1	0
…					…				
255	→	1	1	1	1	1	1	1	1

$$d = \sum_{i=0}^{n-1} a_i \, 2^i$$

- 二進位加法

	44	→	0	0	1	0	1	1	0	0
+	13	→	0	0	0	0	1	1	0	1
	57	→	0	0	1	1	1	0	0	1

以硬體直接計算

整數的內部表示方法

- Sign-Magnitude 負數表示方法: 0 代表正數, 1 代表負數

18	→	0	0	0	1	0	0	1	0
-18	→	1	0	0	1	0	0	1	0

不方便: 加法沒有辦法用同樣的硬體來計算

- n 位元 2's Complement (2 的補數) 負數表示方法:

	18	→		0	0	0	1	0	0	1	0	x
	-18	→	0	1	1	1	0	1	1	1	0	2^n-x
+	30	→	0	0	0	0	1	1	1	1	0	y
	12	←	1	0	0	0	0	1	1	0	0	2^n-x+y

隱藏的
輔助位元

無號整數

- 加上 unsigned，整數資料型態便可成爲無號整數
  - 無號整數只存放非負整數，可存放的資料範圍大一倍

表 3.2.2 無號整數的資料型態

資料型態	型態說明	位元組	表示範圍
unsigned long int	無號長整數	4	0 到 4294967295
unsigned int	無號整數	4	0 到 4294967295
unsigned short int	無號短整數	2	0 到 65535

```
unsigned int num = 123U; /* 宣告num爲無號整數,並設值爲 123U */
unsigned short int sum;  /* 宣告sum爲無號短整數 */
```

溢位 (overflow) (1/2)

- 溢位：當儲存的數值超出容許範圍時

```
/* prog3_2 OUTPUT---

s+1= -32768
s+2= -32767
--------------------*/
```

```
01  /* prog3_2, 短整數資料型態的溢位*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     short sum,s=32767;          /* 宣告短整數變數 sum 與 s */
07
08     sum=s+1;
09     printf("s+1= %d\n",sum);     /* 列印出 sum 的值 */
10
11     sum=s+2;
12     printf("s+2= %d\n",sum);     /* 列印出 sum 的值 */
13
14     system("pause");
15     return 0;
16  }
```

```
    0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
+   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
-------------------------------------
    1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

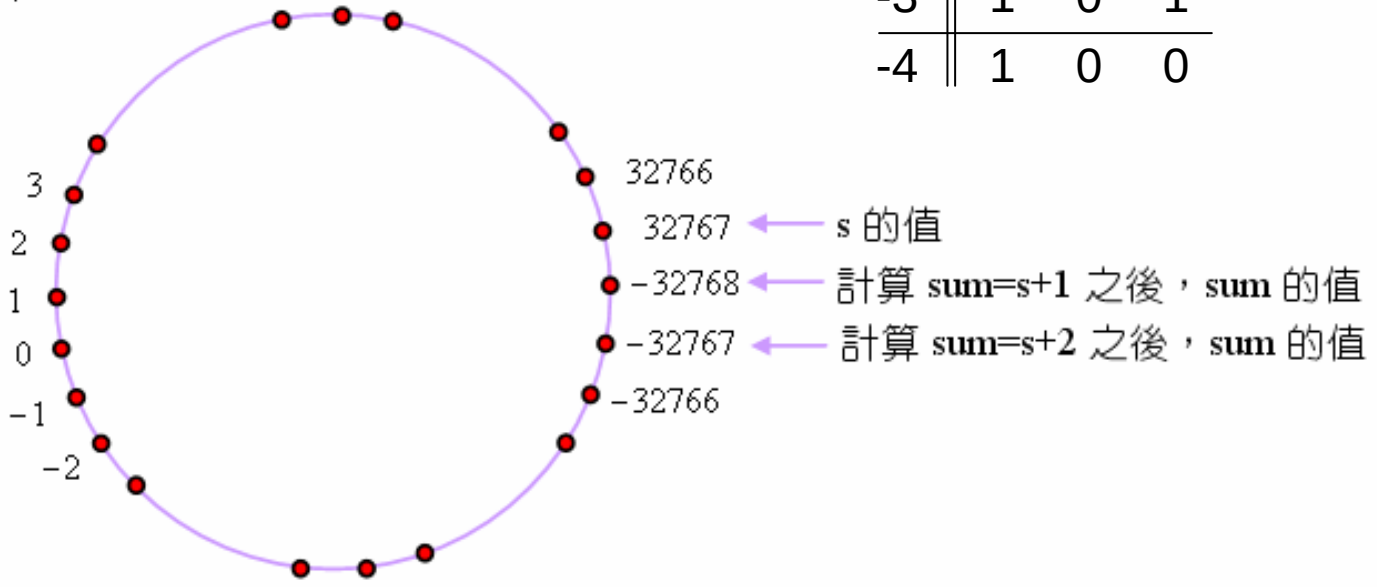
溢位 (overflow) (2/2)

- 下圖說明溢位發生的原因：

2's complement

3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0
-1	1	1	1
-2	1	1	0
-3	1	0	1
-4	1	0	0

```
short sum,s=32767;
sum=s+1;
```



字元型態 char

- 字元型態佔記憶體中 1 個位元組，用來儲存字元的內碼
- 宣告字元變數，並設值給它：

```
char ch;        /* 宣告字元變數ch */
ch='A';         /* 將字元常數'A'設值給字元變數ch */
```

- 在宣告的同時便設定初值

```
char ch='A';  /* 宣告字元變數ch，並將字元常數'A'設值給它 */
char ch=97;   /* 將ch設值爲ASCII碼爲97的字元 */
char ch='7';  /* 將ch設值爲字元常數'7' */
char ch=7;    /* 將ch設值爲設值爲ASCII碼爲7的字元 */
```

- 因爲是 8 個位元，也可以存放 -128 到 127 之間的整數

```
char ch=-56;  /* 宣告字元變數ch，初始化爲-56 */
```

ASCII 字元內碼表

	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DCL	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	SP	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

字元型態的範例 (1/4)

- 下面的程式以不同的格式列印字元變數ch：

```
01  /* prog3_3, 字元的列印*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      char ch='a';                    /* 宣告字元變數 ch，並設值為'a' */
07      printf("ch= %c\n",ch);          /* 印出 ch 的值 */
08      printf("ASCII of ch= %d\n",ch);      /* 印出 ch 的十進位值 */
09
10      system("pause");
11      return 0;
12  }
```

```
/* prog3_3 OUTPUT---

ch= a
ASCII of ch= 97
--------------------*/
```

字元型態的範例 (2/4)

- 以ASCII碼設定字元的範例：

```
01  /* prog3_4, 以 ASCII 碼設定字元 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char ch=90;                  /* 將整數 90 設給字元變數 ch */
07     printf("ch=%c\n",ch);        /* 印出 ch 的值 */
08
09     system("pause");
10     return 0;
11  }
```

```
/* prog3_4 OUTPUT--

ch=Z
----------------------*/
```

字元型態的範例 (3/4)

- 數字字元與其相對應的ASCII碼：

```
01  /* prog3_5, 數字字元與其相對應的 ASCII 碼 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      char ch='2';                    /* 宣告字元變數 ch，並設值為'2' */
07      printf("ch=%c\n",ch);           /* 印出字元變數 ch */
08      printf("the ASCII of ch is %d\n",ch);  /* 印出 ch 的 ASCII 碼 */
09
10      system("pause");
11      return 0;
12  }
```

```
scanf("%c",&ch);
ch=ch - '0';
printf("%d", ch);
------------------
8
8
```

```
/* prog3_5 OUTPUT---
ch=2
The ASCII of ch is 50
---------------------------*/
```

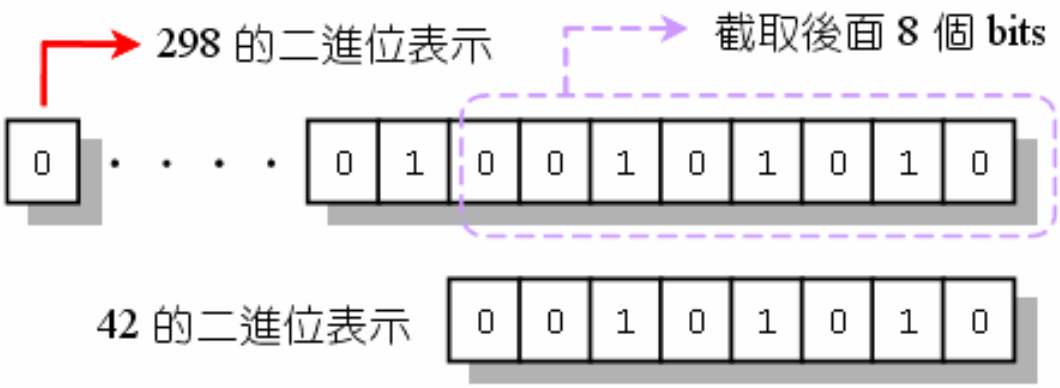
字元型態的範例 (4/4)

- 字元型態溢位的問題：

```
01   /* prog3_6, 字元型態的列印問題*/
02   #include <stdio.h>
03   #include <stdlib.h>
04   int main(void)
05   {
06      int i=298;
07      printf("ASCII(%d)=%c\n",i,i);    /* 印出 ASCII 碼為 i 的字元 */
08
09      system("pause");
10      return 0;
11   }

/* prog3_6 OUTPUT---

ASCII(298)=*
---------------------*/
```



跳脫字元 (1/3)

- 反斜線「\」稱爲跳脫字元 (escape character)
- 反斜線「\」加上控制碼，稱爲跳脫序列

表 3.2.3 常用的跳脫序列

跳脫序列	所代表的意義	十進位 ASCII
\a	警告音(alert)	7
\b	倒退一格(backspace)	8
\n	換行(new line)	10
\r	歸位(carriage return)	13
\0	字串結束字元(null character)	0
\t	跳格(tab)	9
\\	反斜線(backslash)	92
\'	單引號(single quote)	39
\"	雙引號(double quote)	34

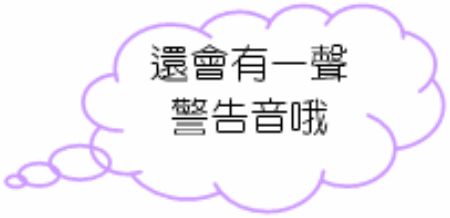
跳脫字元 (2/3)

- 利用跳脫序列控制響鈴：

```
01  /* prog3_7, 跳脫序列的列印*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char beep='\a';          /* 宣告字元變數 beep，並設定其值為'\a' */
07     printf("%c", beep);      /* 響一聲警告音 */
08     printf("ASCII of beep=%d", beep);  /* 印出 beep 的 ASCII 值 */
09
10     system("pause");
11     return 0;
12  }
```

```
/* prog3_7 OUTPUT---

ASCII of beep=7
--------------------*/
```



跳脫字元 (3/3)

- 利用跳脫字元列印特殊符號：

```
01  /* prog3_8, 跳脫序列「\"」的列印 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06      char ch='\"';       /* 宣告字元變數 ch，並設值為'\"' */
07      printf("%cWe are the World%c\n",ch,ch);       /* 印出字串 */
08
09      system("pause");
10      return 0;
11  }

/* prog3_8 OUTPUT---

"We are the World"

----------------------*/
```

浮點數型態 float (1/2)

- 浮點數佔 4 個位元組，有效範圍 $\pm$ 1.2×10^{-38} ~ 3.4×10^{38}

```
float num;              /* 宣告浮點數變數num */
float num=5.46F;        /* 宣告浮點數變數num，並設值爲5.46F */
```

- 要印出浮點數，可用「%f」格式碼
- 要以科學記號的型式列印浮點數，可用「%e」格式碼

浮點數型態 float (2/2)

- 浮點數使用的範例：

```
01  /* prog3_9, 浮點數的列印 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     float num1=123.45F;      /* 宣告 num1 為浮點數，並設值為 123.45F */
07     float num2=4.56E-3F;     /* 宣告 num2 為浮點數，並設值為 4.56E-3F */
08
09     printf("num1=%e\n",num1);    /* 以指數的型態印出 num1 的值 */
10     printf("num2=%f\n",num2);    /* 以浮點數的型態印出 num2 的值 */
11
12     system("pause");
13     return 0;
14  }
```

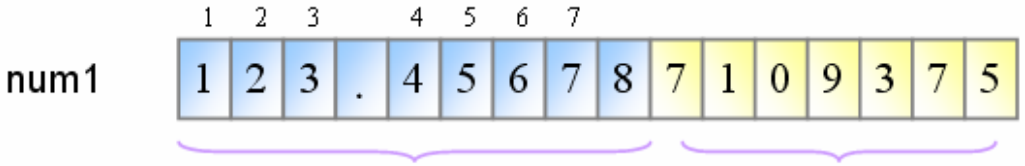
```
/* prog3_9 OUTPUT---

num1=1.234500e+002
num2=0.004560
----------------------*/
```

倍精度浮點數型態 double (1/2)

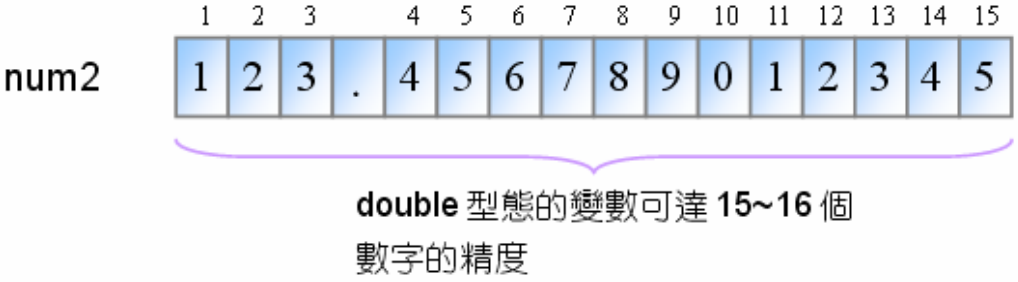
- double 型態佔 8 個位元組，範圍爲± 2.2×10^{-308} ~ 1.8×10^{308}
- float 只有 7~8 位數的精度，double可達 15~16 位數

```
float num1=123.456789012345F;
```



是printf列印的演算法在由二進位換成十進位時一定會產生的,
例如: 二進位 0.10101 =>
十進位 $2^{-1} + 2^{-3} + 2^{-5}$

```
double num2=123.456789012345;
```



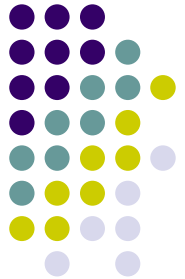
二進位小數點後第 15 位
2^{-15}=0.000030517578125
需要十進位小數點後 15 位數才能精確表示
2^{-23} 需要十進位小數點後 23 位數才能精確表示
2^{-52} 需要十進位小數點後 48 位數才能精確表示

倍精度浮點數型態 double (2/2)

- 下面的範例是 float 與 double 精度的比較 ：

```
01  /* prog3_10, float 與 double 精度的比較 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     float  num1=123.456789012345F; /* 宣告 num1 為 float，並設定初值 */
07     double num2=123.456789012345;  /* 宣告 num2 為 double，並設定初值 */
08
09     printf("num1=%16.12f\n",num1); /* 列印出浮點數 num1 的值 */
10     printf("num2=%16.12f\n",num2); /* 列印出倍精度浮點數 num2 的值 */
11
12     system("pause");
13     return 0;
14  }
```

```
/* prog3_10 OUTPUT-----
num1=123.456787109375
num2=123.456789012345
------------------------*/
```



浮點數表示法的量化誤差

- 實數線上有無窮多個點, double 型態用 64 個位元最多只能表達 2^{64} 種不同的數字, 一定有數字沒有辦法無誤差地表示出來, 只能用最接近的數值來表示
- 更何況 double 型態浮點數要表達的數字含括 -10^{308} - 10^{308} 的範圍, 科學記號表示法會維持相對誤差大約在 $\pm 10^{-15}$, 也就是說對於一個 10^{100} 大小的數字來說, 絕對誤差值大概是 $\pm 10^{85}$, 對於一個 10^{-100} 大小的數字來說, 絕對誤差值大概是 $\pm 10^{-115}$
- 爲什麼需要知道這個? 對寫程式有什麼影響? 後續我們會談到條件判斷以及迴圈, 請務必記得絕對不要使用下面的兩種寫法
  - double x, y;
    ….
    ✗ if (**x == y**) …
    ✗ if (**x != y**) …
  - double x;
    ….
    ✗ for (x=0; **x!=1.0**; x+=0.1)
    …

sizeof 指令

compile-time operator

- 查詢變數佔了多少個位元組的語法：

sizeof 指令

```
sizeof 變數名稱或常數;
```

或

```
sizeof(變數名稱或常數);
```

- 查詢資料型態所佔的位元組：

sizeof 指令

```
sizeof(資料型態名稱);
```

sizeof 指令的應用

```
01  /* prog3_11, 列印出各種資料型態的長度 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char ch;              /* 宣告字元變數 ch */
07     float num;            /* 宣告浮點數變數 num */
08
09     printf("sizeof(2L)=%d\n",sizeof(2L));     /* 查詢常數 2L 所佔位元組 */
10
11     printf("sizeof(ch)=%d\n",sizeof(ch));   /* 查詢字元變數 ch 所佔位元組 */
12     printf("sizeof(num)=%d\n",sizeof(num));   /* 查詢變數 num 所佔位元組 */
13
14     printf("sizeof(int)=%d\n",sizeof(int));   /* 查詢 int 型態所佔位元組 */
15     printf("sizeof(long)=%d\n",sizeof(long));   /* 查詢 long 型態所佔位元組 */
16     printf("sizeof(short)=%d\n",sizeof(short)); /* 查詢 short 所佔位元組 */
17
18     system("pause");
19     return 0;
20  }
```

```
/* prog3_11 OUTPUT---

sizeof(2L)=4
sizeof(ch)=1
sizeof(num)=4
sizeof(int)=4
sizeof(long)=4
sizeof(short)=2
----------------------*/
```

資料型態的轉換 (1/3)

- 將資料型態轉換成另一種型態的語法：

資料型態的強制轉換

(欲轉換的資料型態) 變數名稱

```
int num=12;
float total;
total=(float) num;    /* 將int型態轉換成float型態 */
num = (int) 12.345;
```

資料型態的轉換 (2/3)

- 把浮點數轉換成整數的範例：

```
01  /* prog3_12, 資料型態的轉換*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int n1,n2;
07     float num1=3.002F,num2=3.988F;
08
09     n1=(int) num1;       /* 將浮點數 num1 轉換成整數 */
10     n2=(int) num2;       /* 將浮點數 num2 轉換成整數 */
11
12     printf("num1=%f, num2=%f\n",num1,num2);  /* 印出浮點數的值 */
13     printf("n1=%d, n2=%d\n",n1,n2);  /* 印出浮點數轉成整數後的值 */
14
15     system("pause");
16     return 0;
17  }
```

```
/* prog3_12 OUTPUT---------

num1=3.002000, num2=3.988000
n1=3, n2=3
-------------------------------*/
```

n1 = num1;
g++ compiler warning messages (not gcc)
test.cpp:9: warning: converting to `int` from `float'

資料型態的轉換 (3/3)

- 把整數轉換成浮點數，再進行除法運算：

```
01  /* prog3_13, 資料型態的轉換*/
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num=5;
07
08     printf("num/2=%d\n",num/2);              /* 整數相除 */
09     printf("(float)num/2=%f\n",(float)num/2); /* 將整數轉成浮點數，再做除法 */
10
11     system("pause");          /* prog3_13 OUTPUT--------
12     return 0;
13  }                            num/2=2
                                 (float)num/2=2.500000
                                 ----------------------------*/
```