IM1003: Programming Design, Spring 2017
Lab 02

Jhih-Bang Hsieh
bigelephant29

National Taiwan University

Outline

- HW1 Code Review
- if-else
- Logical Operators
- switch-case
- while, do-while
- for
- Nested Structure
- Practice

17010103

- 輸入一個整數，取其十位、百位
- 將十位、百位組成新數，求其平方

```
#include <iostream >
```

submit.cpp:1:21: fatal error: iostream : No such file or directory
compilation terminated.

- No such file: 沒有找到該檔案
- iostream 後面多打了一個空格所以找不到

```
#include <math.h>
```

- <math.h> 屬於 C，在 C++ 中要用 <cmath>

```
pow(((num0 % 1000) / 10), 2)
```

- pow 隱含轉型，會先將傳入值轉成浮點數，並回傳浮點數
- 整體隱含兩次轉型，後面的課程會提到轉型

```
if(n == 0, n == 1, n == 2)
```

- 會依序處理每一行
- 以最後一行作為 if 的判斷標準
- 別這樣寫 QQ

```
system("pause");
```

- 很多程式設計課程都會教這個函式
- system 其實是很危險的，盡量不要用

取一數十位、百位

- 假設該數為 n
- 取千位：$n_1 = n\%10000/1000$
- 取百位：$n_2 = n\%1000/100$
- 取十位：$n_3 = n\%100/10$
- 取個位：$n_4 = n\%10$

- $n_2 * 10 + n_3$
- $n\%1000/10$
- $n/10\%100$

輸出時高位補零

- 判斷該數有幾位
- $0 \leq n \leq 9 \rightarrow$ 補 3 個 0
- $10 \leq n \leq 99 \rightarrow$ 補 2 個 0
- $100 \leq n \leq 999 \rightarrow$ 補 1 個 0
- $1000 \leq n \leq 9999 \rightarrow$ 不用補 0

- in C:

```
printf("%04d", n);
```

- in C++:

```
#include<iomanip>
cout << setfill('0') << setw(4) << n;
```

17010104

- 大致上和第三題一樣
- 有了這週學的 for 迴圈，就輕鬆很多了！

Q & A

- 我跑出來的結果跟 PDOGS 上的不一樣，發生什麼事了？
  - PDOGS 有開啟 C++11 標準、O2 優化
  - -std=c++11 -O2
  - 編譯器版本、作業系統也會影響執行結果
- 如果環境不一樣，我要怎麼讓答案跟 PDOGS 一致？
  - 基本上邏輯正確、用法正確，答案就會一樣
  - 不一樣的地方在於某些預設行為（初始化），以及編譯優化
  - 換句話說，大部分不一樣的情形，可能是誤用或觀念錯誤
  - 所以要乖乖上正課跟助教課！
- 我寄信給助教，結果助教回信對我一點幫助都沒有 QQ
  - 幫 QQ
  - 只要問到上課、助教課有提過的東西，助教基本上不會幫忙
  - 問到新東西、觀念錯誤、系統操作問題，助教會試著回信
  - 請先自己 google，用網路上的資源解決問題

if-else

- 三個組成元件：if、else if、else
- if 為一切的開頭
- else if 只能接在 if 或 else if 後面
- else 只能接在 if 或 else if 後面
- 一旦寫了 else，後面就不能再接任何條件

- 如果今天是晴天，帥哥哥會想出去運動。(if)
- 否則如果今天是陰天，帥哥哥會想出去逛街。(else if)
- 否則的話，帥哥哥會想宅在家裡打 LOL。(else)

if-else

- 你不會劈頭就說「否則」，同理 else if、else 不會當作條件式的開頭。
- 你可以一直有各種特例，所以有很多「否則如果（else if）」是很合理的。
- 最後一個否則，如果沒有帶有條件，則囊括所有其他例外。
- 囊括所有例外的否則（else）最多只有一個。

Logical Operators

- Logical and: &&
- Logical or: ||
- Logical not: !
- Unary Operator（單元運算子）: !
- Binary Operator（二元運算子）: ||、&&
- 不確定運算優先順序的話，記得加個括號！

switch-case

```
1  switch(operand) {
2      case value_1:
3      case value_2:
4          // do something
5      break;
6      case value_3:
7          // do something
8      break;
9      default:
10         // otherwise, do something
11 }
```

- case 跟 case 之間記得要 break。
- 多條件可以省略 break，如範例的 value_1、value_2。

switch-case

```
1  switch(n) {
2      case 1:
3      case 2:
4          int tmp;
5          break;
6      case 3:
7          int tmp;
8          break;
9      default:
10         int tmp;
11 }
```

- error: redefinition of 'tmp'
- 因為所有 case 目前屬於同一個 block，所以這樣寫是不好的。
- 盡量避免在 switch-case 裡面宣告變數。
- 不得已的時候，請加上大括號！

switch-case

```
1  switch(n) {
2      case 1:
3      case 2:
4      {
5          int tmp;
6      }
7          break;
8      case 3:
9      {
10         int tmp;
11     }
12         break;
13     default:
14     {
15         int tmp;
16     }
17 }
```

while, do-while

- do-while 非常少用到，但是有些時候它非常好用。
- 無論任何條件下，至少必須執行一次的 while 迴圈。
- 大家可以回想一下 Lab01 Practice B！

for

- 帶有計數器（counter）的迴圈。
- 使用時必須清楚了解 for 的執行順序！

for

```
1  for ( (1) ; (2) ; (3) )
2  {
3      (4)
4  }
```

1. 執行 (1) 進行初始化，在這裡可以宣告暫時變數。
2. 判斷 (2) 條件是否成立，成立則繼續，否則離開。
3. 執行 (4)。
4. 執行 (3)。
5. 回到第 2 點。

for

```
1  for(int i = 0; i < n; i++) {
2      cout << i << endl;
3  }
```

```
1  int i;
2  for(i = 0; i < n; i++) {
3      cout << i << endl;
4  }
```

Nested Structure

- 剛剛介紹的所有結構，都可以寫成巢狀結構。（好潮乙）
- 巢狀結構賦予程式碼更多彈性，可以組成更強、更簡潔的邏輯架構。

```
1  for(int i = 0; i < n; i++) {
2      for(int j = 0; j < m; j++) {
3
4      }
5  }
6
7  if(n < 100) {
8      if(n < 10) {
9
10     } else {
11
12     }
13 }
```

Practice

請以本週所學，實作上週的 Practice A、B、C、D。

Practice E

請用巢狀結構輸出九九乘法表，沒有格式限制。