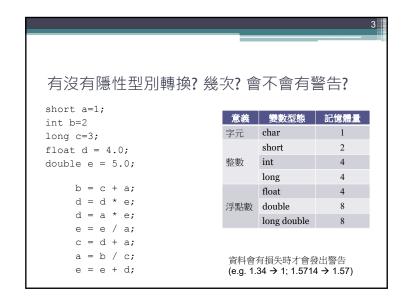
```
#include <iostream>
                           using namespace std;
回顧
                           int main() {
• 第一個程式
                              return 0;
 。 基本架構 → 五行必寫的公式
 註解的寫法、cout、<<、endl</li>
 。可用 sizeof() 取得一變數或型別
   的記憶體使用量。
· 變數 (variables)
 。變數名稱 - 限制?
 變數型別 (type)
   · 整數型別 - int, short, long
   · 浮點數型別 - float, double
 。在一列運算的敍述中型別不同時的電腦運算處理
```



基本變數型態 basic variable type 變數型態 記憶體量* 數值範圍 準確度 (Accuracy) 準確 字元 char $-128 \sim +127$ short int ** 2 -32768 ~ 32767 進確 $-2^{31} \sim 2^{31}-1$ 準確 整數 int long int *** $-2^{31} \sim 2^{31}-1$ 準確 -3.4x10³⁸~+3.4x10³⁸ 6-7位有效數字 float 浮點數 double -1.7x10308~+1.7x10308 14-15位有效數字 -1.7x10³⁰⁸~+1.7x10³⁰⁸ 14-15位有效數字 long double *單位為 bytes。記憶體量與所使用之作業系統、硬體、以及編譯器等皆有關係, 本表則依據win32作業系統之 Visual C++ 9.0。 ** 可以簡寫為 short, int 可省略不寫 *** 可以簡寫為 long, int 可省略不寫

```
Α
                                               65
1-4.cpp
                                               1
#include<iostream>
using namespace std;
int main() {
                      宣告一個變數 a, 並初始為 'A' 字元
  char a = 'A';
                                      輸出 a 變數的值
  cout << a << endl;
                               以整數方式輸出 a 變數的值
  cout << (int) a << endl;
                                     將 67 存入 a 變數
  a = 67;
                                      輸出 a 變數的值
  cout << a << endl;</pre>
                               輸出 a 變數所佔記憶體大小
  cout << sizeof(a) << endl;</pre>
                               輸出 char 型別所佔記憶體
  cout << sizeof(char) << endl;</pre>
  return 0;
                               大小
```

5

說明

- 字元型別為一特殊的整數型別,其實質資料為一個整數,此例來說以 美國國家標準碼 (ASCII, American Standard Code for Information Interchange) 存入與輸出。
- cout << (int) a << endl;
- 。將變數 a 「強制型別轉換」或「顯性型別轉換」為 int 型別後輸出,因此會看到輸出一整數 65 (代表 A)
- a = 67;
- 。將 67 存入變數 a,此處會發生隱性型別轉換,將 67 轉換為字元型別後存入 a。在 C/C++ 中直接寫出來 (e.g. 67) 的整數資料為 int 型別。
- sizeof()用來取得小括弧內的變數或是型別名稱所需的記憶 體空間大小,單位為 bytes。

7

基本變數型態 - 整數 (integer)

- 三種整數: short int、int、long int 或是 short、int、long
- 。 記憶體用量與 CPU & 編譯器 (compiler) 有關
- 一般來說 int: 4 bytes (32 bits → 2³²=4,294,967,296 ~ 40億)
- short (int): 短整數, 佔用記憶體量 <= int
- 。 long (int): 長整數, 佔用記憶體量 >= int
- Modifier (修飾字): unsigned vs. signed
- unsigned int signed short...
- 有號數或無號數
- 。當資料不可能為負值時,可使用無號數,使可代表的數值大小擴大 一倍。

6

基本變數型態 - 字元 (<u>char</u>acter)

- 它其實是一種整數,但它的意義是給人看的「符號」。
- 所以字元型別在存入變數或輸出 (e.g. 螢幕、檔案) 時,會作特別的處理。
- 每個電腦上看到的「符號」背後都依循某一種編碼標準
 - 。 英文字母: ASCII (American Standard for Information Interchange)
 - 。 正體中文: Big-5、UTF-8 (萬國碼)
- 。簡體中文:GB-2312
- 有了統一的編碼標準,電腦之間才能進行資料的交換

8

基本變數型態 - 實數/浮點數

- real number / floating point number
- 三種浮點數: float, double, long double
 - 。 佔用記憶體量與 CPU & 編譯器 (compiler) 有關
 - □ 目前一般來說 double: 8 bytes (64 bits)
 - float: 佔用記憶體量 <= double
 - 。 long double: 佔用記憶體量>= double
- IEEE-754 規定了浮點數在記憶體裡的儲存方式。
- Example:
- 。 float a; ← 宣告變數 a , 型別是浮點數
- 。 double b = 3.0e8; ← 宣告變數 b, 並初始化為 3*108

```
1-5.cpp

#include<iostream>
using namespace std;
int main() {
    short int a = 32767;
    unsigned short b = 65535;

    cout << a << ", " << b << endl;
    a = a + 1;
    b = b + 1;
    cout << a << ", " << b << endl;
    -32768, 0

    return 0;
}

int main() {
    short int a = 32767;
    unsigned short b = 65535;

    cout << a << ", " << b << endl;
    -32768, 0

    id 理所看到的情况稱
    為溢位 (overflow)
```

```
О
                                                1
1-7.cpp
                                               1
#include<iostream>
using namespace std;
int main() {
                             宣告一布林變數 a,並初始為 false
 bool a = false;
                             宣告一布林變數 b,並初始為 true
 bool b = true;
  bool c = 100;
                             宣告一布林變數 c,並初始為100
                               輸出變數a的值
  cout << a << endl;
                               輸出變數b的值
  cout << b << endl;
  cout << c << endl;</pre>
                               輸出變數c的值
  return 0;
           在 C/C++ 中,非 o 值為真 (true), o為假 (false)
```

14

Lecture 2

變數 (續)C++ 字串 (string)算術運算與布林運算

```
2-1.cpp

#include<iostream>
using namespace std;

int main() {
  int a=3, c;
  const int b=4;

  c = a + b;
  // b = 5;
  cout << "a+b=" << c;
  return 0;
}</pre>
```

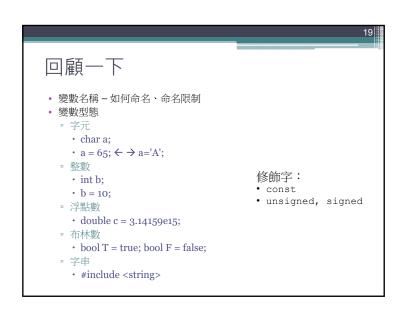
常數變數 (constant variable)

- 在宣告變數時,對於整數的宣告可使用修飾字 unsigned/signed 將一個整數或字元型態的變數宣告為無 號數或有號數。
- 另一個修飾字 const , 可用於所有變數型別,讓所宣告出來的變數成為常數變數。
- Example:
- const double a = 3.14159;
- 在宣告常數變數時,必需同時進行初始化的動作。
- 程式內若試圖對常數變數進行修改,將導致編譯錯誤。

16

字串 (string)

- 在 C++ 裡有兩種字串
- 。C-字串,之後會介紹,由C語言繼承過來。
- 。C++字串,方便易用,但不能用在C語言。
- C++字串型別為 string、需要 #include <string>
- 記得在 C/C++ 裡字串前後需要加雙引號 "



一些常見的跳脫字元 (escape sequence) \(n \text{ new line (換行)} \\ b \text{ back space (擦去前一個字)} \\ r \text{ carriage return (回到列開頭)} \\ a \text{ alert (警告音)} \\ t \text{ horizontal tab (水平定位)} \\ " 雙引號 \\ " 單引號 \\ backslash (反斜線) \\ o \text{ null character} \\ xhh \text{ 以 16 進位方式指定字碼 (e.g. \x41 → 'A')}

回答以下問題

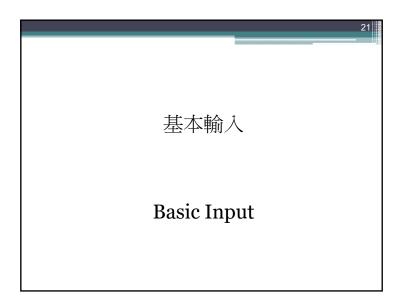
• 我今天想要儲存 3.141592654,應該選用何種變數型別?

• 今天有一個資料是 512,那些變數型別可以使用?

• 今天我想要儲存一個字元 'q',可以使用何種變數型別?

• -4.2 可以使用何種變數型別儲存?

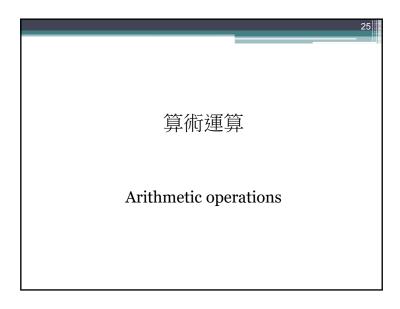
• 如何確定讓一個變數的值不會在程式執行時被修改?



```
2-3.cpp
 #include <iostream>
 using namespace std;
                              若輸入的 a = 3, b=5, 輸出的結果?
  int a, b;
                                     注意到 \n,其中 \稱
  cout << "請輸入 a: ";
                                     為 escape sequence,
   cin >> a;
                                     主要用來輸入一些鍵盤
   cout << "請輸入 b: ";
                                     無法直接輸入的字元。
   cin >> b;
   cout << "\na+b=" << a+b;
   cout << "\na-b=" << a-b;
   cout << "\na*b=" << a*b;
   cout << "\na/b=" << a/b;
  cout << "\na%b=" << a%b << endl;</pre>
```

```
基本輸入

• cout (console output) 用來將資料輸出至螢幕上。
int a=3;
cout << a;
• cin (console input) 用來從鍵盤輸入資料,經過適當地轉
換成為目的變數的型態後,存入目的變數內。
int a;
cin >> a;
• 在此, << 、 >> 稱為串流運算子 (stream operator)。
• cin, cout 都可以連結多個串流運算子進行輸入輸出的操
作。
```



```
2-5.cpp

#include <iostream>
using namespace std;
int main() {
  int a=14;
  int b=4;

  cout << "a= " << a << endl;
  cout << "b= " << b << endl;
  cout << "a+b=" << a+b << endl;
  cout << "a-b=" << a+b << endl;
  cout << "a'b=" << a'b << endl;
  cout << "a'b=" << a'b- << endl;
  cout << "a'b=" << a'b- << endl;
  cout << "a'b=" << a'b- << endl;
  cout << "a'b- << endl;
  cout << endl;
  cout << "a'b- << endl;
  cout << endl;
  cou
```

算術運算 arithmetic operations • 我們在 1-2.cpp 中已看過了兩個運算子 (+, =) 以進行算術運算。 int a=3; int b, c; b = 4; c = a + b; cout << "a+b=" << c; • 在 C/C++ 的運算中有以下重點: 。 運算子 (operator, 操作) 與運算元 (operand, 被操作的資料) 。 算術運算子的優先順序 。 型別轉換 (type conversion)

試試看

- •請將以上程式的 a 改成 3, a/b 的結果?
- 將以上程式的變數形別改成 char, 結果?
- · 將以上程式的變數形別改成 double, 結果?
- 在每一個輸出的字串裡加上"\a"

```
二元算術運算子
=:指定(assignment)
                              a = 3 + 2; // a=5
+:相加 (addition)
                              b = 3 - 2; // b=1
-:相減 (subtraction)
                              c = 3 * 2; // c=6
*: 相乘 (multiplication)
                              d = 3 / 2; // d=1 ?!
/: 相除 (division)
                              e = 3 \% 2; // e=1
%:取餘數 (modulus)
       運算的順序上與數學相同: 先乘除後加減, % 視為除法運算
          • 3 + 2 * 6 = 15
          \cdot 3 \cdot 6 + 3 = 21
          • 3 * 6 / 2 = 9
       也同於數學運算式,可用括號 (...)來改變運算的順序
          \cdot (3 + 2) * 6 = 30
          \cdot 3 * (6 + 3) = 27
          \cdot (3 \cdot 6) / 2 = 9
```

```
... 32766 32767 -32768 -32767 -32766 ...
2-7.cpp
                         (-30000 - 10000) - (-32768) = 7232
                                   32768 + (-7232) = 25536
 #include<iostream>
                            此種數字超出範圍的狀況, 專業術語稱為
                                      Underflow (向下溢位)
int main() {
                                     若是加上去超出範圍,稱為
  short a = -30000;
                                          Overflow (溢位)
  short b=10000;
  short c = a - b;
   short d = a - b / 1000;
   short e = (a - b) / 1000;
   cout << "a=" << a << endl;
                                 a = -30000
   cout << "b=" << b << endl;
                                 b=10000
   cout << "c=" << c << endl;
                                 c=25536
  cout << "d=" << d << endl;
                                 d=-30010
  cout << "e=" << e << endl;
                                 e = -40
  return 0;
```

```
2-6.cpp

#include <iostream>
using namespace std;
int main() {
  int a=2;
  int b;

  b = a + 3 * a;
  cout << "b=" << b << endl;

  b = b + 3 * a;
  cout << "b=" << b << endl;

cout << "b=" << b << endl;

cout << "b+3 is " << b+3 << endl;
  cout << "b=" << b << endl;
  cout << "b+3 is " << b << endl;
  cout << "b+3 is " << b << endl;
  cout << "b=" << b << endl;
  cout << "b=" << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-2 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << " <> endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b << endl;
  cout << "b-1 is " << b <<
```

```
—元運算子 (Unary Operator)

- 元運算子即僅需一個運算元即可進行計算的運算子
- ++: 將運算元的值遞增 (increment),在算術運算時即為+1
- --: 將運算元的值遞減 (decrement),在算術運算時即為-1

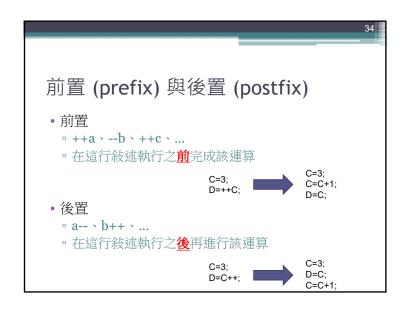
- 範例:
A=3;
A++; ← A變為4
A--; ← A的值變為3

- 但是!
```

```
2-8.cpp
#include <iostream>
using namespace std;
int main() {
  int A=5;

  cout << A++ << endl;
  cout << ++A << endl;
  cout << A++ << endl;
  cout << A++ << endl;
  cout << A+ << endl;
  cout << A << endl << endl
```



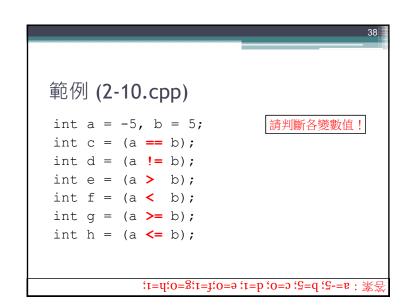


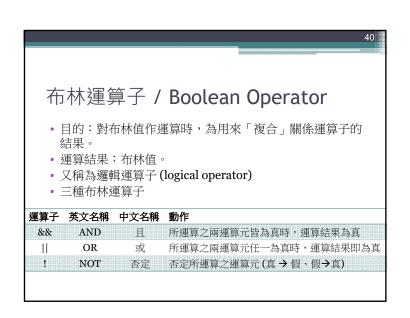


```
範例 (2-9.cpp)

int a = 10, b = 4;
bool c = (a == b);
bool d = (a != b);
bool e = (a > b);
bool f = (a < b);
bool g = (a >= b);
bool h = (a <= b);
```







```
節例 (2-11.cpp)

bool a = false;
bool b = true;
bool c = (a && b);
bool d = (a || b);
bool e = ! a;
```

```
節例 (2-12.cpp)

bool a = true;
bool b = true;
bool c = (a && b);
bool d = (a || b);
bool e = ! a;

served and provided in the ser
```

真值表 (truth table)

- 真值表列出了各種邏輯運算的結果
- 在C/C++語言中,非o為真(true),o為偽(false)

&&	0	1
0	0	0
1	0	1

II	0	1
0	0	1
1	1	1

小結

- 關係運算子與布林運算子常放在條件敍述句(之後介紹)中用來控制流程/選擇性地執行程式
- 關係運算子:==、!=、>、<、>=、<=
- 布林運算子: &&、||、!

45

3/3隨堂練習

無論是樑或是柱都會因為溫度的改變而有熱脹冷縮的效果。請撰寫一程式,讓使用者輸入1)目前溫度 T_0 、2) 樑或柱的目前長度 L_0 、3)樑或柱的溫度係數 α 、以及4) 想要知道樑或柱長度的溫度 T。並使用以下公式輸出溫度 T時梁或柱的長度:

$$L = L_0 \left(1 + \alpha \left(T - T_0 \right) \right)$$

請透過以下網址繳交:

http://140.118.105.174/Courses/CVB/2017/system/hw01.php

47

補充

作業一

Due: 3/10/2017

有天就讀台科的小新去三餐的7-11買了 x 條七七乳加、 y 罐 多多和 z 杯中熱美,請寫一程式,依序輸入 x, y, z 以及各項 單品價格,並幫小明計算出總花費為多少。

p.s. 小明有攜帶學生證,享九折優惠!!

請透過以下網址繳交:

http://140.118.105.174/Courses/CVB/2016/system/lab-20160304.php

輸出格式的設定

Formatting Output

輸出格式設定 iomanip • 使用 cout 輸出資料時,內定格式依資料型別與數字大小決定。 • 若要設定數字的格式,可使用一些定義於 iomanip 標頭檔內的功能: • setprecision(n): 設定接下來所有輸出資料的精確度為 n • setw(n): 設定下一個輸出資料的最小寬度為 n • setfill(ch): 設定接下來所有輸出有寬度時,空白地方所填入的字元

```
2-14.cpp

#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    cout << setw(5) << "*" << endl;
    cout << setw(4) << "*" << endl;
    cout << setw(3) << "*" << endl;
    cout << setw(2) << "*" << endl;
    cout << setw(1) << "*" << endl;
    return 0;
}
```

```
| E程式開始加入以下程式碼看看發生什麼事?
| Cout << left; | Cout
```

串;	流操控器	效果		
set	w(n)	將欄位寬度設定為n。		
set	fill('*')	設定欄位內空白處填入的字元。		
lef	t	在欄位內靠左對齊。		
right 在欄位內靠右對齊。		在欄位內靠右對齊。		
-	串流操控器	效果	範例	
	showpos	永遠顯示正負號,即正數之前加上+ 號。	+1	
	noshowpos	不強制顯示正號。(預設值)	1	
int	dec	以10進位法表示。(預設值)	17	
	oct	以8進位法表示。	21	
hex noshowbase showbase	以16進位法表示。	11		
	不顯示數字的基底格式 (預設值)	11		
	showbase	顯示數字的基底格式	0x11	

	串流操控器	效果	範例
	setprecision(n)	將浮點精度設為 n 位數。(內定為6)	, , , ,
	fixed	使用一般的浮點數表示法。	12.300000
float, double	scientific	使用科學記號表示法。	1.230000e+001
t, do	showpoint	永遠顯示小數點	1.0
float	noshowpoint	不強制顯示小數點 (預設值)	1
	showpos	永遠顯示正負號,即正數之前加 上+號。	+1
	noshowpos	不強制顯示正號	1
	串流操控器	效果	範例
bool	boolalpha	以true、false字串表示bool值。	true
	noboolalpha	以1、0表示bool值(預設值)。	1

```
|12345678901234567890
                                   100|
                            |+100|
2-16.cpp
                            | 64|
                                   0x64|
                            |+100|
                                100|
                                0144|
const int i=100;
cout << "|12345678901234567890" << endl;
cout << "|" << setw(10) << i << "|" << endl;
cout << "|" << showpos << i << "|" << endl;
cout << "|" << setw(5) << hex << i << "|" << endl;</pre>
cout << "|" << setw(10) << showbase << i << "|" << endl;</pre>
cout << "|" << dec << left << i << "|" << endl;
cout << right;
cout << "|" << setw(7) << noshowpos << i << "|" << endl;</pre>
cout << "|" << setw(8) << oct << i << "|" << endl;
```

```
2-17.cpp
double pi=1;
cout << showpoint << pi << endl;</pre>
                                              1.00000
cout << noshowpoint << pi << endl;</pre>
pi = 314.159265358979323846;
cout << pi << endl;</pre>
                                              314.159
cout << scientific << pi << endl;</pre>
                                              3.141593e+002
cout << fixed << pi << endl;</pre>
                                              314.159265
cout << setprecision(10);</pre>
cout << setw(6) << scientific << pi << endl; | 3.1415926536e+002</pre>
cout << setw(6) << fixed << pi << endl;</pre>
                                              314.1592653590
```