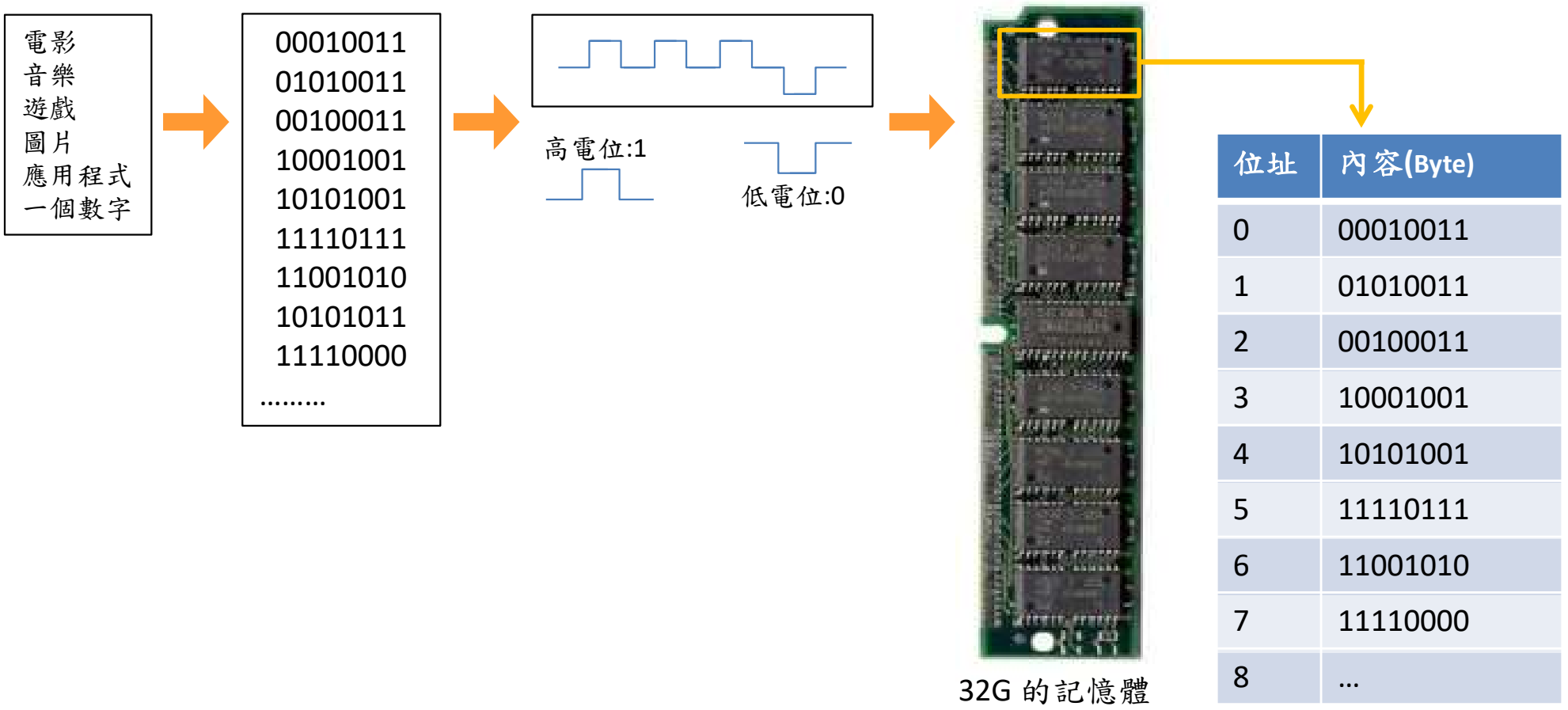
第二章變數 大綱

- 產生變數
  - 資料型態與值
- 變數與記憶體
  - 整數
  - 字串
- 補充:
  - 位元(bit)與位元組(Byte)
  - 進制轉換
    - 10進制與2進制的轉換
    - 小數點轉換
    - 負數表示

第二章變數　位元(bit)與位元組(Byte)

- 電腦存的是0或1
- 所有資料用0或1的排列組合表示
- 每一個0或1稱呼為bit(位元); 也就是說bit(位元)有0或1兩種狀態
- 8 bits (位元) 為 1 Byte(位元組)



位址	內容(Byte)
0	00010011
1	01010011
2	00100011
3	10001001
4	10101001
5	11110111
6	11001010
7	11110000
8	…

參考
K = 2^{10} = 1024
M = 2^{20} = 1024x1024 = 1048576(大約1百萬)
G = 2^{30} = 1024x1024x1024 = 1073741824(大約10億)

要怎麼在程式中拿幾個Byte來存東西?

第二章變數　宣告變數(產生變數)

變數的作用: 程式中最基本的存取單位, 參與程式的運算、判斷、儲存

宣告變數的語法:

資料型態　變數名稱=值;

我是空格　英文字母

資料型態	名稱意義	值(最小值)	值(最大值)	記憶體(Bytes)	注意事項
byte	整數(位元組)	-128	127	1	
short	整數(範圍短)	-32768	32767	2	
int	整數(一般常用)	-2147483648	2147483647	4	21億
long	整數(範圍長)	-9223372036854775808	9223372036854775807	8	值要加l
float	浮點數(有小數點)	$-3.4028235x10^{38}$	$3.4028235x10^{38}$	4	值要加f
double	浮點數(有小數點)	$-1.7976931348623157x10^{308}$	$1.7976931348623157x10^{308}$	8	預設浮點數
char	字元(一個字)	Unicode 編碼		2	值要放在''中
String	字串(多個字元, 句子)	字串物件		字串長度	值要放在""中
boolean	布林值(真, 假)	true, false		1bit	依據JVM定義

注意

1. 使用float看的到的數字最多有8位
2. 使用double看的到的數字最多有17位
3. 數字 $3.4028235x10^{38}$ 可以表示為
   數字 3.4028235E+38
4. 自Java7開始數值太長可用底線_做作區隔

不考慮整數部分，只考慮最多可以寫幾個小數		
	負數的最大值	正數的最小值
float	$-1.4023984x10^{-45}$	$1.4023984x10^{-45}$
double	$-4.940656458412465x10^{-324}$	$4.940656458412465x10^{-324}$

第二章變數　宣告變數(命名規則)

變數名稱命名要點:英文單字

變數名稱命名規則:

1. 盡量用有意義的英文命名
2. 命名區分大小寫， e.g. Apple 與 apple 是兩個不同命名
3. 最開頭請用英文小寫， e.g. hello
4. 兩個英文以上,串接後的英文開頭盡量大寫， e.g. helloWorld
5. 命名可以穿插包含數字，但不可以將數字放在名稱開頭
6. 命名可以穿插包含底線_，可以將底線放在名稱開頭
7. 命名不可以包含空格或者某些特殊字元(+-*'"\/等)
8. 不可以用關鍵字(被系統用掉的字)來命名

```
/*程式碼範例*/
int apple = 13;
float usd = 29.37f;
char c = 'x';
String message = "你好嗎 :)";
boolean isGood = true;
```

第二章變數　宣告常數(final)

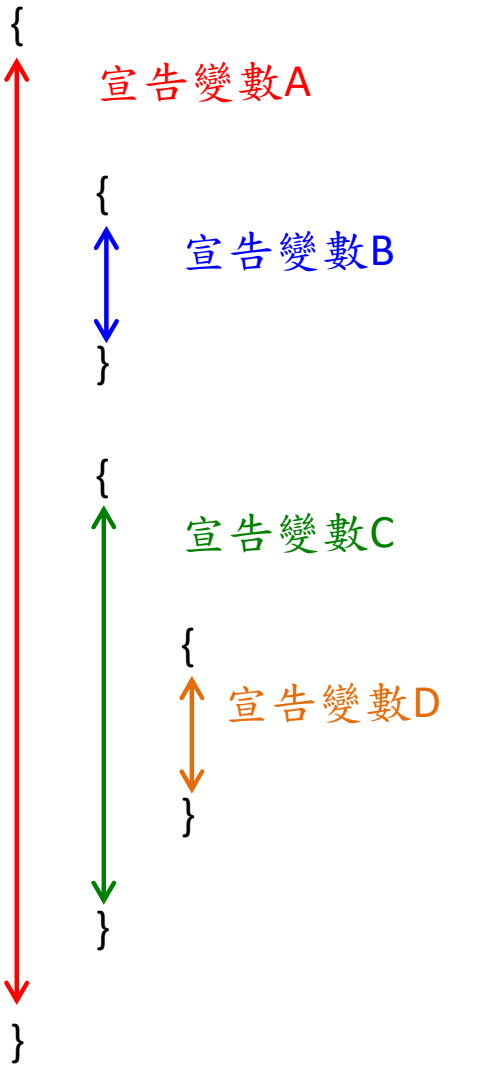
1. 如果我們希望變數的值無法被改變，那該怎麼辦? 使用 final 修飾變數
2. 使用 final 修飾變數後，該變數無法再設定其他值
3. 變數名稱習慣上全大寫，多個單字可用底線區隔

final 資料型態 變數名稱=值;

```
/*程式碼範例*/
final int ID = 168;
final float PI = 3.14f;
final char WORD_A = 'A';
final String TAIWAN = "台灣";
final boolean GOOD = true;
```

第二章變數　變數生命範圍(可用範圍)

- 變數的可用範圍以包含宣告他的大括號{}為主




```
{
    int vA = 0;
    {
        int vB=1;
    }
    {
        int vC=2;
        {
            int vD=3;
        }
    }
}
```

- 於主控台輸出字串指令
  - System.out.println()
    - 輸出字串後換行
  - System.out.print()
    - 輸出字串後不換行

程式範例	輸出
`System.out.println("列印完會換行");`	列印完會換行 |◄---------- 輸出完以後游標會跑到下一行
`System.out.print("列印完不會換行");`	列印完不會換行|◄--------- 輸出完以後游標不會跑到下一行,等下如果有其他字串輸出，會直接出現在同一行

逃逸字元(跳脫字元)	功能
\'	於字串中顯示單引號
\”	於字串中顯示雙引號
\\	於字串中顯示反斜槓
\t	於字串中產生水平空格(4格)
\n	於字串中產生換行

- 於主控台輸出字串指令
  - System.**out**.printf
    - 可格式化輸出字串
    - 於字串中使用轉換字元來表示變數
    - 藉此修改變數值的輸出格式
    - 可調整變數在字串中的位置

```
char    cV = 'S';
String  sV = "HAHA";
int     iV = 21;
float   fV = 3.14f;
double  dV = 3.14159265358979323846;
boolean bV = false;
```

轉換字元	功能	程式範例	輸出
%c	轉換成字元	System.*out*.printf("替換:%c\n", cV);	替換:S
%s	轉換成字串	System.*out*.printf("替換:%s\n", sV);	替換:HAHA
%d	轉換成10進制整數	System.*out*.printf("替換:%d\n", iV);	替換:21
%f	轉換成10進制浮點數	System.*out*.printf("替換:%f\n", fV); System.*out*.printf("替換:%f\n", dV);	替換:3.140000 替換:3.141593
%x	轉換成16進制整數	System.*out*.printf("替換:%x\n", iV);	替換:15
%a	轉換成16進制浮點數	System.*out*.printf("替換:%a\n", fV); System.*out*.printf("替換:%a\n", dV);	替換:0x1.91eb86p1 替換:0x1.921fb54442d18p1
%b	轉換成布林值	System.*out*.printf("替換:%b\n", bV);	替換:false
%e	轉換成科學記號	System.*out*.printf("替換:%e\n", fV); System.*out*.printf("替換:%e\n", dV);	替換:3.140000e+00 替換:3.141593e+00

第二章變數　從主控台輸出變數(printf)

- 轉換字元與格式字元公式：%[格式字元][字串總長度位數][.小數位數]轉換字元

- 中括號[]：　表示可選，不一定要有
- 字串總長度：　表示字串顯示時，要顯示的長度位數
  - 如果欲顯示的字串超過字串總長度並不會發生截斷
  - 如果欲顯示的字串不足字串總長度會用空格補齊
- 小數位數：　表示小數顯示時，要顯示的小數位數
  - 如果欲顯示的小數超過小數位數會發生截斷
  - 如果欲顯示的小數不足小數位數的長度會有亂數問題產生
  - 小數點也會佔字串總長度一位
  - %f預設小數位數顯示6位
- 格式字元：　不足位數時，用格式字元取代空格

格式字元	功能	備註
+	強調正號顯示	無順序,只要放在字串總長度前即可
,	整數部分每3個位數補一個逗號	無順序,只要放在字串總長度前即可
0	不足位數補0	無順序,只要放在字串總長度前即可
-	顯示改成靠左對齊	A. 不可跟0一起用 B. 無順序,只要放在字串總長度前即可

第二章變數　從主控台輸出變數(printf)

程式範例　int iV = 1234567;	輸出	備註
System.out.printf("01.　格式化:%d\n", iV);	01. 格式化:1234567	
System.out.printf("02.　格式化:%9d\n", iV);	02. 格式化: 1234567	多兩個空格
System.out.printf("03.　格式化:%09d\n", iV);	03. 格式化:001234567	2個空格換成2個0
System.out.printf("04.　格式化:%+9d\n", iV);	04. 格式化: +1234567	1個空格與正號
System.out.printf("05.　格式化:%0+9d\n", iV);	05. 格式化:+01234567	格式字元無順序性
System.out.printf("06.　格式化:%+09d\n", iV);	06. 格式化:+01234567	格式字元無順序性
System.out.printf("04-1.格式化:%+9d\n", -iV);	04-1.格式化: -1234567	**負號本身會佔一位**
System.out.printf("05-2.格式化:%0+9d\n", -iV);	05-2.格式化:-01234567	格式字元無順序性
System.out.printf("06-3.格式化:%+09d\n", -iV);	06-3.格式化:-01234567	格式字元無順序性
System.out.printf("7.　格式化:%+0,9d\n", -iV);	7.　格式化:-1,234,567	每3個整數位數補一個逗號
System.out.printf("8.　格式化:%-+9d\n", iV);	8.　格式化:+1234567	7後面多兩個空格

轉換字元	功能
%d	轉換成10進制整數
%f	轉換成10進制浮點數

%[格式字元][字串總長度位數][.小數位數]轉換字元

- 中括號[]: 表示可選，不一定要有
- 字串總長度: 表示字串顯示時，要顯示的長度位數
  - 如果欲顯示的字串超過字串總長度並不會發生截斷
  - 如果欲顯示的字串不足字串總長度會用空格補齊
- 小數位數: 表示小數顯示時，要顯示的小數位數
  - 如果欲顯示的小數超過小數位數會發生截斷
  - 如果欲顯示的小數不足小數位數的長度會有亂數問題產生
  - 小數點也會佔字串總長度一位
  - %f預設小數位數顯示6位
- 格式字元: 不足位數時，用格式字元取代空格

第二章變數　從主控台輸出變數(printf)

程式範例　float fV = 3000.14f;	輸出	備註
System.out.print("00.　未格式化:" + fV + "\n");	00. 未格式化:3000.14	
System.out.printf("01.　格式化:%f\n",　　fV);	01. 格式化:3000.139893	不指定小數位數 預設顯示小數6位
System.out.printf("02.　格式化:%3f\n",　　fV);	02. 格式化:3000.139893	超過字串長度3 (字串長度包含小數點)
System.out.printf("03.　格式化:%3.6f\n",　　fV);	03. 格式化:3000.139893	
System.out.printf("04.　格式化:%3.3f\n",　　fV);	04. 格式化:3000.140	只顯示3位小數位數
System.out.printf("05.　格式化:%3.9f\n",　　fV);	05. 格式化:3000.139892578	出現亂碼
System.out.printf("06.　格式化:%9.3f\n",　　fV);	06. 格式化: 3000.140	最前面多一個空格 (字串長度包含小數點)
System.out.printf("07.　格式化:%12f\n",　　fV);	07. 格式化: 3000.139893	最前面多一個空格 (小數點預設長度)

轉換字元	功能
%d	轉換成10進制整數
%f	轉換成10進制浮點數

%[格式字元][字串總長度位數][.小數位數]轉換字元

- 中括號[]: 表示可選，不一定要有
- 字串總長度: 表示字串顯示時，要顯示的長度位數
  - 如果欲顯示的字串超過字串總長度並不會發生截斷
  - 如果欲顯示的字串不足字串總長度會用空格補齊
- 小數位數: 表示小數顯示時，要顯示的小數位數
  - 如果欲顯示的小數超過小數位數會發生截斷
  - 如果欲顯示的小數不足小數位數的長度會有亂數問題產生
  - 小數點也會佔字串總長度一位
  - %f預設小數位數顯示6位
- 格式字元: 不足位數時，用格式字元取代空格

Q1:一個int花費四個記憶體(Byte)是指?

A1:888168 = 00000000,00001101,10001101,01101000

會跟系統要4個Byte來儲存

宣告一個整數變數並且值等於23

```
int k = 23;
```

23用4個Byte表示

2進制	0b00000000	0b00000000	0b00000000	0b00010111

存到記憶體

	記憶體位址	記憶體內容	
K →	0x0d00_0000	0b00010111	1byte
	0x0d00_0001	0b00000000	1byte
	0x0d00_0002	0b00000000	1byte
	0x0d00_0003	0b00000000	1byte
	0x0d00_0004		1byte
	0x0d00_0005		1byte
	0x0d00_0006		1byte

參考
0x 開頭表示16進制
0b 開頭表示2進制
0 開頭表示8進制

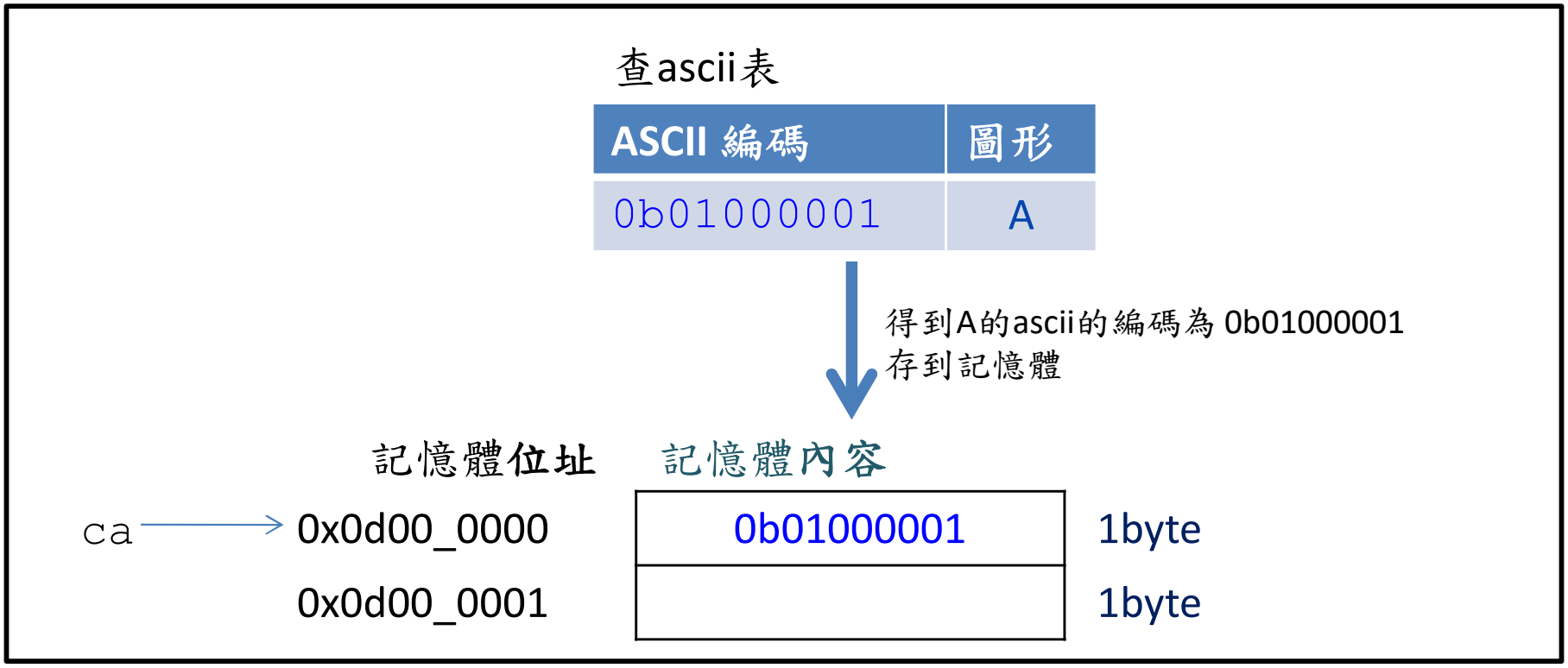
ASCII碼(二進位)	圖形
0100 0000	@
0100 0001	A
0100 0010	B
0100 0011	C
0100 0100	D
0100 0101	E
0100 0110	F
0100 0111	G
0100 1000	H
0100 1001	I
0100 1010	J
0100 1011	K
0100 1100	L
0100 1101	M
0100 1110	N
0100 1111	O
0101 0000	P
0101 0001	Q
0101 0010	R
0101 0011	S
0101 0100	T
0101 0101	U
0101 0110	V
0101 0111	W
0101 1000	X
0101 1001	Y
0101 1010	Z

```
char ca = 0b0100_0001;
char ca2= 'A';
System.out.println("ca:"+ca + ", ca2:"+ca2);
// a:65, ca:A, ca2:A
```

參考
https://zh.wikipedia.org/wiki/ASCII
https://zh.wikipedia.org/zh-tw/Unicode%E5%AD%97%E7%AC%A6%E5%88%97%E8%A1%A8

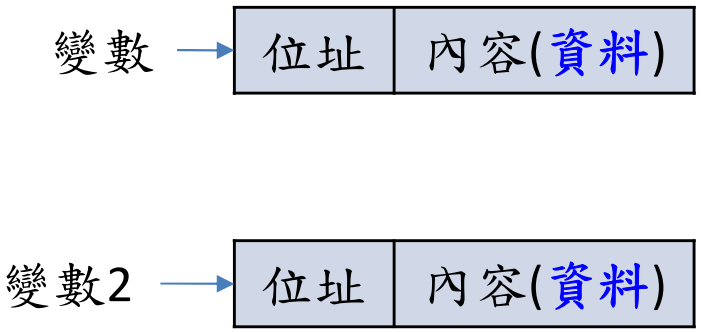
宣告一個字元變數並且值等於 A

```
char ca = 'A';
```



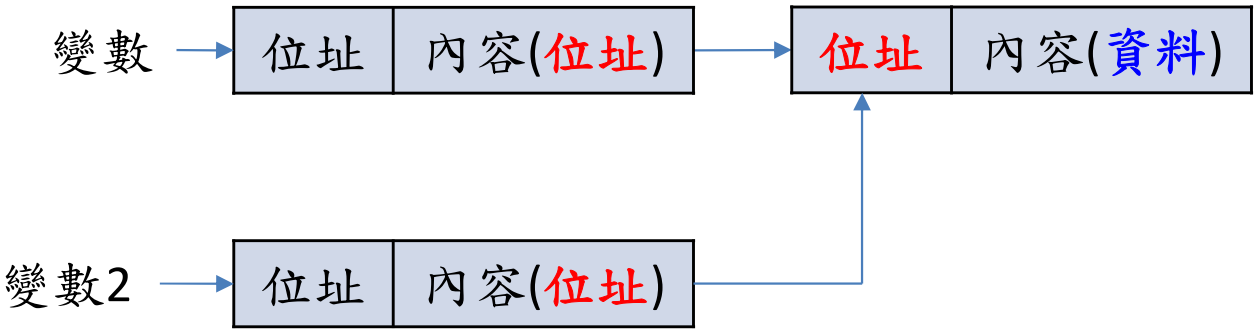
參考
0x 開頭表示16進制
0b 開頭表示2進制

基礎資料型別




```
int k =3;
int k2=k;
```

參考資料型別




```
String s  = "Hello";
String s2 = s;
```

基礎資料型態
Byte
Short
int
long
float
double
char
boolean

參考資料型態
String
屬於物件需要new

- Java有三種記憶體
  1. Stack: 存放基礎資料型態的內容
     - 基礎資料型態的變數佔用空間小
     - 存取**速度快**
  2. Heap: 存放參考資料型態的內容
     - 以間接方式存取(但我們使用時感覺不出來)
     - 利於**資料重複**使用
     - 可執行Garbage collection(垃圾收集)，管理未被參考的記憶體
  3. Global: 存放全域資料類型的內容
     - 使用關鍵字static修飾的變數

第一天複習

1. 宣告變數 `final 資料型態 變數名稱=值;`

2. 資料型別有
   - 數值類 (byte, short, int, double)
   - 文字類 (char, String)
   - 真假 (boolean)

3. 變數生命範圍看{}

4. 變數輸出至控制台(Run 視窗)的常用方法
   - `System.out.println("列印完會換行");`
   - `System.out.print("列印完不會換行");`
   - `System.out.printf("替換:%d\n", iV);`

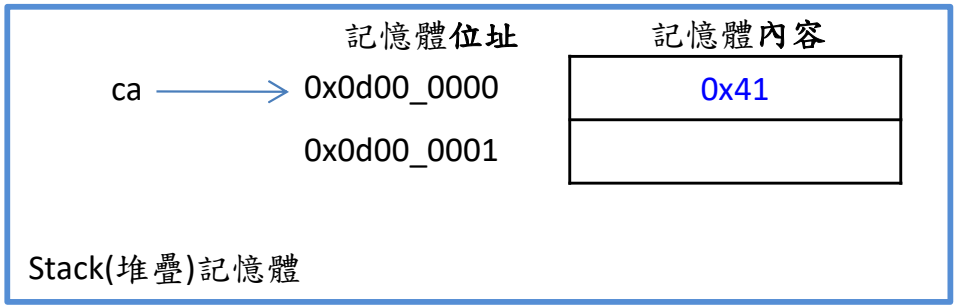
5. 文字格式化輸出 `%[格式字元][字串總長度位數][.小數位數]轉換字元`

6. 記憶體存取方式
   - 基礎資料型別: 直接定址 (位置對應內容)
   - 參考資料型別: 間接定址 (位置對應實際位置，實際位置對應內容)

7. 參考記憶體種類
   - Stack: 存基礎資料型別
   - Heap: 存參考資料型別
   - Global: 存static修飾

第二章變數　字串怎麼放在記憶體的

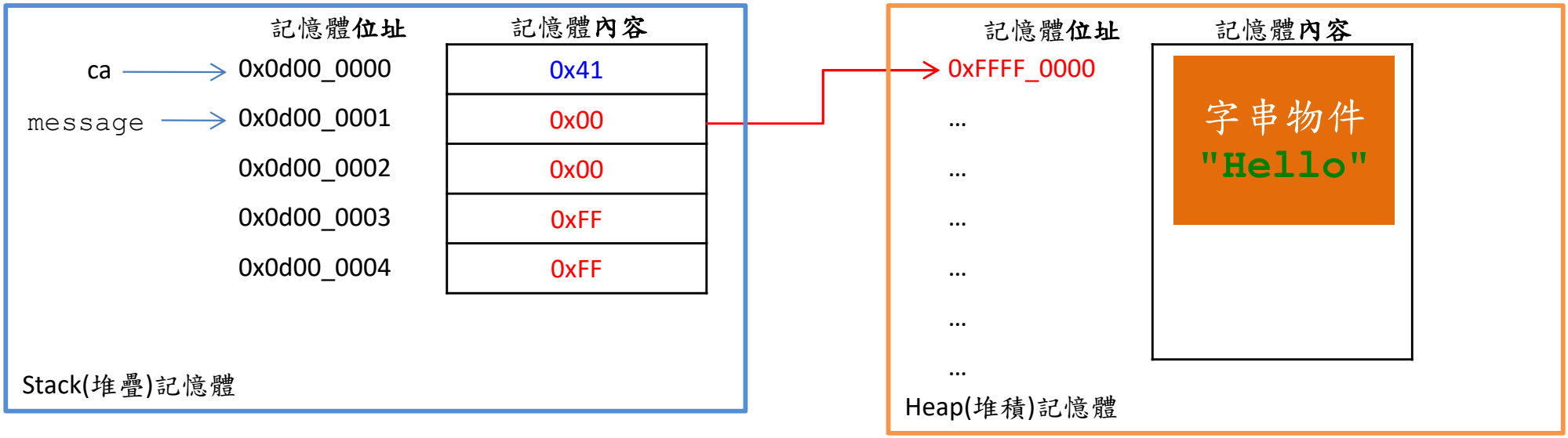
```
char ca = 'A';
```



基礎資料型態
byte
short
int
long
float
double
char
boolean

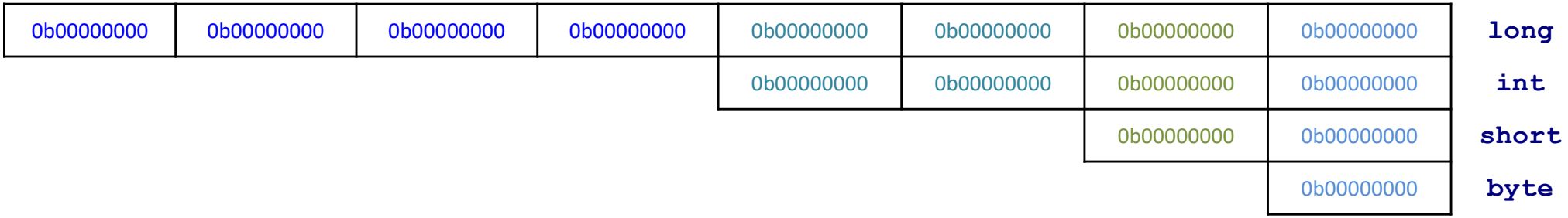
參考資料型態
String
屬於物件需要new

```
String message = "Hello";
```

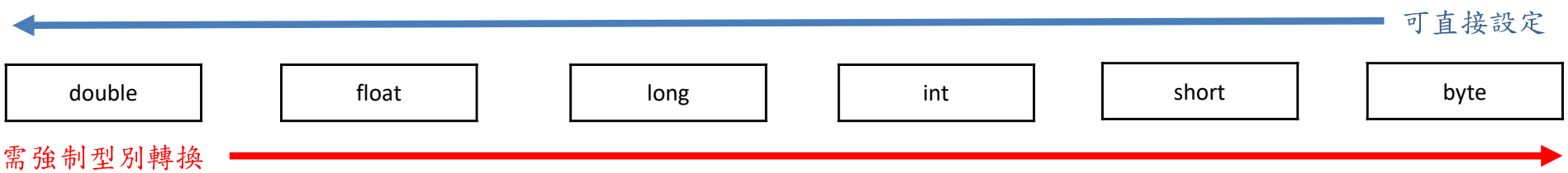


強制型別轉換

記憶體的分配



不同型態的變數間設定值




```
byte    bV = 127;
short   sV = 32767;
int     iV = 2147483647;
long    lV = 92233720368547758071;
float   fV = 3.14f;
double  dV = 3.1415926535897932384626;
```

```
iV = bV;
iV = sV;
iV = (int) lV;
iV = (int) fV;
iV = (int) dV;
```

```
lV = bV;
lV = sV;
lV = iV;
lV = (int) fV;
lV = (long) dV;
```

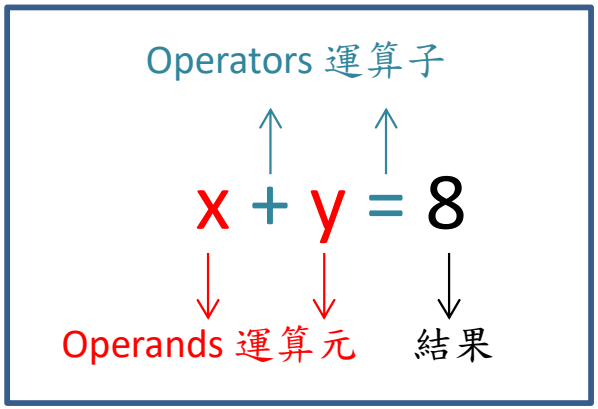
```
fV = bV;
fV = sV;
fV = iV;
fV = lV;
fV = (float) dV;
```

```
dV = bV;
dV = sV;
dV = iV;
dV = lV;
dV = fV;
```

第二章變數　運算子(指派、算數、一元)

運算子:計算用的符號, 邏輯運算用的符號...
運算元:變數

```
int x = 5;
int y = 3;
int z = x + y;
boolean a = true;
```



Expression 運算式

指派運算子	解釋功用	範例	備註
=	設定值	x = 5	等號左手邊放變數

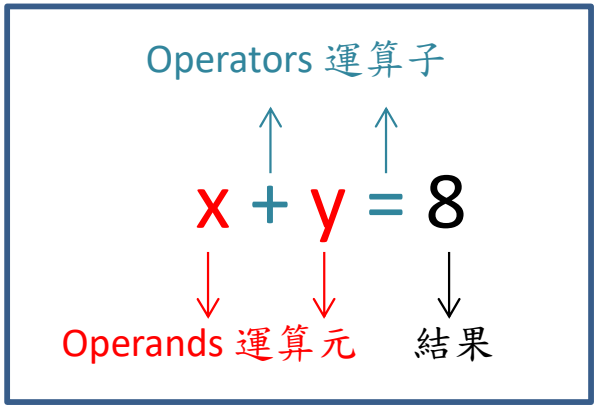
算術運算子	解釋功用	範例	備註
+	加	z = x + y	
-	減	z = x – y	
*	乘	z = x * y	
/	除	z = x / y	5/3 得 1
%	求餘數	z = x % y	5%3 得 2

一元運算子	解釋功用	範例	備註
+	正	+x	
-	負	-x	x*-1
++	遞增	++x, x++	x=x+1
--	遞減	--x, x--	x=x-1
!	布林值反相	!a	真變假, 假變真

注意
1. 整數除法將無條件捨去

第二章變數　運算子(指派、關係、條件，三元)

```
int x = 3;
int y = 5;
```



Expression 運算式

指派運算子	解釋功用	範例	備註
=	設定值	x = 3	等號左手邊放變數
+=	先相加再設定值	x += 3	x=x+3
-= *= /= %=	先相(減, 乘, 除, 求餘數)再設定值		

關係運算子	解釋功用	範例	備註
==	相等	x == y	
!=	不相等	x != y	
>	大於	x > y	
>=	大於等於	X >= y	x>y-1
<	小於	x < y	
<=	小於等於	x <= y	

條件運算子	解釋功用	範例	備註
&&	邏輯 And (且) (兩邊都要成立)	a && b	true && true
||	邏輯 Or (或) (一邊成立就好)	a || b	true || true

三元條件運算子 Elvis	解釋功用	範例	備註
?:	條件選擇	a ? b : c	真假？值1 : 值2

a	b	a&&b
F	F	F
F	T	F
T	F	F
T	T	T

a	b	a||b
F	F	F
F	T	T
T	F	T
T	T	T

- 變數輸出到控制台:

```
System.out.println("列印完會換行");
System.out.print("列印完不會換行");
System.out.printf("替換:%d\n", iV);
```

- 控制台輸入給變數

```
import java.util.Scanner;
```

```
Scanner scanner = new Scanner(System.in);
```

```
boolean boolInput   = scanner.nextBoolean();
short shortInput    = scanner.nextShort();
int intInput        = scanner.nextInt();
long longInput      = scanner.nextLong();
float floatInput    = scanner.nextFloat();
double doubleInput  = scanner.nextDouble();

String stringSpaceInput = scanner.next();
String stringLineInput  = scanner.nextLine();
char charInput          = scanner.next().charAt(0);
```

scanner.next基礎資料型別()

1. 以空格當作區隔
2. 以換行當作結束輸入
3. 若一開始讀取的不是資料而是空格或換行會嘗試掠過
4. 需小心執行完會殘留空格或換行在輸入緩衝區(主控台)

```
Scanner scanner = new
Scanner(System.in);
System.out.print("請輸入(int):");



int intInput        = scanner.nextInt();



System.out.println("intInput:" +
intInput);
```

Run

```
請輸入(int):
```

Run

```
請輸入(int):123
```

輸入 123

Run

```
請輸入(int):123
inInput:123
```

輸入 enter

Run

```
請輸入(int):
```

Run

```
請輸入(int):123456
```

輸入 123 456

Run

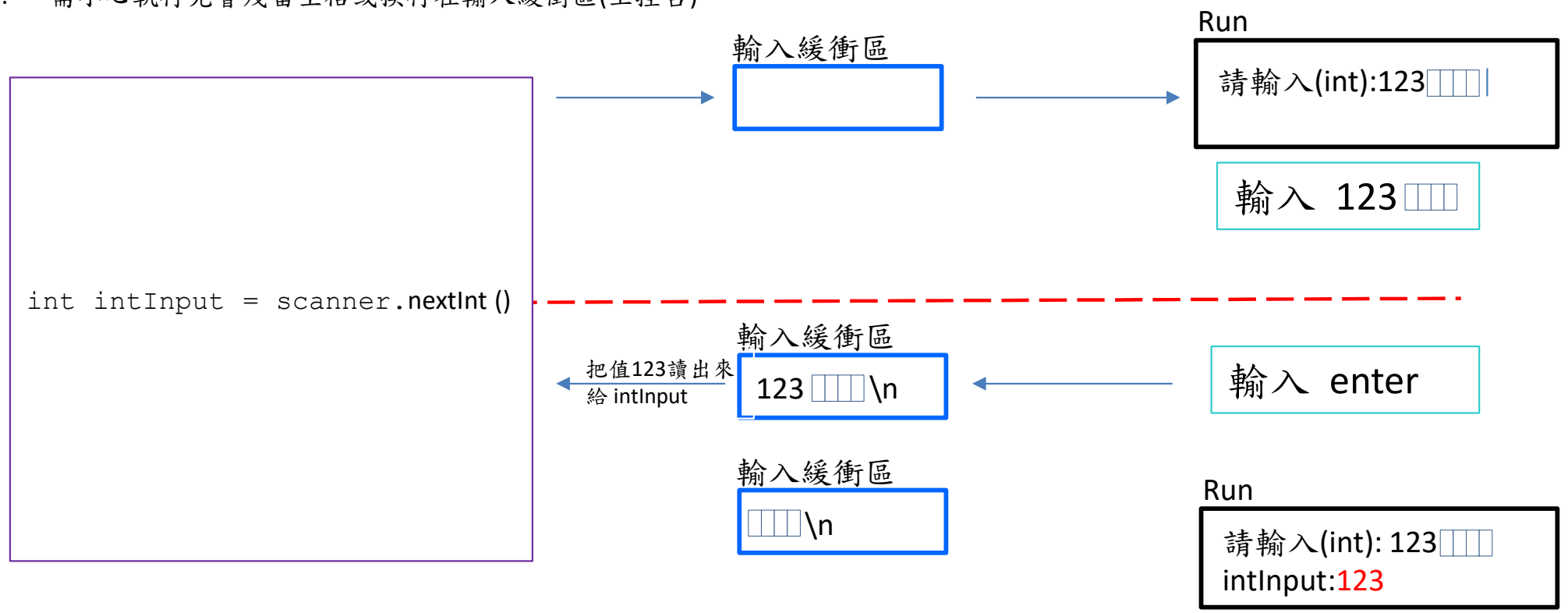
```
請輸入(int):123 456
inInput:123
```

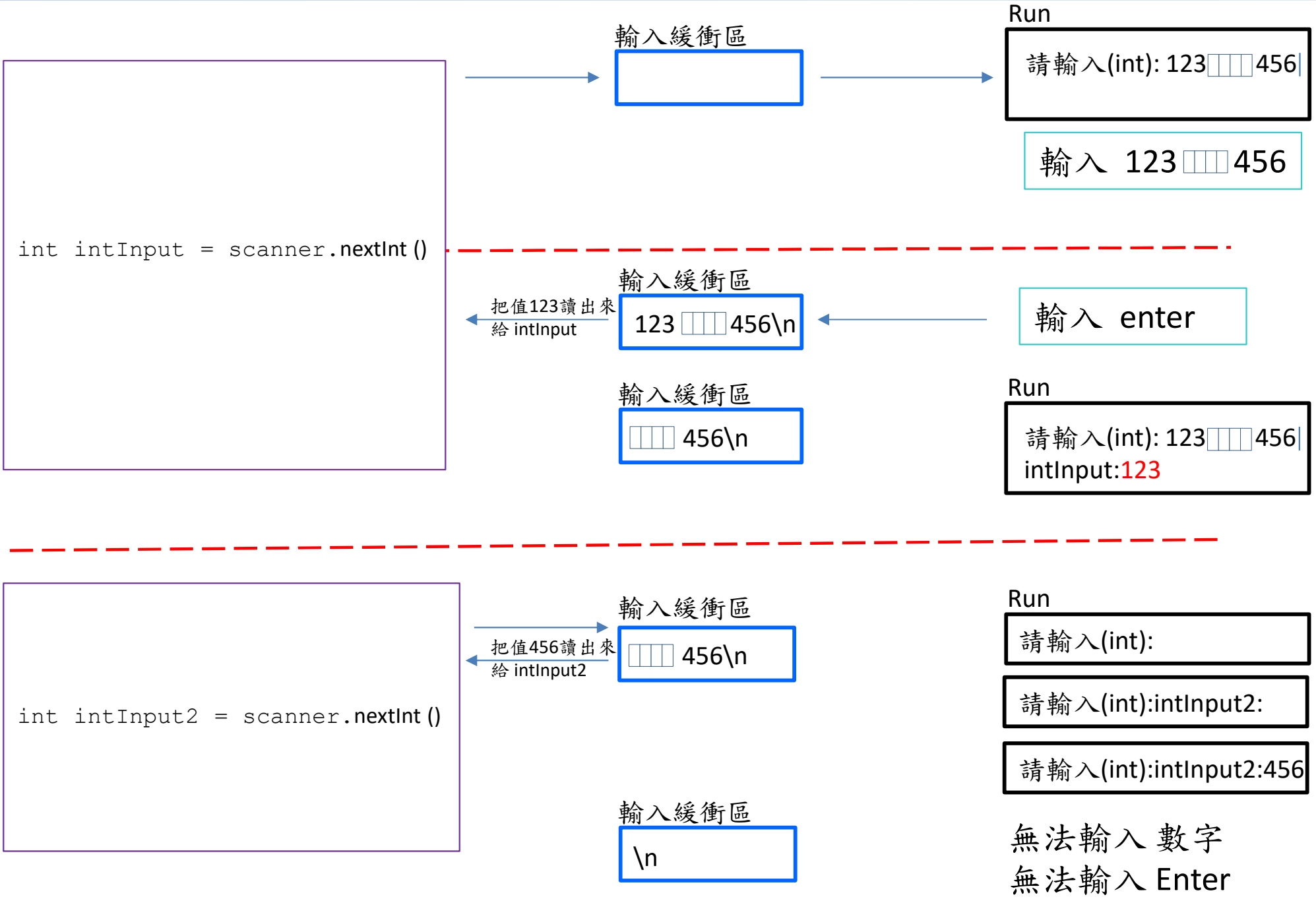
輸入 enter

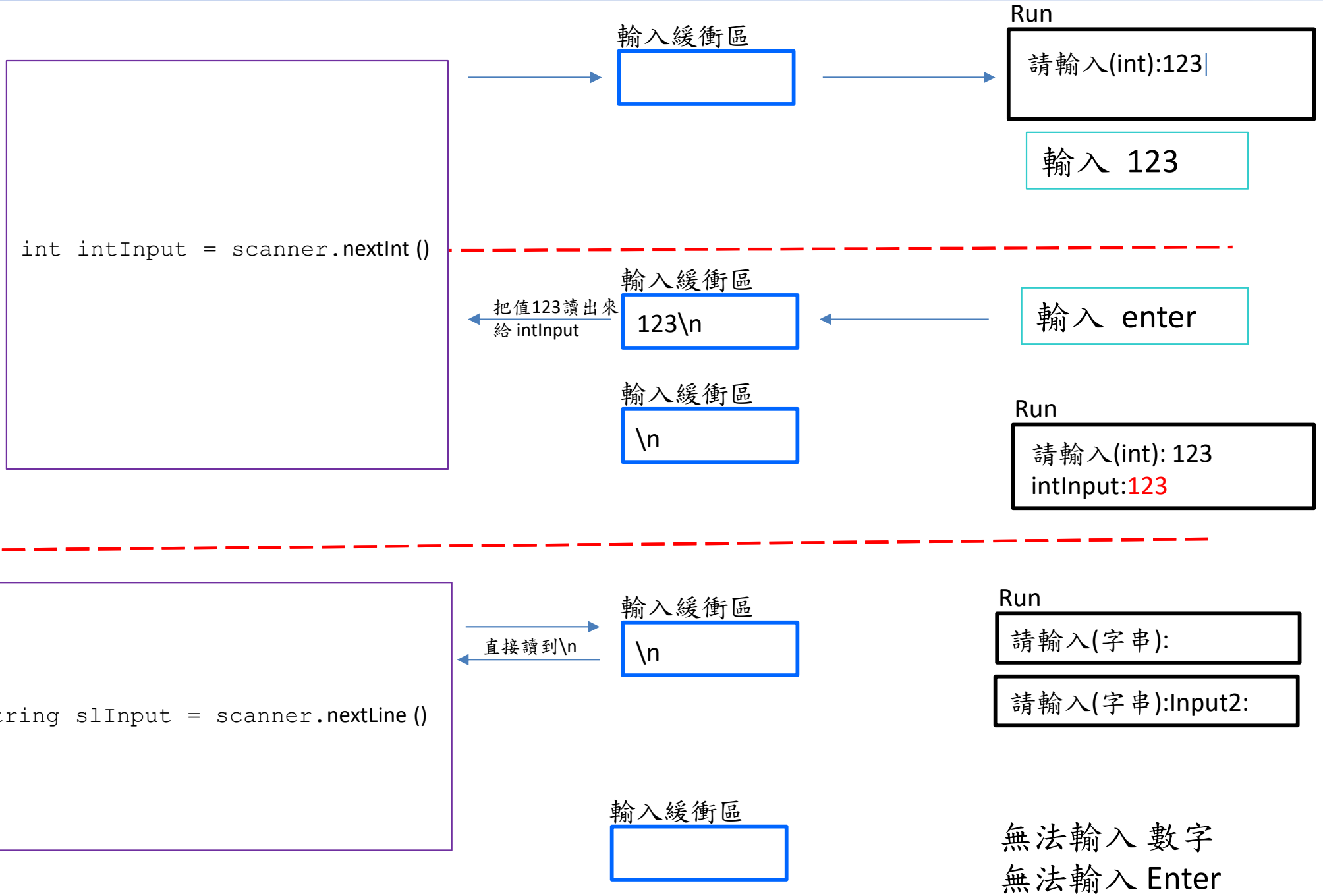
scanner.next基礎資料型別()

1. 以空格當作區隔
2. 以換行當作結束輸入
3. 若一開始讀取的不是資料而是空格或換行會嘗試掠過
4. 需小心執行完會殘留空格或換行在輸入緩衝區(主控台)

```
Scanner scanner = new Scanner(System.in);
System.out.print("請輸入(int):");
int intInput        = scanner.nextInt();
System.out.println("intInput:" + intInput);
```




Run
輸入緩衝區
請輸入(int): 123 456
輸入 123 456
int intInput = scanner.nextInt ()
輸入緩衝區
把值123讀出來
給 intInput
123 456\n
輸入 enter
輸入緩衝區
456\n
Run
請輸入(int): 123 456
intInput:123
int intInput2 = scanner.nextInt ()
輸入緩衝區
把值456讀出來
給 intInput2
456\n
Run
請輸入(int):
請輸入(int):intInput2:
請輸入(int):intInput2:456
輸入緩衝區
\n
無法輸入 數字
無法輸入 Enter



scanner.nextLine 用換行當區隔!!

第二章變數　運算子(位元)

```
int y = 9;
```

位元運算子	解釋功用	範例	備註
~	取補數	~y	代字號
&	位元**且**	y & 3	and
|	位元**包含或**	y | 3	or
^	位元**互斥或**	y ^ 3	xor
<<	向左位移(補零)	y << 2	
<<<	沒有這個東西喔		
>>	**保留**正負號 向右位移	y >> 2	
>>>	**去掉**負號(補零) 向右位移	y >>> 2	

```
                                  ~Y
0b00000000000000000000000000001001
0b11111111111111111111111111110110
```

```
y & 3     y | 3     y ^ 3
1001      1001      1001
&0011     |0011     ^0011
0001      1011      1010
```

```
    y=0b00000000000000000000000000001001
y<<2  y=0b00000000000000000000000000100100
y>>2  y=0b00000000000000000000000000000010
y>>>2 y=0b00000000000000000000000000000010

    nY=0b11111111111111111111111111110111
nY<<2 y=0b11111111111111111111111111011100
nY>>2 y=0b11111111111111111111111111111101
nY>>>2 y=0b00111111111111111111111111111101
```

將10進制-9轉為2進制
00000000000000000000000000001001 先轉2進制
因為為負數所以需要進行 **2的補數(反相加1)**
反相　　11111111111111111111111111110110
+1　　11111111111111111111111111110111

將2進制11111111111111111111111111110111轉為10進制
由於11111111111111111111111111110111 最高位為1,
所以轉換後的結果要補上負號
由於為負數, 所以須經過**2的補數(反相加1)**才知道是多少數值
反相　　00000000000000000000000000001000
+1　　00000000000000000000000000001001
為10進制-9

a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

AND 運算: 全部都為1，結果才為1

a	b	a|b
0	0	0
0	1	1
1	0	1
1	1	1

OR 運算:只要一個為1，結果就為1

a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

XOR 運算:只要兩個不一樣，結果就為1

```
Scanner scanner = new Scanner(System.in);
System.out.print("請輸入(int):");
int intInput = scanner.nextInt();
String intInputBS = Integer.toBinaryString(intInput);
System.out.println("intInput 二進制顯示: 0b" + intInputBS);

System.out.print("請輸入(float):");
float floatInput = scanner.nextFloat();
int iFloatToIntBits = Float.floatToIntBits(floatInput);
String iFloatToIntBitsBS = Integer.toBinaryString(iFloatToIntBits);
System.out.printf("floatInput 二進制顯示: 0b%32s\n", iFloatToIntBitsBS);
```

整數用

浮點數用

第二章變數　運算子優先順序

優先權	運算子								同等順序
(最高)1	()								**由右至左**
2	遞增++	遞減--	負號-	反相!	補數~				由左至右
3	乘*	除/	求餘數%						由左至右
4	加+	減-							由左至右
5	左移<<	右移>>	無正負號右移>>>						由左至右
6	小於<	大於>	小於等於<=	大於等於>=					由左至右
7	等於==	不等於!=							由左至右
8	AND&								由左至右
9	XOR^								由左至右
10	OR|								由左至右
11	邏輯AND&&								由左至右
12	邏輯OR||								由左至右
13	三元條件選擇?:								**由右至左**
14	指定運算=								**由右至左**
(最低)15	+=	-=	*=	/=	%=	&=	|=	^=	**由右至左**

同等順序:
運算式中如果遇到相同優先權的話依照同等順序方向運算。

```
y=2;
y=y+9*3/6*-1;
```

```
int i = 3;
i = (i<3)?(i>1)?2:1:3;
i = (i<3)?((i>1)?2:1):3;
```

字串 String 也可以相加喔!!

```
int x = 3;
String message = "Hello";
String message2 = "Bye";
message = message + " Tom";
message = message + " Tom" + " 說" + x + "次";
message = message + " Tom" + " 說" + x + 1 + "次";
message = message + " Tom" + " 說" + (x + 1) + "次";
```

指派運算子	解釋功用	範例	備註
=	設定值	message = "Hello"	

算術運算子	解釋功用	範例	備註
+	字串串接	message + " Tom"	Hello Tom

關係運算子	解釋功用	範例	備註
==	是否指向同一個位置	message == message2	判斷字串是否相等時，請使用equals()

如何判斷字串是否相等

語法: 字串.equals(字串)

範例: message.equals(message2)

第二章補充　位元(bit)與位元組(Byte)

- 1 Byte (位元組) 有 0000_0000 – 1111_1111 的狀態(2進制)
- 1 Byte (位元組) 有　　0 –　　255 的狀態(10進制)

10進制	2進制	16進制
0	00000000	00
1	00000001	01
2	00000010	02
3	00000011	03
4	00000100	04
5	00000101	05
6	00000110	06
7	00000111	07
8	00001000	08
9	00001001	09
10	00001010	0A

10進制	2進制	16進制
11	00001011	0B
12	00001100	0C
13	00001101	0D
14	00001110	0E
15	00001111	0F
16	00010000	10
17	00010001	11
18	00010010	12
19	00010011	13
20	00010100	14

10進制	2進制	16進制
255	11111111	FF
256	00000001 00000000	0100

0+0=0
0+1=1
1+0=1
1+1=10(進位)

本章節就是要學會怎麼用程式寫一個變數來存放這些資料

13 = 00000000 00000000 00000000 00001101

```
int apple2 = 0b00000000_00000000_00000000_00001101;

int apple = 13;
```

第二章補充　10進制轉2進制

2 | 23

求23除以2的商與餘數?

2 | 23
11

求商得11(不考慮小數)

2 | 23.....1
11

求餘數得1

2 | 23.....1
2 | 11

求11除以2的商與餘數?

2 | 23.....1
2 | 11
5

求商得5(不考慮小數)

2 | 23.....1
2 | 11.....1
5

求餘數得1

2 | 23.....1
2 | 11.....1
2 | 5

求5除以2的商與餘數?

2 | 23.....1
2 | 11.....1
2 | 5.....1
2

求商得2(不考慮小數)

2 | 23.....1
2 | 11.....1
2 | 5.....1
2

求餘數得1

2 | 23.....1
2 | 11.....1
2 | 5.....1
2 | 2

求2除以2的商與餘數?

2 | 23.....1
2 | 11.....1
2 | 5.....1
2 | 2
1

求商得1(不考慮小數)

2 | 23.....1
2 | 11.....1
2 | 5.....1
2 | 2.....0
1

求餘數得0

這裡為1時停止計算

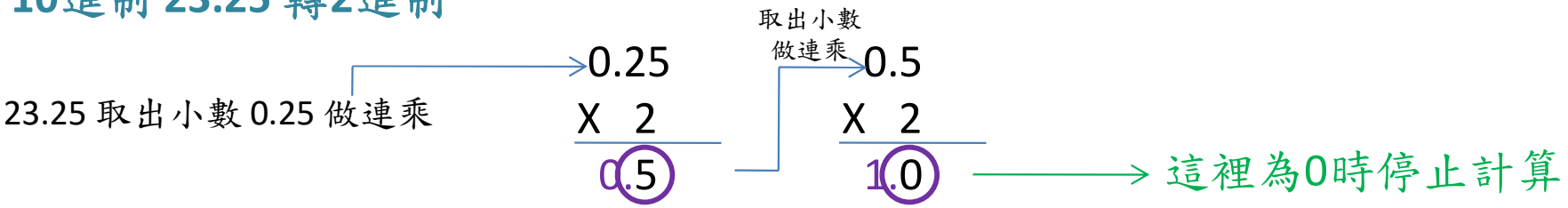
結果

2 | 23.....1
2 | 11.....1
2 | 5.....1
2 | 2.....0
1

依據紅色方向
排列結果
00010111即為
23的2進制表示

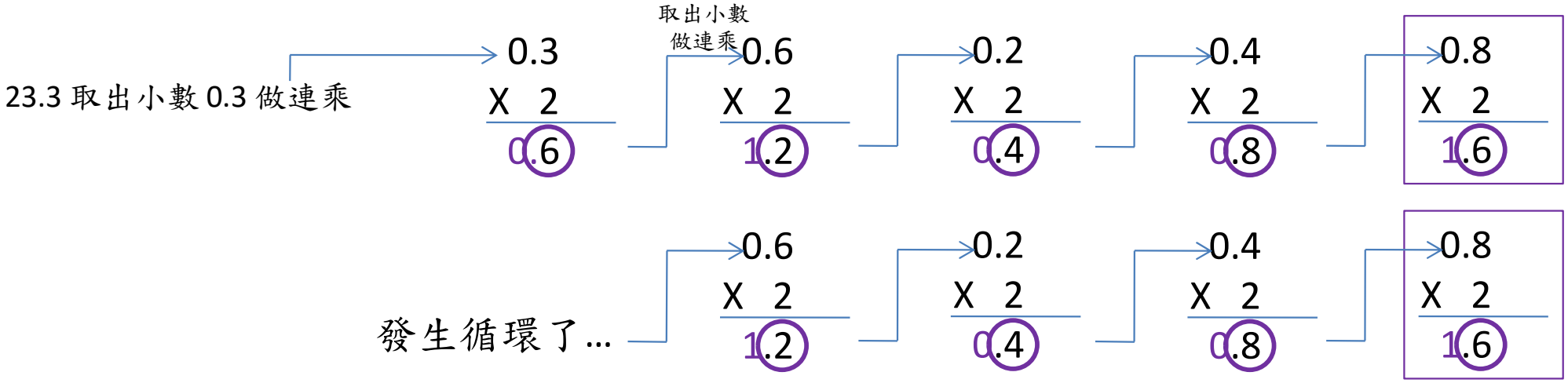
第二章補充　10進制(含小數點)轉2進制

10進制 23.25 轉2進制



用小數點合併10111與01為10111.01即為 23.25 的 2進制表示

10進制 23.3 轉2進制



用小數點合併10111與01001為10111.010011001即為 近似 23.3 的 2進制表示

第二章補充　2進制(含小數點)轉10進制

2進制 10111.010011001 轉10進制

1. 整數部分:該位數值 乘上 $2^{位數-1}$
2. 小數部分:該位數值 乘上 $2^{-位數}$
3. 分別計算以後用小數點串起就好

參考

$2^{11}=2048$
$2^{10}=1024$
$2^9=512$
$2^8=256$
$2^7=128$
$2^6=64$
$2^5=32$
$2^4=16$
$2^3=8$
$2^2=4$
$2^1=2$
$2^0=1$

$a^{-n} = 1/a^n$

整數部分

1	0	1	1	1
x	x	x	x	x
$16(2^{5-1})$	$8(2^{4-1})$	$4(2^{3-1})$	$2(2^{2-1})$	$1(2^{1-1})$
=	=	=	=	=
16	0	4	2	1

→ 16+0+4+2+1=**23**

小數部分

0	1	0	0	1
x	x	x	x	x
0.5 ($2^{-1}=1/2$)	0.25 ($2^{-2}=1/4$)	0.125 ($2^{-3}=1/8$)	0.0625 ($2^{-4}=1/16$)	0.03125 ($2^{-5}=1/32$)
=	=	=	=	=
0	0.25	0	0	0.03125

→ 0.25+0.03125=**0.28125**

23 + **0.28125** = 23.28125 近似 23.3

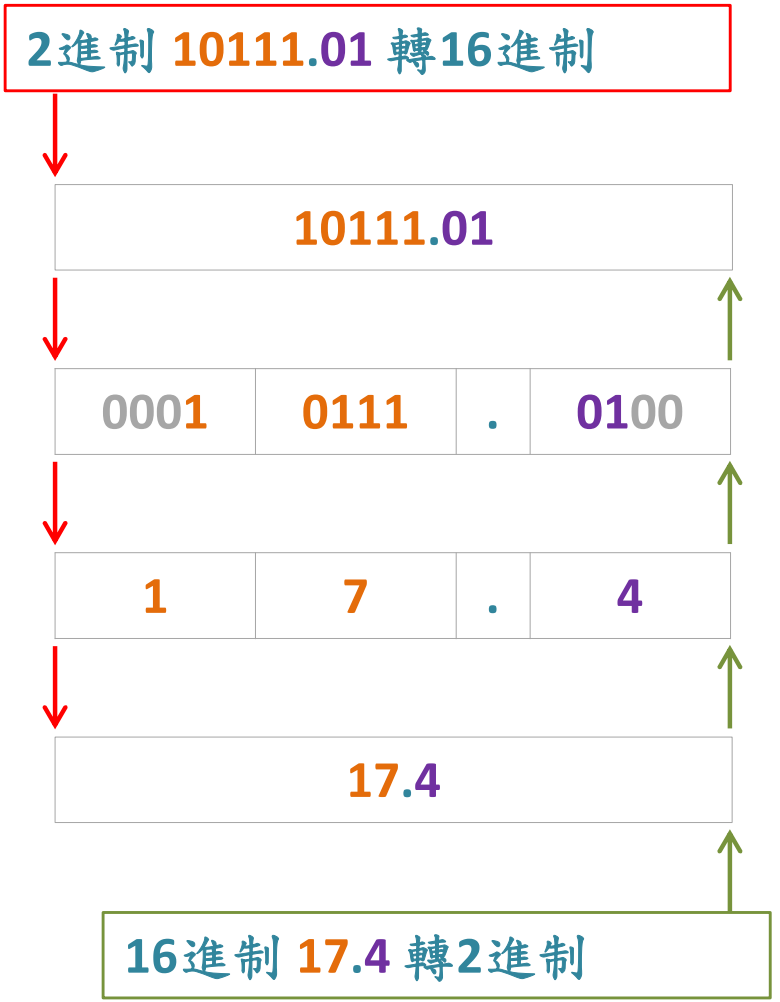
第二章補充　2進制與16進制互轉

2轉16

1. 以小數點為基準，向左取整數,向右取小數
2. 整數部分的**位數**如果不是**4的倍數**則往**前**用 0 補齊
3. 小數部分的**位數**如果不是**4的倍數**則往**後**用 0 補齊
4. 每4個位數為一組並轉成10進制

16轉2

1. 將每個位數的數值,個別轉成2進制即可



第二章補充　如何用2進制表示負數(2的補數)

2的補數:

1. 一種用2進制表示有號數的方法，概念為將數值範圍切成兩半(一半給正數,一半給負數)
2. 正數的2進制轉負數的2進制產生方式:
   - A. 將2進制反相(也稱1的補數)
   - B. +1　(避免有負零這種東西)(+1以後不考慮溢位)(2的補數)
3. 因此我們可以透過判斷2進制最高位(32bit)是否為1來得知是否為負數
4. 可以用+法取代減法的方式

無號 (以4bit為例)

10進制	2進制
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

有號 1的補數

10進制	2進制
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-1	1110
-2	1101
-3	1100
-4	1011
-5	1010
-6	1001
-7	1000
0	1111

這七個表示正數

這七個表示負數

有號 2的補數

10進制	2進制
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

這七個表示正數

這八個表示負數

Q.何謂補數?
A.兩數相加為 某數 則稱
兩數 互為 某數 的補數

3,7 相加為 10 則稱
3,7 互為 10 的補數

使用補數讓電腦可以透過加法器來執行減法運算(省了減法器)

```
 8          8
-3    ⇒    +7
 5          15
            ↓
          溢位刪除
```

```
 4          4
-7    ⇒    +3
-3          7
            ↓
```

1.沒有發生溢位,表示結果為負值
2.然後結果再取一次補數3
3.所以為 -3

```
 4
+3
 7
```

沒有溢位取補數

-3

第二章補充　如何用2進制表示負數(2的補數)

1. 負數的10進制轉負數的2進制產生方式:
   A. 先不考慮負號直接將10進制轉2進制
   B. 將2進制反相(也稱1的補數)
   C. +1　(避免有負零這種東西)(+1以後不考慮溢位)(2的補數)

2. 負數的2進制轉負數的10進制產生方式:
   A. 觀察最高位元是否為1，確定為1則轉完成10進制後要記得補負號
   B. 將2進制反相(也稱1的補數)
   C. +1　(避免有負零這種東西)(+1以後不考慮溢位)(2的補數)
   D. 執行正數2進制轉10進制
   E. 補上負號

有號2的補數

10進制	2進制
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

這七個表示正數

這八個表示負數

第二章補充　浮點數存在記憶體的格式到底是什麼?

- 浮點數存在記憶體的格式是依據 IEEE754 的規範

正負號	指數偏移值	尾數(分數,小數)
1個bit	8個bits	23個bits

IEEE754的float

- 正負號(Sign)
  - 1: 負數
  - 0:正數
- 指數偏移值(exponent bias)
  - 範圍為 -126~127。　(-127和128被用作特殊值處理)
  - 這裡存放規則為指數的實際值加上127　(方便其他運算)
- 尾數(Mantissa, fraction)
  - 存放小數資料的地方

- 使用 IEEE 754 表示 21.5
  - 先轉2進制 $21.5_{(10)} = 10101.1_{(2)}$
  - 為正數 Sign = 0
  - 正規化(整數只剩個位數): $10101.1 = 1.01011 \times 2^4$ 因此得到指數為4
  - 將指數為 $4 + 127 = 131_{(10)} = 10000011_{(2)}$ 即exponent bias = 10000011
  - 將1.01011 去掉1，保留 01011 即 Mantissa = 01011

正負號	指數偏移值	尾數(分數,小數)
1	10000011	0101100 00000000 00000000

參考
https://www.h-schmidt.net/FloatConverter/IEEE754.html
https://zh.wikipedia.org/zh-tw/IEEE_754

第二天複習

1. 資料型別記憶體存取方式分類
   - 基礎資料型別: byte, short, int, long, float, double, char, Boolean
   - 參考資料型別: String, 物件(需要new)

2. 強制型別轉換
   - 使用 (資料型別)變數

3. 運算子與運算元的注意事項
   - ++x 與 x++ 差異
   - 運算子的優先順序
   - (++a ==1) && (++b==3) 須注意
     - 若左邊 ++a==1為假, 就不會執行到 右邊 ++b==3 的程式
     - 若左邊 ++a==1為真, 就執行 右邊 ++b==3 的程式
   - 三元運算子 ?:
   - 基本布林運算 AND, OR, XOR

4. Scanner 輸入

```
boolean boolInput    = scanner.nextBoolean();
short shortInput     = scanner.nextShort();
int intInput         = scanner.nextInt();
long longInput       = scanner.nextLong();
float floatInput     = scanner.nextFloat();
double doubleInput   = scanner.nextDouble();

String stringSpaceInput = scanner.next();
String stringLineInput  = scanner.nextLine();
char charInput          = scanner.next().charAt(0);
```

5. 輸入緩衝區注意事項
   - 輸入需要符合 Scanner預期的型別
   - 殘留換行字元可嘗試使用 `scanner.nextLine();`清空

6. 二進制轉換
   - 整數 10轉2, 2轉10
   - 小數點 10轉2, 2轉10
   - 負數: 1的補數與2的補數
   - 浮點數儲存格式 IEEE754