1

Lecture 5

for 迴圈
break / continue
while 迴圈
do ... while 迴圈

2

今日內容

- for 迴圈
- break / continue
- while 迴圈
- do ... while 迴圈

3

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
  - 之後會介紹到配合陣列 (array) 以對大量資料進行處理 (其實，就是重複性的計算)
- 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

4

for 迴圈

5

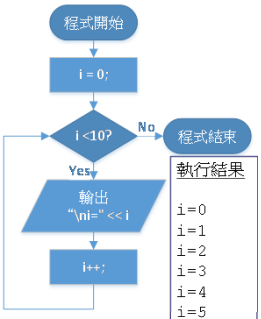
5-1.cpp

```
#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
```

6

for 的語法

- for (初始運算式；條件運算式；控制運算式) {
  迴圈內容或欲重複執行的程式片段;
  ...
  }
- 初始運算式：迴圈內容開始執行前執行的敘述
- 條件運算式：條件運算式在 1) 迴圈執行前或是 2) 迴圈內容完成後執行，若運算結果為真，才會執行或是再執行迴圈的內容
- 控制運算式：迴圈內容完成一次後執行的敘述
- 以上三個運算式都可以省略 → 無窮迴圈，迴圈內容會不斷地重複執行

7

for 的語法

- 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)。

8

for 迴圈

```
    1   2   3
for (i=0; i<3; i++) {
4       cout << "a";
}

5 cout << "b";
```

1 迴圈被執行到時初始化的動作
2 迴圈要執行或要被繼續執行的條件
3 迴圈完成一次後要做的動作
4 迴圈內欲重複執行的敘述
5 迴圈結束後要執行的敘述

1	令 i = 0
2	0 < 3 → *true*
4	cout << "a";
3	i++;
2	1 < 3 → *true*
4	cout << "a";
3	i++;
2	2 < 3 → *true*
4	cout << "a";
3	i++;
2	3 < 3 → ***false***
5	cout << "b";

9

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
```

10

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
```

11

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,

12

5-4.cpp

- 如何使用電腦幫我們計算從 1 加到 100 的總和？
  - 這裡，我們要先教電腦笨的方法 (聰明的方法是什麼？)
  - 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，而且要把步驟拆得很細，因為電腦很笨 ...

13

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
```

14

注意事項

- 寫迴圈時記得將迴圈的內容縮排 (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 ，也應養成縮排的好習慣!!
- 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...

15

for 的語法

- 迴圈的內容執行完一次，回到控制運算式，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。有時，會利用此計數器變化每次迭代所作的計算。

  `for(i=0;i<123;i++) { … }`
- 三個運算式中初始運算式 (e.g. i=0) 與控制運算式 (e.g. i++) 可以有一個以上的運算，並以逗點隔開。其中條件運算部份不宜使用逗點，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;
}
}
}
```

16

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？

17

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
```

18

5-5.cpp

```
#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;
}
```

```
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
```

如果現在想要知道 60 天後的結果呢？

19

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。

20

5-6.cpp

- 變數在使用前宣告即可。
- 此處的變數 *i* 僅在 for 迴圈內有被定義，以此範例中，*i* 變數只有在緊接的 {…} 內看得到，到了最後的 cout 處其實 *i* 變數沒有被定義。

```
#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;
}
```

21

break / continue

22

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

23

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;
}
```

24

5-7 & 5-8 執行結果

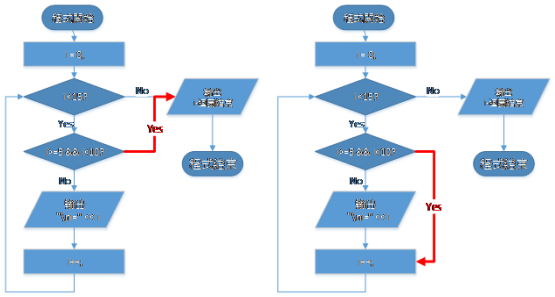
```
i=0
i=1
i=2
i=3
i=4
i-迴圈結束
```

```
  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
```

25

5-7.cpp & 5-8.cpp 的流程圖比較



28

while 迴圈

```
int a = 0;
        1
while( a < 3) {
        cout << “a”;
        a++;
}
3 cout << “b”;
```

2

1 判斷迴圈內容是否執行或繼續執行

2 迴圈內欲重複執行的敘述

3 迴圈結束後要執行的敘述

1	0 < 3 → *true*
2	cout << “a”;
2	a++;
1	1 < 3 → *true*
2	cout << “a”;
2	a++;
1	2 < 3 → *true*
2	cout << “a”;
2	a++;
1	3 < 3 → ***false***
3	cout << “b”;

26

while 迴圈

29

5-9.cpp

```
#include <iostream>
using namespace std;
int main() {
  int count, sum=0;

  cout << "請輸入一個整數: ";
  cin >> count;
  while(count>0) {
      sum += count;
      cout<< "count=" << count << ", sum=" << sum << endl;
      count--;
  }

  return 0;
}
```

```
請輸入一個整數: 10
count=10, sum=10
count=9, sum=19
count=8, sum=27
count=7, sum=34
count=6, sum=40
count=5, sum=45
count=4, sum=49
count=3, sum=52
count=2, sum=54
count=1, sum=55
```

27

while 的語法

- while (條件運算式) {
  欲重複執行的程式片段;
  …
  }
- 其中條件運算式 / 布林運算式的部份與 if 的寫法一樣
  - count>0
  - a<10 && b<10
  - …
- while 和 for 迴圈一樣，都會**先檢查**條件運算式的結果決定迴圈的內容是否會被執行，所以**有可能一次都不會被執行**。

30

5-10.cpp

```
#include <iostream>
using namespace std;
int main() {
  int ans;
  bool isOdd = true;

  while(isOdd) {
      cout << "請輸入一個偶數: ";
      cin >> ans;

      isOdd = (ans % 2 == 1);
  }
  cout << "你輸入了一個偶數: " << ans << endl;

  return 0;
}
```

```
請輸入一個偶數: 1
請輸入一個偶數: 3
請輸入一個偶數: 5
請輸入一個偶數: 7
請輸入一個偶數: 6
你輸入了一個偶數: 6
```

31

5-11.cpp

- 費氏級數 (Fibonacci number) 之定義如下:
  - 1, 1, 2, 3, 5, 8, 13, 21, ...
  - 級數中每個數字皆為級數中的前兩個數字加總
- 如何寫一個程式依序列出級數中小於 150 的各項?
  1. 我們自己會怎麼作?
  2. 過程當中我們需要同時記得幾個資料?

32

5-11.cpp

```
#include <iostream>
using namespace std;
int main() {
  int a=1, b=1, c=a+b;

  cout << a << ", " << b << ", ";
  while(c<=150) {
      a=b;
      b=c;
      cout << b << ", ";
      c=a+b;
  }

  return 0;
}
```

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,

33

do/while 迴圈

34

do/while 迴圈

```
int a = 0;

do {
      cout << "a";
      a++;
} while(a<3);
cout << "b";
```

1 迴圈內欲重複執行的敘述
2 迴圈要執行或要被繼續執行的條件
3 迴圈結束後要執行的敘述

1	cout << "a";
1	a++;
2	1 < 3 → *true*
1	cout << "a";
1	a++;
2	2 < 3 → *true*
1	cout << "a";
1	a++;
2	3 < 3 → ***false***
3	cout << "b";

35

do/while 的語法

- do {
  欲重複執行的程式片段;
  ...
  } while (布林運算式); ← 很重要
- 其中布林運算式的部份與 if 的寫法一樣
  - count>0
  - a<10 && b<10
  - ...
- do/while 迴圈與 while 或 for 最大不一樣的地方在於它的條件判斷在迴圈內容之後，亦即它的迴圈內容**被保證至少執行到一次**。
- 記得 while (...) 之後要加分號！

36

5-12.cpp

```
#include <iostream>
using namespace std;
int main() {
  char again;
  float weight, height, BMI;
  do {
      cout << "請輸入你的身高 (公分): ";
      cin >> height;
      cout << "請輸入你的體重 (公斤): ";
      cin >> weight;
      BMI = weight*10000 / height / height;
      cout << "你的 BMI 值為: " << BMI << endl;
      cout << "再算一次請按 y: ";
      cin >> again;
  } while(again=='y' || again=='Y');
  return 0;
}
```

```
請輸入你的身高 (公分): 170
請輸入你的體重 (公斤): 60
你的 BMI 值為: 20.7612
再算一次請按 y: y
請輸入你的身高 (公分): 180
請輸入你的體重 (公斤): 60
你的 BMI 值為: 18.5185
再算一次請按 y: y
請輸入你的身高 (公分): 172
請輸入你的體重 (公斤): 55
你的 BMI 值為: 18.5911
再算一次請按 y: n
```

5-13.cpp

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

  do {
      cout << "請輸入一個偶數: ";
      cin >> ans;
  } while(ans%2 != 0);

  cout << "你輸入了一個偶數" << endl;

  return 0;
}
```

```
請輸入一個偶數: 1
請輸入一個偶數: 3
請輸入一個偶數: 5
請輸入一個偶數: 7
請輸入一個偶數: 6
你輸入了一個偶數
```


小結

- while
- do ... while
- 注意, 之前介紹過的 break 與 continue 一樣也可以使用在 while / do-while 迴圈內

隨堂練習

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,
```

HW04

Due: 4/17/2015

1. 花錢要有節制

請撰寫一程式，輸入某人的月薪、第一個月的花費、以及之後每個月花費的成長倍數。輸入完成後，輸出每個月的花費、每個月的剩餘金額直到餘額為0為止。其中必需注意到最後一個月的花費不能使餘額為負值。

```
請輸入月薪：40000
請輸入第一個月的花費：25000
請輸入每固月花費的成長倍數 (需大於1)：1.2

第 1 個月，花費 25000 元，剩 15000元。
第 2 個月，花費 30000 元，剩 25000元。
第 3 個月，花費 36000 元，剩 29000元。
第 4 個月，花費 43200 元，剩 25800元。
第 5 個月，花費 51840 元，剩 13960元。
第 6 個月，花費 53960 元，剩 0元。
```

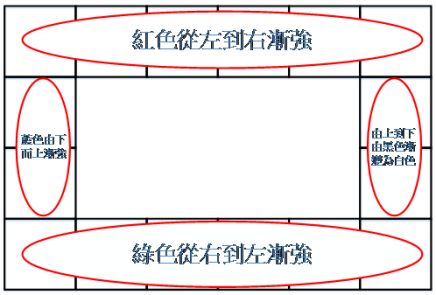
請輸入月薪：100000
請輸入第一個月的花費：30000
請輸入每固月花費的成長倍數(需大於1)：0.5
請輸入每固月花費的成長倍數(需大於1)：0.75
請輸入每固月花費的成長倍數(需大於1)：1.2

第 1 個月，花費 30000 元，剩 70000元。
第 2 個月，花費 36000 元，剩 134000元。
第 3 個月，花費 43200 元，剩 190800元。
第 4 個月，花費 51840 元，剩 238960元。
第 5 個月，花費 62208 元，剩 276752元。
第 6 個月，花費 74649 元，剩 302103元。
第 7 個月，花費 89578 元，剩 312525元。
第 8 個月，花費 107493 元，剩 305032元。
第 9 個月，花費 128991 元，剩 276041元。
第 10 個月，花費 154789 元，剩 221252元
第 11 個月，花費 185746 元，剩 135506元
第 12 個月，花費 222895 元，剩 12611元。
第 13 個月，花費 112611 元，剩 0元。

請輸入月薪：25000
請輸入第一個月的花費：12000
請輸入每固月花費的成長倍數 (需大於1)：1.1

第 1 個月，花費 12000 元，剩 13000元。
第 2 個月，花費 13200 元，剩 24800元。
第 3 個月，花費 14520 元，剩 35280元。
第 4 個月，花費 15972 元，剩 44308元。
第 5 個月，花費 17569 元，剩 51739元。
第 6 個月，花費 19325 元，剩 57414元。
第 7 個月，花費 21257 元，剩 61157元。
第 8 個月，花費 23382 元，剩 62775元。
第 9 個月，花費 25720 元，剩 62055元。
第 10 個月，花費 28292 元，剩 58763元。
第 11 個月，花費 31121 元，剩 52642元。
第 12 個月，花費 34233 元，剩 43409元。
第 13 個月，花費 37656 元，剩 30753元。
第 14 個月，花費 41421 元，剩 14332元。
第 15 個月，花費 39332 元，剩 0元。

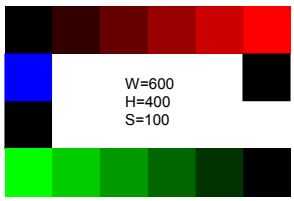
填色的規則

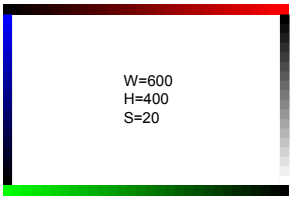


強度最弱為0、最強為255

2. 繪圖

- 請撰寫一程式，讓使用者輸入寬度(W)、高度(H)、以及邊長(S)等三個資料，之後根據使用者所輸入的資料產生如下的圖形：





矩形大小的規則