1

上週隨堂練習

1. 請使用者輸入三個數字，之後由使用者**選擇**讓程式計算 1) 加總、 2) 平均、3) 找最大值、或4) 找最小值，並執行相對應之功能。~~請使用 switch(...) { case ...} 撰寫。~~

2

switch ... case

這是另外一組分支用的指令，通常比 if ... else if ... 更有效率

3

4-8.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";
```

```
  switch(age/6) {
    case 0:
      cout << "普通級";
      break;
    case 1:
      cout << "保護級";
      break;
    case 2:
      cout << "輔導級";
      break;
    default:
      cout << "限制級";
  }

  return 0;
}
```

在此範例裡共有四個分支

4

說明

- switch(...) { case ...} 為多重分支 (branching) 的另一種寫法，可用來取代if (...) {...} else if(...) {...} else (...) {...}。
- switch(A)，其中A
  - 需為整數(short, int, long) 或是字元型態 (char) 的**變數**。
  - 決定之後的那一個分支被執行到
- case B: ... break;
  - B 為前面前A之值的可能值
  - 每一個 case 被稱為一個分支 (branch)
  - 程式在執行時會跳躍到檢查 A 變數當時的值來決定那一個分支被執行。若所有的 case 指定之 B 都與 A 之值不符合時，則程式會跳躍至 default (內定的) 分支。
  - break 很重要！switch() 只檢查一次參數，如果沒有 break 的話所有在符合條件之後的動作都會被執行到！

5

4-9.cpp

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

int main() {
  char grade;

  cout << "請輸入成績 (A, B, C, D, F): ";
  cin >> grade;

  switch(grade) {
    case 'A':
      cout << "90 - 100";
      break;
```

6

4-9.cpp

```
    case 'B':
      cout << "80 - 89";
      break;
    case 'C':
      cout << "70 - 79";
      break;
    case 'D':
      cout << "60 - 69";
      break;
    default:
      cout << "0 - 59";
  }
  return 0;
}
```

7

玩玩看

- 如果把4-8.cpp 中的 break; 拿掉，程式是否可以執行？結果是否正確？
- 同樣的，如果把 4-9.cpp 中的 break; 拿掉，你是不是可以預測它的結果？

8

Fall through

- 在執行 switch (...) {case ...} 時，程式的分支判斷只執行一次，而一支分支的終點以 break; 決定之。

```
case 'B':
     cout << "80 - 89";
     break;
case 'C':
     cout << "70 - 79";
     break;
```

```
case 'B':
     cout << "80 - 89";
case 'C':
     cout << "70 - 79";
     break;
```

- 若一分支沒有 break 的話，接下來的分支亦會被執行，直到碰到 break 或是整個 switch 的區塊結束。
- switch case 中的 case 實質上只是一個標籤，而分支的動作是由 switch 執行的！！！

9

4-9a.cpp (改自 4-9.cpp)

```
switch(grade) {
  case 'A': case 'a':
    cout << "90 - 100";
    break;
  case 'B': case 'b':
    cout << "80 - 89";
    break;
  case 'C': case 'c':
    cout << "70 - 79";
    break;
```

```
  case 'D': case 'd':
    cout << "60 - 69";
    break;
  default:
    cout << "0 - 59";
}
```

10

if ... 和 switch ... 效率比較

- 在多重的 if ... else if ... 中，若判斷條件是簡單的 == 時，switch ... case ... 和 if ... else if ... 常常可以互相取代的。
- switch case 在編譯時，實際上是產生跳躍表 (jump table)，根據 switch 的值直接跳躍到對應的程式碼。
- if ... else if ... else if ... 則是在執行時期一個條件一個條件的檢查，所以執行上的效能會相對較差。

11

觀察

- 在引入 if() 和 switch() 之後, 程式的執行不再是由上而下的逐行執行了, 而可以根據程式執行當中所產生之資料而有不同的執行路徑.
- 從此, 電腦可以比一般計算機有更多的功能了!

12

Lecture 5

迴圈 – for, while, do ... while

13

今日內容

- for 迴圈
- break / continue

14

迴圈 (loop)

- 迴圈的用途
  - 用來重複執行某一段程式 → 執行**重複性**的事務
  - 配合陣列 (array) 以對大量資料進行處理 (later)
- C/C++ 的迴圈 (loop) 指令
  - for(...;...;...) {...} – 多應用於具固定次數的重複性
  - while() { ... } – 應用於重複次數不固定的場合
  - do {...} while(...); – 應用於重複次數不固定的場合

15

for 迴圈

16

5-1.cpp

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

int main() {
  int i;

  for(i=0; i<10; i++) {
      cout << "\n i=" << i;
  }

  return 0;
}
```

執行結果

```
i=0
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
```

17

for 的語法

- for (初始運算式；條件運算式；控制運算式) {
  欲重複執行的程式片斷;
  ...
  }
- for(i=0; i<10; i++) { cout << "\i=" << i; }
  - i = 0: 「先」設定 i 變數的值為 0
  - i < 10: 檢查迴圈的內容是否**要執行**或是否要**繼續執行**？
  - i++: 當迴圈內容作完一次後要執行的動作
  - i 為迴圈的控制變數，通常稱之為計數器 (counter)。
- 三個運算式都可以省略 → 無窮迴圈

18

5-2.cpp

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

int main() {
  int i;

  for(i=0; i<10; i=i+2) {
      cout << "\n i=" << i;
  }

  return 0;
}
```

i+=2

執行結果

```
i=0
i=2
i=4
i=6
i=8
```

19

5-3.cpp

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

int main() {
  int i;

  for(i=1; i<10; i=i+2) {
      cout << "\n i=" << i;
  }

  return 0;
}
```

執行結果

```
i=1
i=3
i=5
i=7
i=9
```

20

練習

- 請填入以下程式空白處，使程式結果輸出如下數列。

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

int main() {
  int i;

  for(i=_____; _____; _____) {
    cout << i << ", " ;
  }
  cout << endl;

  return 0;
}
```

10, 11, 12, 13, 14, 15, 16, 17, 18, 19,

10, 12, 14, 16, 18, 20, 22, 24, 26, 28,

5, 10, 15, 20, 25, 30, 35, 40, 45,

45, 40, 35, 30, 25, 20, 15, 10, 5,

-10, -7, -4, -1, 2, 5, 8,

1, 2, 4, 8, 16, 32, 64,

21

5-4.cpp

- 如何使用電腦幫我們計算從 1 加到 100 的總和？
  - 記得，如果你不會算，程式大概就寫不出來了 ...
  - 所以，你在寫程式的時候，一定要想想自己怎麼作一件事情 ...

22

5-4.cpp

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

int main() {
  int i, total=0;

  for(i=1; i<=100; i++) {
      total = total + i;
      cout << "\ni=" << i << ", 總和=" << total;
  }

  return 0;
}
```

```
i=1, 總和=1
i=2, 總和=3
      .
      .
      .
i=95, 總和=4560
i=96, 總和=4656
i=97, 總和=4753
i=98, 總和=4851
i=99, 總和=4950
i=100, 總和=5050
```

23

注意事項

- 寫迴圈時記得將迴圈的內容內縮 (indent), 用來便於識別那一部份是迴圈的內容!
  - 寫巢狀式的 if 或 switch-case 時 ，也應養成內縮的好習慣!!
- 如果沒有**很好的理由**, 不要在 for-loop 迴圈的區塊內改變迴圈計數器的值, 否則在除錯時會很困難...

24

for 的語法

- for (初始運算式；條件運算式；控制運算式) {
  欲重複執行的程式片斷;
  ...
  }
- 迴圈內的程式完成一次稱作一個**迭代** (iteration)
- for 迴圈又被稱作計數器控制迴圈 (counter-controlled loop)，通常都會利用一個或一個以上的變數作為「計數器」，用來計算並控制迴圈的迭代次數，並利用此計數器變化每次迭代的計算。
- 以上三個運算式，每個都可以有一個以上都運算，並以逗點隔開。其中條件運算部份不建議使用逗點，而應使用邏輯運算子複合你想要表達的迴圈執行條件。

```
for(i=0,sum=0; i<10; i=i+1,sum=sum+i) {
    cout << "\n i=" << i;
}
```

25

5-5.cpp

- 若從今天起，第一天存 1 元，每一天就存入前一天存入錢的兩倍 (e.g. 1, 2, 4, 8, 16, 32, …)，請問到第 30 天結束後，你總共存入了多少錢？

26

5-5.cpp

```
第 1 天: 存入 1 元, 共有 1 元
第 2 天: 存入 2 元, 共有 3 元
第 3 天: 存入 4 元, 共有 7 元
第 4 天: 存入 8 元, 共有 15 元
第 5 天: 存入 16 元, 共有 31 元
第 6 天: 存入 32 元, 共有 63 元
第 7 天: 存入 64 元, 共有 127 元
第 8 天: 存入 128 元, 共有 255 元
第 9 天: 存入 256 元, 共有 511 元.
.
.
第 28 天: 存入 134217728 元, 共有 268435455 元
第 29 天: 存入 268435456 元, 共有 536870911 元
第 30 天: 存入 536870912 元, 共有 1073741823 元
```

27

5-5.cpp

如果現在想要知道 60 天後的結果呢？

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

int main() {
	int saving = 1;
	int total = 0;

	for(int i=1;i<=30;i++) {
		total = total + saving;
		cout << "第 " << i << " 天: ";
		cout << "存入 " << saving << " 元, ";
		cout << "共有 " << total << " 元" << endl;
		saving = saving * 2;
	}	return 0;
}
```

28

5-6.cpp

- 請列出 1 – 50 之間，可以被 3 或可以被 5 整除的所有數字。

29

5-6.cpp

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

int main() {
	for(int i=1;i<=50;i++) {
		if(i%3 == 0 || i%5 == 0) {
			cout << i << ", ";
		}
	}
	cout << endl;

	return 0;
}
```

30

break / continue

31

break / continue

- break;
  - To pre-terminate a loop
  - 用來提早結束一個迴圈的執行。
  - 跳脫目前所在的區塊、從目前所在區塊外下面一行敍述繼續執行。
- continue;
  - To pre-terminate an iteration
  - 用來提早結束一次迭代的執行。
- 可使用在 for、do/while、while三種迴圈中

32

5-7.cpp & 5-8.cpp

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

int main() {
  int i;

  for(i=0;i<15;i++) {
    if(i>=5 && i<10) break;
    cout << "i=" << i << endl;
  }
  cout << "i-迴圈結束" << endl;

  return 0;
}
```

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

int main() {
  int i;

  for(i=0;i<15;i++) {
    if(i>=5 && i<10) continue;
    cout << "i=" << i << endl;
  }
  cout << "i-迴圈結束" << endl;

  return 0;
}
```

33

5-7 & 5-8 執行結果

```
i=0
i=1
i=2
i=3
i=4
i-迴圈結束
```

```
  for(i=0;i<15;i++) {
    if(i>=5 && i<10) break;
    cout << "i=" << i << endl;
  }
  cout << "i-迴圈結束" << endl;
```

```
i=0
i=1
i=2
i=3
i=4
i=10
i=11
i=12
i=13
i=14
i-迴圈結束
```

34

隨堂練習

35

1. 列出所有因數 (factors)

- 請使用者輸入一正整數 x，接著由你的程式列出該正整數的所有因數 (可以整除 x 的數字)。

```
請輸入一正整數: 24
1, 2, 3, 4, 6, 8, 12, 24,
```