英哩數轉換公里

丁培毅

實習目標：

1. 瞭解資料變數、變數型態、變數命名
2. 瞭解基本數值運算方法
3. 鍵盤輸入與螢幕輸出、引入檔案
4. 程式編輯、修改、編譯、與執行
5. 模仿範例的基本學習方法
6. 細心觀察編譯訊息與執行結果以逐步修改成為所需要的程式

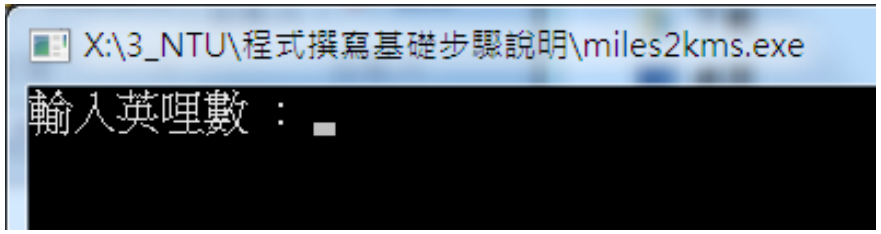
英哩數轉換公里

請撰寫一個程式

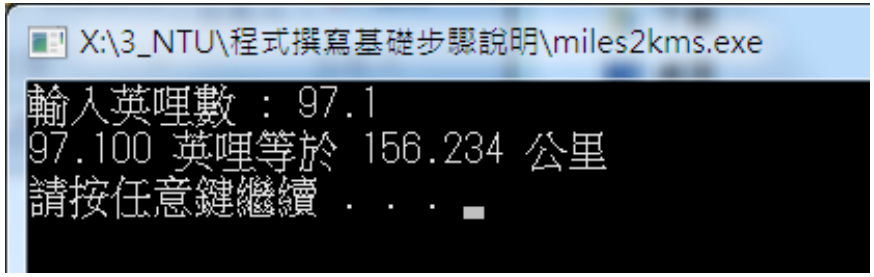
- 要求使用者由鍵盤鍵入英哩數 (例如MLB投手投出球速97.1英哩)
- 計算並且在螢幕上印出對應的公里數 (156.234公里)

要求的程式外觀

- 寫好的程式一執行起來會看到如下的畫面，提示執行程式的人「輸入英哩數：」



- 輸入 97.1 以後，程式計算出來等於 156.234 公里，並且印出「 97.100 英哩等於 156.234 公里」



1. 你把 “執行你寫的程式的電腦” 看成是一個運算速度很快的人，你由鍵盤敲進去一個數字 (例如 97.1) 代表你希望它幫你轉換的英哩數，然後電腦執行你寫的程式，最後在螢幕上顯示另外一個數字 (例如 156.234) 代表對應的公里數
2. 這個過程裡電腦只是依照你給它的命令運作而已，你不給它指令，它就不知道該做什麼，撰寫程式的過程就是去組織電腦能夠執行的動作，完成你希望它做的事情
3. 我們知道它可以在變數裡紀錄數值資料：
   例如：int x; 或是 double y;
   x = 123; y = 45.67;
   在這裡我們設計一個 double 型態的變數 miles 來紀錄有小數點的英哩數實數值 (名稱 miles 是你自己可以決定的, 不要用 x, y, z, a, b, c, a1, a2，要用你打算存放的資料的意義來命名, 以後你再看到這個變數名稱就知道裡面該放些什麼)
   double miles;
   如果我們紀錄 97.1 在這個變數裡，例如
   miles = 97.1;

4. 我們知道它可以執行加減乘除的運算
   例如: x * y, x + 2, (3.1 * 5.5) / (x - 6.6)
   在這個應用程式裡，可以用乘法運算，很快地算出英哩轉換成公里的結果，我們再定義一個 double 型態的變數 kilometers 來存放計算的結果

```
double kilometers;
kilometers = miles * 1.609;
```

5. 因爲這個題目的要求不多，這樣就完成了 “核心程式功能” 的設計，你可以嘗試編譯，確定語法是正確的。不過這樣還沒有完成整個程式，還需要處理 “鍵盤輸入” 以及 “螢幕輸出” 的部份；這兩個部份我們需要運用 C 語言的標準輸入輸出函式庫 (stdio) 來設計，你需要參考使用範例，調整一下來完成你需要的功能
6. 鍵盤讀取倍精準浮點數 (double)：你可能會從書裡面看到使用範例: #include <stdio.h>

```
double x;
scanf("%lf", &x);
```

6. 配合上面的核心程式，我們希望由鍵盤讀到的英哩數值可以存放在 double 型態的變數 miles 裡，所以我們改成

```
scanf("%lf", &miles);
```

7. 題目要求需要在螢幕上輸出下面藍色文字及倍精準浮點數

輸入英哩數：97.1

97.100 英哩等於 156.234 公里

你也許有看過 stdio 函式庫的使用範例：

```
printf("hello world\n");
double x = 1.234;
printf("%f", x);
```

配合我們的題目要求修改爲

```
printf("輸入英哩數：");
printf("%f 英哩等於 %f 公里", miles, kilometers);
```

印出來的結果是:

97.100000 英哩等於 156.233900 公里

只差小數點後面位數不對了

8. 再參考 stdio 裡在螢幕上印出浮點數時的格式範例:

```
double x=1234.5638; printf("%8.3f", x);
```

會在螢幕上印出 1234.564，8.3 代表包含小數點至少 8 位 (如果是 %.3f 的話就是不限定總共有幾位)、.3 代表小數點後有 3 位 (第四位數字四捨五入)，所以我們可以修改爲：

```
printf("%.3f 英哩等於 %.3f 公里", miles, kilometers);
```

9. 到這個階段基本上已經完成一個可以執行的程式了，需要測試其功能是否符合需求

10. 你給電腦的指令只要符合基本的格式要求，電腦就會遵照命令來運作，但是這些指令如果沒有辦法達成目標，電腦還是死板地依循指令來運作，不會去更改你給的指令，所以任何時候程式沒有達成你想像的目標時，大概都不是電腦的錯，應該是指令給錯了，或是指令的順序給錯了，這時候你需要

    a. 仔細檢查你給它的指令 (就是你寫的 C 程式)，用紙筆模擬執行一下，看看這樣的指令究竟電腦執行的過程中會產生什麼結果，需要怎樣修改程式才能得到要的結果

b. 有的時候製作的程式比較複雜，用眼睛看常常會發生失誤，這時候我們也可以修改程式，增加一些 printf 敘述，把執行的過程在螢幕上顯示出來，如此可以很清楚看出來什麼地方和自己想像的有所出入，進而可以分析該如何修改程式

以後會帶大家看原始碼偵錯工具，也可以輔助讓你看出來目前的程式到底在做些什麼，需要特別注意的是一定要完全知道自己寫的程式在做些什麼以後才開始修改，千萬不要盲目地修改，想說碰運氣也許會改對，也許會吧，但是就算對了，你還是不知道原先的寫法是哪裡錯了，通常就是一些語法上或是運算模型上認知的錯誤，本來可以就此校正過來的，因爲你亂改又隱藏起來了，等待下次寫程式時這些錯誤的觀念一定會再出來搗亂，程式越寫功能越來越複雜，就會一次一次越來越痛苦的

欲速則不達

- 在這一份投影片裡，我們看到要一步一步地完成一個程式其實步驟相當多，該注意的事情也相當多
- 如果你曾經接觸過程式設計，也許覺得好像先前沒有要注意那麼多，不是可以快一點完成的嗎? 很多時候一心想要看到最後完成的結果，卻發現越急躁越是遇見比別人更多的錯誤，該檢查、該思考的點都快速地跳過以後，常常會在遇到問題時，發現問題的來源是多重混雜而很難確認的
- 前面看到的事項其實都還蠻關鍵的，很多時候就是因爲開始練習寫程式的時候沒有注意，養成了一些不好的習慣，習慣跳過一些步驟，導致錯誤推論，或是不曉得原來需要一步一步去掌握你寫的程式所完成的事項，常常倚賴 callout, 想要找到大神幫你一大步解決程式設計時遇見的問題，造成以後程式規模越來越大、越來越複雜時，完全無法掌控程式的表現，如此不需要太久，你一定會作出『自己不太適合從事軟體設計這種工作』的結論，這樣會不會有點可惜?