

回顧

- `if(...) {...} else {...}`
- `if(...) {...}`
- `if(...) {...} else if(...) {...} else if(...) {...} else {...}`

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

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

```
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";  
}
```

```
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";  
}
```

```
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";  
}
```

```
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";  
}
```

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。

Lecture 5

for 迴圈

今日內容

- for 迴圈
- break / continue

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
- 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

for 迴圈

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
```

for 的語法

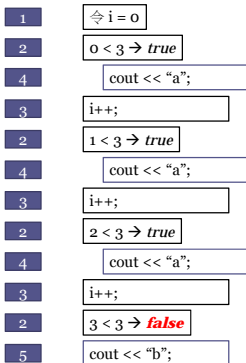
- for (初始運算式; 條件運算式; 控制運算式) {
 - 欲重複執行的程式片斷;
 - ...
- for(i=0; i<10; i++) { cout << "i=" << i; }
 - i = 0: 「先」設定 i 變數的值为 0
 - i < 10: 檢查迴圈的內容是否**要執行**或是否要**繼續執行**?
 - i++: 當迴圈內容作完一次後要執行的動作
 - i 為迴圈的控制變數, 通常稱之為計數器 (counter)。
- 三個運算式都可以省略 → 無窮迴圈

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



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;
}
```

執行結果

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

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
```

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,

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 ...

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
```

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...

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。
- 三個運算式中初始運算式與控制運算式，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建議使用短點，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;
}
}
```

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？

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

```

5-5.cpp

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

如果現在想要知道 60 天後的結果呢？

```

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;
}

```

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。

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;
}

```

break / continue

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敘述繼續執行。
- **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

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;
}
```

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
```

隨堂練習

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,
```

2. 推土機畫圖

- 請下載 <http://140.118.105.174/Courses/CVB/2012/Lab/Labo2.zip> 並將此壓縮檔解開至桌面上。
 - 請利用迴圈並配合以下指令來在棋盤上畫一口字。
 - 範例請參考解開之後的 Labo2.exe.
- ```
car.forward();
car.turnLeft();
car.turnRight();
car.paint();
```

