

基本資料型態

變數與常數

1

■ 變數

- 利用宣告的方式，將記憶體中的某個區域保留下來以供程式使用。
- 不同類型的資料需要不同型態的變數來儲存。
- 宣告變數時，編譯程式會在記憶體內配置一塊足以容納此變數大小的記憶體空間，不管變數如何改變，同一個型態的變數永遠占用相同的記憶體空間。

■ 常數

- 值是固定的

宣告變數

```
#include <stdio.h>

int main(void)
{
    int num1 = 12400;    //宣告num1為整數變數
    double num2 = 5.234; //宣告num2為雙精倍浮點數
    printf("%d is an integer.\n", num1); // %d 十禁制
    printf("%f is an double.\n", num2);  // %f 浮點數
    printf("\n");
    system("pause");
    return 0;
}
```

示意圖

```
int num1=12400;
```

宣告整數變數 num1，
並設值為 12400

num1

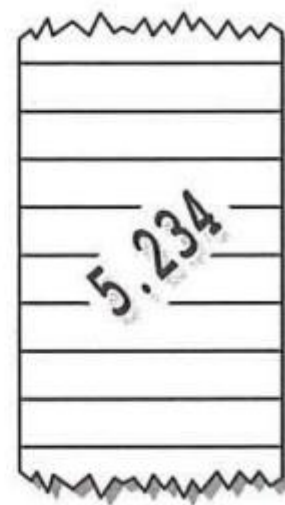


4 bytes

```
double num2=5.234;
```

宣告 double 型態的變數
num2，並設值為 5.234

num2



8 bytes

基本資料型態

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)
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

無號整數

- 資料不會負數時使用，關鍵字「unsigned」

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

溢位

6

- ➡ 當數值大小超過變數可以表示的範圍時發生溢位「overflow」
- ➡ 計數器的內容到最大值時，會自動歸0

溢位發生

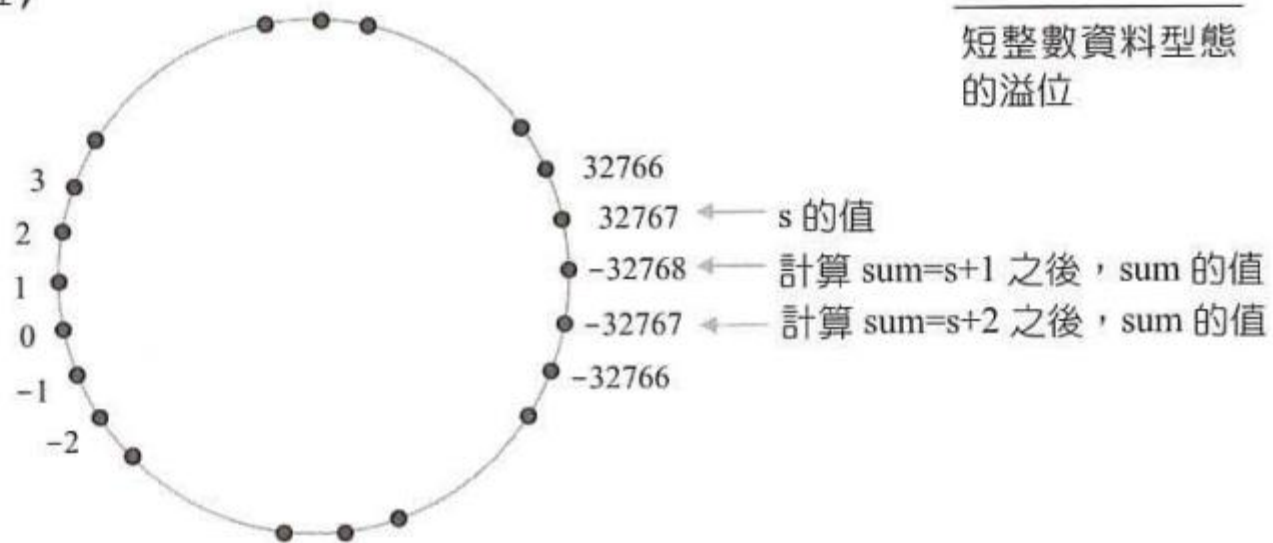
```
#include <stdio.h>

int main(void)
{
    short sum,s=32767;

    sum = s+1;
    printf("s+1=%d\n",sum);
    sum = s+2;
    printf("s+2=%d\n",sum);
    system("pause");
    return 0;
}
```


短整數資料型態的溢位

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



字元型態

- 變數宣告字元時用「char」表示，佔有1個位元組(byte)
- 宣告值時用單引號「'」包住
- 通常字元會被編碼，將一個字元編上一個整數碼，以變數處理字元
- ASCII(American Standard Code for Information Interchange)
 - 用來制訂電腦中每個符號對應的整數代碼，也叫電腦的內碼
 - 每個ASCII值以 1 個位元組儲存，數字0到127代表不同的常用符號
- 延伸ASCII碼(extended ASCII)
 - 由於每個ASCII碼占用了一個位元組，而一個位元組有8個位元，所以每個位元組可以表示 $2^8 = 256$ ，但較高位元的(128~255)未被使用，之夠將其位元編入ASCII中故稱「延伸ASCII碼」
 - 加上了許多數學與表格框線等特殊符號

ASCII碼表

Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00		NUL	32	20	!	64	40	@	96	60	'
1	01		SOH	33	21	!	65	41	A	97	61	a
2	02		STX	34	22	"	66	42	B	98	62	b
3	03		ETX	35	23	#	67	43	C	99	63	c
4	04		EOT	36	24	\$	68	44	D	100	64	d
5	05		ENQ	37	25	%	69	45	E	101	65	e
6	06		ACK	38	26	&	70	46	F	102	66	f
7	07		BEL	39	27	'	71	47	G	103	67	g
8	08		BS	40	28	(72	48	H	104	68	h
9	09		HT	41	29)	73	49	I	105	69	i
10	0A		LF	42	2A	*	74	4A	J	106	6A	j
11	0B		VT	43	2B	+	75	4B	K	107	6B	k
12	0C		FF	44	2C	,	76	4C	L	108	6C	l
13	0D		CR	45	2D	-	77	4D	M	109	6D	m
14	0E		SO	46	2E	.	78	4E	N	110	6E	n
15	0F		SI	47	2F	/	79	4F	O	111	6F	o
16	10		DLE	48	30	0	80	50	P	112	70	p
17	11		DC1	49	31	1	81	51	Q	113	71	q
18	12		DC2	50	32	2	82	52	R	114	72	r
19	13		DC3	51	33	3	83	53	S	115	73	s
20	14		DC4	52	34	4	84	54	T	116	74	t
21	15		NAK	53	35	5	85	55	U	117	75	u
22	16		SYN	54	36	6	86	56	V	118	76	v
23	17		ETB	55	37	7	87	57	W	119	77	w
24	18		CAN	56	38	8	88	58	X	120	78	x
25	19		EM	57	39	9	89	59	Y	121	79	y
26	1A		SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B		ESC	59	3B	;	91	5B	[123	7B	{
28	1C		FS	60	3C	<	92	5C	\	124	7C	
29	1D		GS	61	3D	=	93	5D]	125	7D	}
30	1E	▲	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	▼	US	63	3F	?	95	5F	_	127	7F	Δ*

擴充字元集(Extended Character Set)

128	Ç	144	É	160	á	176	☐	193	⌞	209	⌞	225	ß	241	±
129	ü	145	æ	161	í	177	☐	194	⌞	210	⌞	226	Γ	242	≥
130	é	146	Æ	162	ó	178	☐	195	⌞	211	⌞	227	π	243	≤
131	â	147	ô	163	ú	179		196	-	212	⌞	228	Σ	244	∫
132	ä	148	ö	164	ñ	180	⌞	197	+	213	⌞	229	σ	245	∫
133	à	149	ò	165	Ñ	181	⌞	198	⌞	214	⌞	230	μ	246	÷
134	â	150	û	166	²	182	⌞	199	⌞	215	⌞	231	τ	247	≈
135	ç	151	ù	167	°	183	⌞	200	⌞	216	⌞	232	Φ	248	°
136	ê	152	-	168	¿	184	⌞	201	⌞	217	⌞	233	Θ	249	.
137	ë	153	Ö	169	-	185	⌞	202	⌞	218	⌞	234	Ω	250	.
138	è	154	Û	170	¬	186	⌞	203	⌞	219	■	235	δ	251	√
139	ï	156	£	171	½	187	⌞	204	⌞	220	■	236	∞	252	-
140	î	157	¥	172	¾	188	⌞	205	=	221	■	237	φ	253	²
141	ì	158	-	173	¡	189	⌞	206	⌞	222	■	238	ε	254	■
142	Ä	159	ƒ	174	«	190	⌞	207	⌞	223	■	239	∧	255	
143	Å	192	Ł	175	»	191	⌞	208	⌞	224	α	240	≡		

字元轉換ASCII

```
#include <stdio.h>

int main(void)
{
    char ch='a';
    printf("ch = %c\n",ch);
    printf("ASCII of ch = %d\n",ch);
    printf("-----\n");
    int ch_int = 97;
    printf("ch = %d\n",ch_int);
    printf("ASCII of ch = %c\n",ch_int);
    system("pause");
    return 0;
}
```

練習

```
#include <stdio.h>

int main(void)
{
    char ch='2';
    printf("ch = %c\n",ch);
    printf("ASCII of ch = %d\n",ch);
    system("pause");
    return 0;
}
```

練習

```
#include <stdio.h>

int main(void)
{
    char ch=298;
    printf("ch = %c\n",ch);
    printf("ASCII of ch = %d\n",ch);
    system("pause");
    return 0;
}
```

輸出溢位

- 當%c遇到超過255數值時，只會擷取後面8個bits(一個位元組)
- 298的二進制為 100101010，被%c擷取8個位元後為 00101010 剛好為十進制的42，而ASCII碼的42是「*」
- 這種擷取後面一個位元的方式相當於取餘數的概念

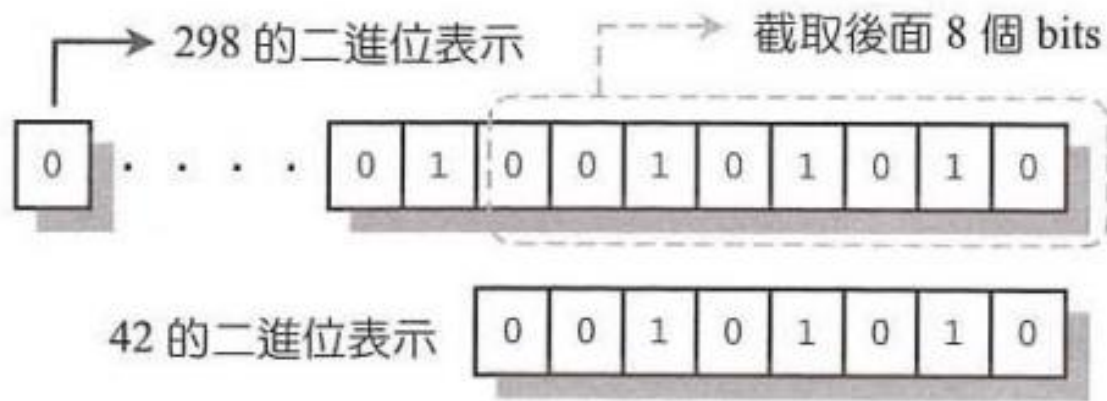


圖 3.2.2

字元型態的變數，若設值超過 255，則只會截取後面 8 個 bits

跳脫字元與跳脫序列

- 對於某些無法用鍵盤輸入的字元，以反斜線「\」加上一個控制碼作為一個完整的特殊字元，以便和正常的字元作區別
- 由於反斜線字元「\」後緊接一個字元時，這個字元會被解譯成控制碼，已經跳脫原本的涵意，因此反斜線「\」稱為跳脫字元(escape character)
- 反斜線後面加上的控制碼為跳脫序列(escape sequence)

常用的跳脫序列與ASCII碼

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

練習

```
#include <stdio.h>

int main(void)
{
    char ch='\a';
    printf("ch = %c\n",ch);
    printf("ASCII of ch = %d\n",ch);
    system("pause");
    return 0;
}
```

浮點數型態

➤ 浮點數(floating point)

➤ 數學中帶有小數點的實數

➤ 4個位元組

➤ 有效範圍 $1.2 \times 10^{-38} \sim 3.4 \times 10^{38}$

➤ 數值後面可加上F或f表示浮點數常數，如不加上編譯器會視為double

➤ 數值後也可加上E或e表示指數型態表示

➤ EX

➤ 245.32 → 2.4532E2 2.4532×10^2

➤ 0.07652 → 7.65E-2 7.652×10^{-2}

➤ printf

➤ %f

➤ %e 以指數方式輸出

浮點數列印

```
#include <stdio.h>

int main(void)
{
    float num1 = 123.456F;
    float num2 = 456E-3F;
    printf("num1 = %e\n", num1);
    printf("num2 = %f\n", num2);
    system("pause");
    return 0;
}
```

雙精倍浮點數

➡ double

➡ 8個位元組

➡ 有效範圍 $2.2 \times 10^{-308} \sim 1.8 \times 10^{308}$

➡ printf輸出時也是使用%f

float與double比較

```
#include <stdio.h>

int main(void)
{
    float  num1 = 123.456789012345F;
    double num2 = 123.456789012345;
    //以16個字元輸出，則小數點後12個位數
    printf("num1 = %16.12f\n", num1);
    printf("num2 = %16.12f\n", num2);
    system("pause");
    return 0;
}
```

程式解說

- ➡ float只能容納8個位數的精度
- ➡ double可容納16個位數的精度

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



圖 3.2.3

float型態只有7~8個數字的精度

float 型態的變數只有 7~8 個數字的精度

此部份的數字已超出 float 的精度範圍，是屬於記憶體內的殘值

- ➡ printf
 - ➡ %16.12f (16個字元輸出，則小數點後12個位數)

查詢常數、變數或資料型態所佔的位元組

➡ sizeof (變數或常數名稱)

```
#include <stdio.h>

int main(void)
{
    char ch;
    float num;
    printf("sizeof(2L) = %d\n", sizeof(2L));
    printf("sizeof(ch) = %d\n", sizeof(ch));
    printf("sizeof(num) = %d\n", sizeof(num));
    printf("sizeof(int) = %d\n", sizeof(int));
    printf("sizeof(long) = %d\n", sizeof(long));
    printf("sizeof(short) = %d\n", sizeof(short));
    system("pause");
    return 0;
}
```


資料型態的轉換

- ➡ (欲轉換的資料型態) 變數名稱
 - ➡ 浮點數強制轉換成整數時，編譯器不會做四捨五入的動作，而是直接將小數點捨棄，只留下整數的部分
 - ➡ 整數相除時只能得到商，並捨棄所有小數點

浮點數強制轉換成整數

```
#include <stdio.h>

int main(void)
{
    int n1,n2;
    float num1=3.002f,num2=3.988F;
    n1 = (int) num1;
    n2 = (int) num2;

    printf("num1=%f, num2=%f\n",num1,num2);
    printf("n1=%d, n2=%d\n",n1,n2);
    system("pause");
    return 0;
}
```

整數相除

```
#include <stdio.h>


int main(void)
{
    int num=5;

    printf("num/2=%d\n",num/2);
    printf("(float)num/2 = %f\n", (float)num/2);
    system("pause");
    return 0;
}
```

習題

27


➡ 請輸出以下結果：

 Z:\C2講義\C語言\C語言教學手冊\課程範例\03_習題1_雙引號輸出.exe

```
Hello World "C"  
請按任意鍵繼續 . . . _
```

習題

- 宣告兩個變數並宣告如下 `ch='1'` 與 `num=1`，使其相加結果為2

 Z:\C2講義\C語言\C語言教學手冊\課程範例\03_習題2_變數轉換運算.exe

```
ch + num = 2  
請按任意鍵繼續 . . .
```