1

回顧

- 算術運算
  - 二元運算子: +, -, *, /, =, %

```
int a = 3, b = 4;
int c;
c = a + b;
c = b - a;
c = c % 3;
b = a * c;
```

  - 一元運算子: --, ++
    - 前置、後置

```
int a = 3;
cout << a++;
cout << ++a;
int b = a++;
cout << a;
```

2

運算子

- 指定運算子: =
- 算術運算子: +、- 、 * 、 / 、 %、++、--
- 關係運算子: ==、!= 、> 、 < 、 >= 、 <=
- 布林運算子: &&、 || 、!
- *複合指定運算子:* +=、-= 、 *= 、 /= 、 %=

3

補充：輸出格式的設定

（第二次的課程講義）

- 請大家自行參閱，考試也不會考，但寫作業時使用到會使輸出較為美觀。
  - 設定小數輸出的位數
  - 數字的對齊控制
  - 排版
  - 整數輸出方式的方式
  - 浮點數的輸出設定

Lecture 3

複合指定運算子
運算子的優先順序
流程控制 – if (I)

5

複合指定運算子

Compound Assignment Operator

6

複合指定運算子

- 在 C/C++ 裡，複合指定運算子可以讓程式撰寫人員減少打字 ...

```
A = A + 3; → A += 3;
B = B - 4; → B -= 4;
C = C * 5; → C *= 5;
D = D / 2; → D /= 2;
E = E % 6; → E %= 6;
```

- 以上這些運算子 (+=, -=, *=, /=, %=) 稱為複合指定運算子

7

3-1.cpp

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
  int a = 3, b = 4, c = 5, d = 6, e = 7;

  a += 3;
  b -= 4;
  c *= 2;
  d /= 3;
  e %= 2;

  cout << "\na: " << a
    << ", b: " << b
    << ", c: " << c
    << ", d: " << d
    << ", e: " << e;
  return 0;
}
```

a: 6, b: 0, c: 10, d: 2, e: 1

8

運算子優先順序

Precedence of operators

9

運算子優先順序

- 當一個程式敘述內含多個運算子時，何者先算、何者後算乃根據運算子的優先順序 (下頁中的 level) ，當順序相同時則根據表中的 grouping 決定運算方向。
- 除了先乘除後加減這個大家從小學都知道的規則外，其它在寫運算敘述時，以括弧將想要先運算的部份括起來，可以避免「驚奇」的發生。
- a = 3 + 4 – 5;
  - 查表可知 + - level 為 7、= level 為17
  - + - 同 level, 根據 grouping 可知運算方向由左至右
  - 3 + 4 → 7
  - 7 – 5 → 2
  - a = 2;

10

運算子優先順序 (level) 與運算方向 (grouping)

Level	Operator	Description	Grouping
2	() ++ --	postfix	Left-to-right
3	++ -- !	unary (prefix)	Right-to-left
4	(type)	type casting	Right-to-left
6	* / %	multiplicative	Left-to-right
7	+ -	additive	Left-to-right
8	<< >>	shift	Left-to-right
9	< > <= >=	relational	Left-to-right
10	== !=	equality	Left-to-right
14	&&	logical AND	Left-to-right
15	||	logical OR	Left-to-right
17	= *= /= %= += -=	assignment	Right-to-left

http://www.cplusplus.com/doc/tutorial/operators/

11

範例

b = c = 8; (運算方向由右至左 → 8 存入 c, c 存入 b)

c *= 3 + 3 * 2 > b --;

- *=: level 17
- *: level 6
- +: level 7
- >: level 9
- -- (postfix): level 2

c *= 3 + 3 * 2 > (**b --**) → c *= 3 + (**3*2**) > 8

→ c *= (**3 + 6**) > 8 → c *= (**9 > 8**) → c *= 1;

結果: b = 7, c = 8;

12

流程控制 – if
分支

Flow Control – if
Branching

13

流程控制

- 有時候，電腦程式需根據資料或運算結果的不同作出不同的反應。
  - 如果成績小於 60 分 → 被當掉了，該科目要重修
  - 否則該科目過了，學分拿到。
- **if (...) { ... } else { ... }**
- **if(...) { ... } else if (...) {...} else if (...) {...} ...**
- switch (...) case ...

14

3-2.cpp

```
#include <iostream>
using namespace std;
int main() {
  int score;

  cout << "請輸入成績: ";
  cin >> score;

  if(score >= 60) {
    cout << "及格囉！" << endl;
  }
  else {
    cout << "當掉囉！" << endl;
  }
  return 0;
}
```

若 score >= 60 為真，則

否則

請輸入成績: **75**
及格囉！

請輸入成績: **59**
當掉囉！

請輸入成績: **19**
當掉囉！

注意到，不是每一行程式都會被執行了 → 分支！

15

if ()

- 如果...就...否則就
  - 如果分數大於等於60分，就成績及格，否則就不及格。
  - 如果成績不及格，就要用功。
  - 所以「否則就」的部份是可以省略的。

```
if (布林值、布林運算) {
  條件成立時執行的敘述
}
else {
  條件不成立時執行的敘述
}
```

```
if (分數大於等於60分) {
    成績及格
}
else {
    不及格
}
```

16

3-3.cpp

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

int main() {
  int a = 30;
  int b;

  cout << "\n本程式檢查 b 是不是 " << a << " 的因數。\n";
  cout << "請輸入整數 b:";
  cin >> b;
```

17

3-3.cpp

```
  cout << "\n" << b;
  if(a % b == 0) {
      cout << " 是 ";
  }
  else {
      cout << " 不是 ";
  }
  cout << a << " 的因數。" << endl;

  return 0;
}
```

本程式檢查 b 是不是 30 的因數。
請輸入整數 b:15.6

15 是 30 的因數。

本程式檢查 b 是不是 30 的因數。
請輸入整數 b:7

7 不是 30 的因數。

18

複合關係運算

- 有時候所需的條件是複雜的, 可能需要複合多種的關係運算子 – 透過邏輯運算子來複合多個關係運算的結果。
  - e.g. 如果我很餓 而且 我很有錢 → 吃牛排
  - e.g. 如果 我很餓 而且 我只有20元 → 吃泡麵
  - e.g. 如果 我不餓 或是 我沒錢 → 不吃東西
  - 如果 (A 且 B) 則 ...
  - 如果 (A 或 B) 則 ...
  - 如果 (非 A) 則 ...

19

3-4.cpp

```
#include <iostream>
using namespace std;
int main() {
  char ans;

  cout << "你今天高與嗎 (Y/N)? ";
  cin >> ans;

  if(ans == 'Y' || ans == 'y') {
    cout << "太好了 :-) \n";
  }
  cout  << "祝你有愉快的一天!\n";

  return 0;
}
```

你今天高與嗎 (Y/N)? n
祝你有愉快的一天!

你今天高與嗎 (Y/N)? p
祝你有愉快的一天!

你今天高與嗎 (Y/N)? Y
太好了 :-)
祝你有愉快的一天!

20

3-5.cpp

```
#include <iostream>
using namespace std;
int main() {
  int data;

  cout << "請輸入一介於 1 到 100 的數字: ";
  cin >> data;
  if( data >= 1 && data <= 100 ) {
    cout << "謝謝您的合作 \n";
  }
  else {
    cout << "你在找我麻煩哦 \n";
  }

  return 0;
}
```

請輸入一介於 1 到 100 的數字: 51
謝謝您的合作

請輸入一介於 1 到 100 的數字: 19
謝謝您的合作

寫成 1 <= data <=100 可以嗎?

請輸入一介於 1 到 100 的數字: 200
你在找我麻煩哦

21

1 <= data <= 100

1<=data<=100

(1<=data)<=100 (true)<=100 1<=100 true

(false)<=100 0<=100

Level	Operator	Description	Grouping
9	< > <= >=	relational	Left-to-right

22

3-6a.cpp

```
#include <iostream>
using namespace std;
int main() {
  int data;

  cout << "請輸入一介於 1 到 100 的數字: ";
  cin >> data;
  if( data < 1 || data > 100 ) {
    cout << "你在找我麻煩哦 \n";
  }
  else {
    cout << "謝謝您的合作 \n";
  }

  return 0;
}
```

23

3-6b.cpp

```
#include <iostream>
using namespace std;
int main() {
  int data;

  cout << "請輸入一介於 1 到 100 的數字: ";
  cin >> data;
  if( ! (data >= 1 && data <= 100)) {
    cout << "你在找我麻煩哦 \n";
  }
  else {
    cout << "謝謝您的合作 \n";
  }

  return 0;
}
```

24

多重條件複合時的運算順序

- if () 裡的條件敍述是從左向右分析的
  - if (A && B && C && D) { *分支甲* } else { *分支乙* }
    - 如果 A 不成立時, B, C, D 完全不會被理會 → *分支乙*
    - 如果 A 成立, B 不成立時, C, D 不會被理會 → *分支乙*
    - 如果 A, B 成立, 但 C 不成立時, D 不會被理會 → *分支乙*
    - 如果 A, B, C, 都成立時, 結果由 D 決定
  - if (A || B || C || D) { *分支甲* } else { *分支乙* }
    - 如果 A 成立時, B, C, D, 完全不會被理會 → *分支甲*
    - 如果 A 不成立, 但 B 成立時, C, D 不會被理會 → *分支甲*
    - 如果 A, B 不成立, 但 C 成立時, D 不會被理會 → *分支甲*
    - 如果 A, B, C 都不成立, 結果由 D 決定

25

範例

```
if(餓 && 想吃漢堡 && 在漢堡店) {
   吃漢堡
}
```

餓	想吃漢保	在漢堡店	結果
0	?	?	沒吃漢堡
1	0	?	沒吃漢堡
1	1	0	沒吃漢堡
1	1	1	吃漢堡
? 代表不重要，無論 0 或是 1 都不會影響結果			

當有多個條件以 **&&** 複合起來時，儘量把不會成立的條件放前面，可以加快程式執行的效率，因為後面的其它條件電腦不需要進行判斷

26

範例

```
if(餓 || 想吃漢堡 || 在漢堡店) {
   吃漢堡
}
```

餓	想吃漢保	在漢堡店	結果
1	?	?	吃漢堡
0	1	?	吃漢堡
0	0	1	吃漢堡
0	0	0	沒吃漢堡
? 代表不重要，無論 0 或是 1 都不會影響結果			

當有多個條件以 **||** 複合起來時，儘量把會成立的條件放前面，可以加快程式執行的效率，因為後面的其它條件電腦不需要進行判斷

27

多重分支

28

多重分支

- 有時，有很多不同的想要作的事情：
  如果現在又渴又餓的話，就買超值全餐;
  否則如果口渴的話，就買中杯可樂;
  否則如果肚子餓的話，就買漢堡;
  否則，就吹冷氣。

```
if ( … ) { … }
else if ( … ) { … }
else if ( … ) { … }
else { … }
```

- 如果寫成如下，會有什麼結果？
  如果現在又渴又餓的話，就買超值全餐。
  如果口渴的話，就買中杯可樂。
  如果肚子餓的話，就買漢堡；
  否則，就吹冷氣。

```
if ( … ) { … }
if ( … ) { … }
if ( … ) { … }
else { … }
```

29

3-7.cpp

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

int main() {
  int score;

  cout << "請輸入成績: ";
  cin >> score;

  if(score >= 90) {
    cout << "A";
  }
  else if(score >= 80) {
    cout << "B";
  }
  else if(score >= 70) {
    cout << "C";
  }
  else if(score >= 60) {
    cout << "D";
  }
  else {
    cout << "F";
  }

  return 0;
}
```

30

3-7a.cpp

若判斷後需執行的敘述僅有一個時 (一個分號)，則區塊符號 { ... } 可以省略。事實上，{...} 區塊存在的目的，是將多個敘述複合視為一個敘述。因此，區塊內有多個敘述時，這整個區塊稱為一複合敘述(compound statement)。

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

int main() {
  int score;

  cout << "請輸入成績: ";
  cin >> score;

  if(score >= 90)
    cout << "A";

  else if(score >= 80)
    cout << "B";

  else if(score >= 70)
    cout << "C";

  else if(score >= 60)
    cout << "D";

  else
    cout << "F";


  return 0;
}
```

31

3-8.cpp

輸入年齡可以辨認適合看什麼級別的電影

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

int main() {
  int age;

  cout << "請輸入年齡: ";
  cin >> age;

  cout << "您可觀賞";

  if(age<6) {
    cout << "普通級" << endl;
  }
  else if(age<12) {
    cout << "保護級" << endl;
  }
  else if(age<18) {
    cout << "輔導級" << endl;
  }
  else {
    cout << "限制級" << endl;
  }

  return 0;
}
```

32

if() 之用途

- 利用 if() 我們可以根據使用者輸入的資料或是計算的結果來決定那一部份的程式被執行到
- 常常我們利用 if() 來檢查使用者輸入資料的合理性 (e.g. weight >=0, age >=0, age <=200, ...))
- 利用 if() 檢查可用來根據計算的結果選擇性地執行不同的反應

33

巢狀分支

Nested If

34

巢狀式判斷

- 有時程式裡需要進行的判斷較為複雜，無法使用簡單的多重分支來表達。
- 有時，使用巢狀式判斷在撰寫程式時，程式比較容易寫
- 巢狀：一區塊的程式({ ... })被另一個區塊的程式({...}) 所包圍。

請問你要**冷飲**還是**熱飲**？
冷飲選項：
- **冰開水**
- **可樂：可口可樂還是百事可樂？**
- **冰奶茶：甜度？冰塊？**
- **冬瓜茶：冰塊**

熱飲選項：
- **茶：烏龍、香片、花茶**
- **咖啡：美式、摩卡、卡布奇諾**

35

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string coldHot;

  cout << "冷飲還是熱飲?";
  cin >> coldHot;

  if(coldHot == "冷飲") {
    // 冷飲

  }
  else if(coldHot == "熱飲") {
    // 熱飲

  }
  else {
    // 錯誤
  }

  return 0;
}
```

可樂：可口可樂還是百事可樂？
冬瓜茶、冰開水：冰塊

茶：烏龍、香片、花茶
咖啡：美式、摩卡、卡布奇諾

36

```
if(coldHot == "冷飲") {
  // 冷飲

}
else if(coldHot == "熱飲") {
  // 熱飲

}
else {
  // 錯誤
}

return 0;
}
```

可樂：可口可樂還是百事可樂？
冬瓜茶、冰開水：冰塊

```
string which;
cout << "那一種冷飲？";
cin >> which;
if(which == "可樂") {
  // 可樂的問題
}
else if(which == "冬瓜茶"  || which
== "冰開水") {
  // 冬瓜茶或冰開水的問題
}
else {

}
```

37

```
if(coldHot == "冷飲") {
  string which;
  cout << "那一種冷飲?";
  cin >> which;
  if(which == "可樂") {
    // 可樂的問題
  }
  else if(which == "冬瓜茶" || which == "冰開水") {
    // 冬瓜茶或冰開水的問題
  }
  else {

  }
}
else if(coldHot == "熱飲") {
  // 熱飲

}
else {
  // 錯誤
}

return 0;
}
```

茶：烏龍、香片、花茶
咖啡：美式、摩卡、卡布奇諾

```
string which;
cout << "那一種熱飲:";
cin >> which;
if(which=="咖啡") {
  // 咖啡的問題
}
else if(which == "茶") {
  // 茶的問題
}
else {

}
```

38

巢狀式 (nested) 的 if 架構

```
if(條件 A) {
  if(條件 B) {
      if(條件 C) {
            (A && B && C) 時執行的動作
      }
      else {
            (A && B && (!C)) 時執行的動作
      }
  }
  else {
      (A && (!B)) 時執行的動作
  }
}
else {
  (!A) 時執行的動作
}
```

39

3-9.cpp

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

int main() {
  int number;

  cout << "請輸入一個介於 1 - 10 之間的整數：";
  cin >> number;
```

40

3-9.cpp (續)

```
if(number >= 1 && number <= 10) {
   if(number%2 == 0) {
     cout << "你輸入的數字" << number << "是偶數";
   }
   else {
     cout << "你輸入的數字" << number << "是奇數";
   }
 }
 else {
     cout << "你輸入的數字" << number << "超出範圍";
 }

 return 0;
}
```

41

巢狀 if(){} else{} 的注意事項

- 若 if, else 欲執行的敍述僅一行，{} 區塊可省略，但小心出錯!!!
  - else 會與向上最靠近且未與 else 配對的 if 相關聯
- 建議都加上 {}, 以減低出錯機率

```
WRONG!

if (a > 0)
    if(b > 0)
        cout << "a>0 and b>0";
else
    cout << "a<=0";
```

```
CORRECT!

if (a > 0) {
    if(b > 0) {
        cout << "a>0 and b>0";
    }
}
else {
    cout << "a<=0";
}
```

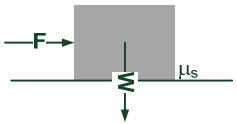
42

if() 之用途

- 利用 if() 我們可以根據使用者輸入的資料或是計算的結果來決定那一部份的程式被執行到
- 常常我們利用 if() 來檢查使用者輸入資料的合理性 (e.g. weight >=0, age >=0, age <=200, ...))
- 利用 if() 檢查可用來根據計算的結果選擇性地執行不同的反應

43

隨堂練習



當一靜置在平面上的物體受到的外力大於平面給它的靜摩擦力時，物體會開始滑動，其中材料於鋼鐵表面上的靜摩擦力如下表所示。請撰寫一程式由使用者依序輸入a) 物體重量 w、b) 作用於其中心之側向力 F、c) 材料 (1為鋼鐵、2為碳、3為石墨)、以及d) 表面狀況 (1為乾淨且乾燥、2為潤滑後)，接著列印出來使用者輸入的前述資料，輸出該塊物體是否會發生滑動。若不會滑動，並請輸出至少需要增加多少的側向力才能使該物體開始滑動。

材料	乾淨且乾燥的表面	潤滑後的表面
鋼鐵	0.65	0.16
碳	0.14	0.125
石墨	0.1	0.1

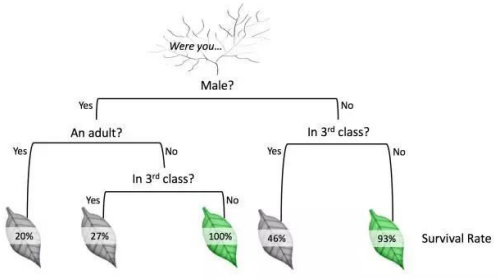
44

作業二

Due: 3/16/2017

作業

有人透過「資料探勘」的方式，得到了如下的預測模型，可以用來預測一個人是否在鐵達尼號事件中生存下來。請撰寫一程式，輸入 a) 性別 (1是男性、2是女性)、b) 大人或小孩 (1 是大人、2是小孩)、c) 是否在三等艙 (1為是、2為否)，並根據前面三個條件以及以下的模型，輸出其生存率。



https://read01.com/yJnAOO.html