

# 程式設計 (113-1)

## 作業七

作業設計：孔令傑  
國立臺灣大學資訊管理學系

繳交作業時，請至 PDOGS (<http://pdogs.ntu.im/>) 為第一、二、三題上傳一份 C++ 原始碼 (以複製貼上原始碼的方式上傳)，並且到 NTU COOL 上傳一份 PDF 檔。每位學生都要上傳自己寫的解答。不接受紙本繳交；不接受遲交。這份作業的程式碼和書面報告的截止時間都是 **11 月 5 日早上八點**。為這份作業設計測試資料並且提供解答的助教是宋凱翔。

在你開始前，請閱讀課本的第 22.1–22.6 節 (關於 self-defined data types in C) 和第 9、10 章 (class) <sup>1</sup>。

本次作業滿分為 110 分，得幾分就算幾分。若整學期有  $n$  份作業，則學期的作業總成績即為  $n$  份作業的總分除以  $n$  (不論超過 100 與否)。

### 第一題

(20 分) 本題內容大致上與作業四的第一題一樣，但我們會用這題讓大家練習課堂上新教的「類別」。為了讓大家不用回頭看題目，底下我們重新敘述這個題目一次。

奧運會上各國之間會有獎牌排名，雖然有各種排名方式，但最常見的是「先比金牌數，多者排前；如果金牌數一樣，則比銀牌數，多者排前；如果銀牌數一樣，則比銅牌數，多者排前」。已知在某一天賽事開始前，沒有任兩個國家的三種獎牌數都一樣，且照此規則排序後， $n$  個國家中排名第  $i$  的國家的金、銀、銅牌數各是  $g_i$ 、 $s_i$  和  $b_i$ 。若在該天賽事中排名第  $k$  的國家多獲得了  $G$ 、 $S$ 、 $B$  面金、銀、銅牌，且其它國家的獎牌數都不變，我們想知道這個有獲得獎牌的國家的排名會上升到第幾名。如果這個國家獲得獎牌後的各種獎牌數剛好跟另一個國家的一模一樣，規定上是剛獲得獎牌的這個國家要被排名在後。

舉例來說，如果原本有  $n = 5$  個國家，每個國家的排名和獎牌數如表 1 所示。則若原本第  $k = 3$  名的國家多得到  $G = 2$  面金牌、 $S = 0$  面銀牌、 $B = 0$  面銅牌，它的排名將升到第二名；若原本第  $k = 3$  名的國家多得到  $G = 2$  面金牌、 $S = 4$  面銀牌、 $B = 0$  面銅牌，它的排名將升到第一名；若原本第  $k = 4$  名的國家多得到  $G = 0$  面金牌、 $S = 5$  面銀牌、 $B = 3$  面銅牌，它的排名將維持在第四名。

排名	金牌數	銀牌數	銅牌數
1	10	6	1
2	10	2	8
3	8	2	9
4	7	5	4
5	7	5	3

表 1: 獎牌排名

<sup>1</sup>課本是 Deitel and Deitel 著的 *C++ How to Program: Late Objects Version* 第七版。

在本題中，給定以上資訊，我們要請你求出有獲得獎牌之國家的新排名。為了讓大家練習「類別」，在本題中我們在 PDOGS 上定義了一個類別 Country：

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

class Country
{
private:
    int gold;    // number of gold medals
    int silver;  // number of silver medals
    int bronze;  // number of bronze medals
public:

    // constructor
    // g: gold medals, s: silver medals, b: bronze medals
    Country(int g, int s, int b);

    // return true is the calling object should be ranked
    // higher than "another" or false otherwise
    bool isRankedHigherThan(const Country& another) const;

    // increment the member variables according to the parameters
    // g: gold medals, s: silver medals, b: bronze medals
    void incrementMedals(int g, int s, int b);
};

// Please implement these member functions by yourselves
```

其中 Country 是個 constructor，會將新建立的物件的三個成員變數的值設定成傳入的三個引數值；isRankedHigherThan 函數會根據兩個國家的金、銀、銅牌的數量進行比較，如果呼叫這個函數的物件應該排在傳入的物件 another 之前則回傳 true，反之回傳 false；incrementMedals 則會讓呼叫函數的物件的獎牌數照傳入的引數值增加。

我們也在 PDOGS 上定義了 main function，基本上就是讓大家只要把前述的三個成員函數都實作出來，就能夠完成這一題。main function 以及前述的類別定義都在附件的 PD113-1\_hw7\_p1.cpp 裡，大家可以自行參考。PD113-1\_hw7\_p1.cpp 裡面的內容就是我們設置在 PDOGS 上的內容，所以大家在本題應該只上傳大家定義的成員函數的內容。請注意大家也不能自行定義更多成員函數或變數。

## 輸入輸出格式

系統會提供一共 10 組測試資料（雖然作業四的第一題也是 10 組測試資料，但本題的 10 組和當時的 10 組內容是不同的），每組測試資料裝在一個檔案裡。在每個檔案中會有  $n + 1$  列，第一列裝著五個正整數，依序是  $n$ 、 $k$ 、 $G$ 、 $S$ 、 $B$ ；在從第二列到第  $n + 1$  列中，第  $i + 1$  列存了三個非負整數，依序是  $g_i$ 、

$s_i$  和  $b_i$ 。已知  $1 \leq n \leq 100$ ，其它所有整數都介於 0 和 50（包含 0 和 50），且不存在不同的  $i$  和  $j$  同時滿足  $g_i = g_j$ 、 $s_i = s_j$  和  $b_i = b_j$ 。每一列的任兩個相鄰的整數間以一個空白字元隔開。

讀入這些資訊後，請輸出原本排名第  $k$  的國家獲得更多獎牌後的新排名，若新排名為第  $r$  名則輸出  $r$ 。舉例來說，如果輸入是

```
6 4 1 2 1
12 8 3
10 7 5
8 6 4
7 5 2
5 5 5
3 2 1
```

則輸出應該是

```
3
```

如果輸入是

```
6 4 1 1 2
12 8 3
10 7 5
8 6 4
7 5 2
5 5 5
3 2 1
```

則輸出應該是

```
4
```

如果輸入是

```
4 2 2 2 1
5 3 2
4 2 1
3 2 3
1 4 5
```

則輸出應該是

```
1
```

## 你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你可以使用任何方法。

## 評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。

## 第二題

(20 分) 有一家有恰好一個醫師的診所，每天都有許多患者前來看診，而每位患者都在不同的時間抵達診所。患者們在到達診所後，都依照抵達的時間先後排隊等待看診，先到的就先接受醫師看診。我們稱呼第  $i$  個抵達診所的患者為患者  $i$ ，而患者  $i$  身上有兩個屬性，分別是他到達診所的時間  $a_i$ ，以及醫生為他看診所耗費的時間長  $s_i$ ，這些數字都是以分鐘為單位的整數，其中  $a_i$  表示患者  $i$  是在診所開門後的第  $a_i$  分鐘抵達診所的。表 2 是一個範例，這天有六個患者前來，第一位在開診後第 6 分鐘抵達，醫師花了 5 分鐘幫他看診，依此類推。我們用  $n$  表示總共的患者人數，例如表 2 顯示共有  $n = 6$  位患者。

項目	抵達順序 $i$					
	1	2	3	4	5	6
到達時間 $a_i$	6	11	18	19	27	38
看診時間長 $s_i$	5	12	4	6	2	3

表 2: 患者資訊

根據這些資訊，我們可以推算每個患者被醫師開始看診的時間，進而幫每個患者計算他的看診完成時間  $f_i$ 、從抵達診所到接受看診的等待時間長  $w_i$ ，以及他到達診所的前一瞬間診所中的總人數  $p_i$ （因為是前一瞬間，所以不包含他自己；診所中的人數包含被看診中的以及等待中的）。舉例來說，如果診所當日的患者資訊如表 2 所示，則我們將依序為每位患者計算如下：

- 第一位患者到達診所時間  $a_1$  為第 6 分鐘，看診時間長  $s_1$  為 5 分鐘。由於他是第一個到的，接受看診的開始時間就是他抵達的時間（第 6 分鐘），因此他的看診完成時間  $f_1 = 6 + 5 = 11$ 、等待時間長  $w_1 = 0$ 。因為這位患者到達的前一瞬間診所中並沒有人在看診或排隊，因此  $p_1 = 0$ 。
- 第二位患者到達診所的時間  $a_2$  為第 11 分鐘，看診時間長  $s_2$  為 12 分鐘。他到達時患者 1 恰恰好離開，因此他也不用排隊，可以直接在他抵達的時間（第 11 分鐘）接受看診，因此他的看診完成時間  $f_2 = 11 + 12 = 23$ 、等待時間長  $w_2 = 0$ 。因為這位患者到達的「前一瞬間」診所中有一個人正在看診，因此  $p_2 = 1$ 。
- 第三位患者到達診所的時間  $a_3$  為第 18 分鐘，看診時間長  $s_3$  為 4 分鐘。他到達時患者 2 還在接受看診，因此他需要排隊，要等到患者 2 離開的時間（第 23 分鐘）才接受看診，因此他的看診完成時間  $f_3 = 23 + 4 = 27$ 、等待時間長  $w_3 = 23 - 18 = 5$ 。因為這位患者到達的前一瞬間診所中有人在看診但沒人在排隊，因此  $p_3 = 1$ 。
- 第四位患者到達診所的時間  $a_4$  為第 19 分鐘，看診時間長  $s_4$  為 6 分鐘。他到達時患者 2 還在接受看診，且患者 3 還在排隊，因此他需要排隊，要等到患者 3 離開的時間（第 27 分鐘）才接受看診，因此他的看診完成時間  $f_4 = 27 + 6 = 33$ 、等待時間長  $w_4 = 27 - 19 = 8$ 。因為這位患者到達的前一瞬間診所中有人在看診且有一人在排隊，因此  $p_4 = 2$ 。

- 第五位患者到達診所的時間  $a_5$  為第 27 分鐘，看診時間長  $s_5$  為 2 分鐘。他到達時患者 3 剛好結束看診，且患者 4 正要接受看診，因此他需要排隊，要等到患者 4 離開的時間（第 33 分鐘）才接受看診，因此他的看診完成時間  $f_5 = 33 + 2 = 35$ 、等待時間長  $w_5 = 33 - 27 = 6$ 。請大家特別注意，這位患者到達的「前一瞬間」診所中有兩個人（患者 3 正在接受看診、患者 4 還在排隊），因此  $p_5 = 2$ 。
- 第六位患者到達診所的時間  $a_6$  為第 38 分鐘，看診時間長  $s_6$  為 3 分鐘。他到達時患者 5 已經離開了，因此他不用排隊，可以直接在他抵達的時間（第 38 分鐘）接受看診，因此他的看診完成時間  $f_6 = 38 + 3 = 41$ 、等待時間長  $w_6 = 0$ 。因為這位患者到達的前一瞬間診所中沒有人在看診或排隊，因此  $p_6 = 0$ 。

前述推算的所有結果可以被統整如表 3 所示。

項目	抵達順序 $i$					
	1	2	3	4	5	6
看診完成時間 $f_i$	11	23	27	33	35	41
等待時間長 $w_i$	0	0	5	8	6	0
患者 $i$ 到達前的診所內人數 $p_i$	0	1	1	2	2	0

表 3: 推算結果

診所希望在任意指定一個序號  $k$  之後，你可以幫他們找出患者  $k$  的看診完成時間  $f_k$ 、從抵達診所到接受看診的等待時間長  $w_k$ ，以及他到達診所的前一瞬間診所中的總人數  $p_k$ 。診所也希望在給定時間點  $t$  後，你可以幫他們算出在第  $t$  分鐘那個瞬間的前一瞬間的診所中的總人數  $x(t)$ ；由於是第  $t$  分鐘的前一瞬間，所以如果有患者在第  $t$  分鐘到達，這位患者不算在  $x(t)$  內。舉例來說，假設  $k = 3$ ，則我們找出患者 3 的看診完成時間  $f_3 = 27$ 、等待時間長  $w_3 = 5$ ，以及其到達前診所中的人數  $p_3 = 1$ 。若給定的時間點為  $t = 16$  當下，則  $x(16) = 1$ （其中有 0 個人在排隊，1 個人在看診）。若  $t = 19$ ，則  $x(19) = 2$ （其中有 1 個人在排隊，1 個人在看診），請注意  $x(19)$  不包括在第 19 分鐘到達的患者。

## 輸入輸出格式

系統會提供一共 10 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有四列，其中第一列裝著一個  $n$  正整數；在第二列裝著  $n$  個正整數，依序是  $a_1$ 、 $a_2$ 、 $a_3$  直到  $a_n$ ；第三列裝著  $n$  個正整數，依序是  $s_1$ 、 $s_2$ 、 $s_3$  直到  $s_n$ ；第四列裝著一個正整數  $k$ 、一個逗點、一個正整數  $t$ 。已知  $1 \leq n \leq 100$ 、 $1 \leq a_1 < a_2 < a_3 < \dots < a_n \leq 100000$ 、 $1 \leq s_i \leq 1000$ 、 $k \in \{1, 2, \dots, n\}$ 、 $1 \leq t \leq 10000$ 。

讀入這些資訊後，請依照題目指定的規則計算  $f_k$ 、 $w_k$ 、 $p_k$  和  $x(t)$ ，並將這些數字依序印出，相鄰的兩個數字中間用一個逗點隔開。舉例來說，如果輸入是

```
6
6 11 18 19 27 38
5 12 4 6 2 3
3,20
```

則輸出應該是

```
27,5,1,3
```

如果輸入是

```
6
6 11 18 19 27 38
5 12 4 6 2 3
3,19
```

則輸出應該是

```
27,5,1,2
```

如果輸入是

```
5
2 3 5 10 30
5 5 7 8 10
4,36
```

則輸出應該是

```
27,9,2,1
```

## 你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你可以使用任何方法。

## 評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。

## 第三題

(50 分) 承上題，現在有另一家診所，該診所也只有恰好一位醫師，也有許多患者陸續前來，這些患者也都有各自的抵達時間和看診時間長。但這間診所是個不想讓年長者等太久的診所，所以在本題中我們也要根據患者的年齡資訊，推算每個人的開始看診時間以及其他資訊。當一位患者抵達時，醫院會獲得他的年齡資訊，患者  $i$  的年齡為  $g_i$ 。醫院接著會引導他詢問排在他前面一位的患者的年齡，如果他比他前面一位年長至少  $b$  歲（包含  $b$  歲）時，醫院會讓這位年長患者跟前面這位相對年輕的患者交換序位，然後再繼續往前看是否可以繼續交換，直到不能交換為止<sup>2</sup>。無論如何，都只能跟等待中的患者交換位置；已經在接受看診的患者無論如何都會被看完，不會被中斷。

<sup>2</sup>這個流程很像 insertion sort 的找到正確位置插入。

舉例來說，假設患者們的資訊如表 4 所示，且  $b = 15$ ，則前兩位患者的看診序號就是他們的抵達序號，但當第四位患者於第 19 分鐘抵達時，第三位患者正在等待中，且因為  $g_4 \geq g_3 + 15$ ，第四位患者比排在他前面一位的第三位患者年長 15 歲以上，因此第四位患者會跟第三位患者交換。需注意此時第二位抵達的患者已經在接受看診了，所以雖然第四位患者的年齡也比第二位患者大 15 歲以上，但第四位患者還是要等第二位患者看診完離開，才會接受看診。等第二位患者於第  $f_2 = 23$  分鐘離開時，第四位患者會接受看診，因此第四位患者的等待時間長為  $w_4 = 23 - 19 = 4$ ，看診完成時間則為  $f_4 = 23 + 6 = 29$ ；在第四位患者離開時，第三位患者會接受看診，因此第三位患者的等待時間長為  $w_3 = 29 - 18 = 11$ ，看診完成時間則為  $f_3 = 29 + 4 = 33$ 。根據以上討論，我們可以推算所有患者的看診序號（跟他的抵達序號可能不一樣），開始接受看診的時間、看診完成時間、等待時間長，以及抵達前一瞬間的診所中人數。這些資訊整理如表 5。

項目	抵達順序 $i$					
	1	2	3	4	5	6
年齡 $g_i$	38	37	34	52	47	25
到達時間 $a_i$	6	15	18	19	27	38
看診時間長 $s_i$	5	8	4	6	2	3

表 4: 患者資訊

項目	抵達順序 $i$					
	1	2	3	4	5	6
看診序號	1	2	4	3	5	6
看診完成時間 $f_i$	11	23	33	29	35	41
等待時間長 $w_i$	0	0	11	4	6	0
患者 $i$ 到達的診所內人數 $p_i$	0	0	1	2	2	0

表 5: 推算結果

在本題中，診所依然希望在任意指定一個抵達順序  $k$  之後，你可以幫他們找出患者  $k$  的看診完成時間  $f_k$ 、從抵達診所到接受看診的等待時間長  $w_k$ ，以及他到達診所的前一瞬間診所中的總人數  $p_k$ 。診所也希望在給定時間點  $t$  後，你可以幫他們算出在第  $t$  分鐘那個瞬間的前一瞬間的診所中的總人數  $x(t)$ 。

**特別說明 1：**實作本題時，一個顯然不錯的方法是使用「類別」，例如建立一個叫 `Patient` 的類別，然後讀取資料後建立  $n$  個患者物件，並且在每次建立患者物件的時候，看看是否要把這個患者物件往前交換數個位置，而患者物件間的比較可以用一個 `Patient` 的成員函數來完成。若要記錄患者看診的先後順序，我們可以建立一個指標陣列 `Patient** serviceOrder = new Patient*[patientCnt];`，其中 `patientCnt` 是病患個數，讓 `serviceOrder[i]` 去指向第  $i + 1$  個被看診的病患，當患者物件間要交換看診序號時，就交換 `serviceOrder` 裡相對應元素的內容（記憶體位址）即可，跟教 C String 的時候那個幫一堆字串排序的例子很像。簡言之，善用類別和物件和指標，會讓這一題的開發變得簡單許多，程式碼也會比較容易理解和好維護。

**特別說明 2：**雖然實務上這不是唯一的作法，但的確是不錯的做法，也是這一次作業要讓大家練習的內容，所以我們會請助教檢視大家的程式碼，看看大家是否有合理地使用類別、物件和指標，如果沒有的話，還是會酌予扣分。

## 輸入輸出格式

系統會提供一共 15 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有五列，其中第一列裝著兩個正整數，依序是  $n$  和  $b$ ；在第二列裝著  $n$  個正整數，依序是  $g_1$ 、 $g_2$ 、 $g_3$  直到  $g_n$ ；在第三列裝著  $n$  個正整數，依序是  $a_1$ 、 $a_2$ 、 $a_3$  直到  $a_n$ ；第四列裝著  $n$  個正整數，依序是  $s_1$ 、 $s_2$ 、 $s_3$  直到  $s_n$ ；第五列裝著一個正整數  $k$ 、一個逗點、一個正整數  $t$ 。已知  $1 \leq n \leq 100$ 、 $0 \leq b \leq 20$ 、 $10 \leq g_i \leq 100$ 、 $1 \leq a_1 < a_2 < a_3 < \dots < a_n \leq 100000$ 、 $1 \leq s_i \leq 1000$ 、 $k \in \{1, 2, \dots, n\}$ 、 $1 \leq t \leq 10000$ 。

讀入這些資訊後，請依照題目指定的規則計算  $f_k$ 、 $w_k$ 、 $p_k$  和  $x(t)$ ，並將這些數字依序印出，相鄰的兩個數字中間用一個逗點隔開。舉例來說，如果輸入是

```
6 15
38 37 34 52 47 25
6 15 18 19 27 38
5 8 4 6 2 3
3,20
```

則輸出應該是

```
33,11,1,3
```

如果輸入是

```
5 20
70 62 45 68 63
2 3 5 10 30
5 5 7 8 10
4,36
```

則輸出應該是

```
20,2,2,1
```

如果輸入是

```
5 5
20 25 30 35 40
10 11 12 13 14
5 5 5 5 5
5,1
```

則第五個抵達的患者會一路往前交換，變成第二個接受看診，但他抵達的時候（第 14 分鐘）診所內共有四個人，因此輸出應該是

```
20,1,4,0
```



## 你上傳的原始碼裡應該包含什麼

你的原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你**不可以**使用上課沒有教過的方法：

- 確定可以使用的語法包含自定義資料型態、類別，以及在之前的作業中正面表列過的語法。
- 確定不可以使用的語法包含 `printf`、`scanf`、運算子多載、C++ 字串、繼承、多型等等。

請注意正面表列的固然是都確定可以用，但沒有被負面表列的不表示可以用喔！

## 評分原則

- 這一題的其中 30 分會根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。
- 這一題的其中 20 分會根據你所寫的程式的品質來給分。助教會打開你的程式碼並檢閱你的程式的結構、運算邏輯、可讀性（包含排版、變數命名、註解等等）、可擴充性、模組化程度，以及是否使用了還沒教過的語法。請寫一個「好」的程式吧！

## 第四題

**特別說明：**本題為作文題而非程式題，請不要上傳東西到 PDOGS；請把你的答案放在 PDF 檔中上傳到 COOL。由於班上同學很多，但助教人力有限，所以原則上我們會從所有繳交中隨機批改一部份（可能是 50%、90%、100%，但不會低於 50%）。沒被抽到的同學如果有認真回答，在本題就會得到滿分，但不會得到直接的助教回饋，如果沒回答或隨便亂寫當然就不會得分；有被抽到的同學們可能會被扣分，但相對應地也會得到助教的回饋。

（20 分）請敘述一個 C++ 程式開發的任務，是「用物件導向程式設計做會比用程序式程式設計」好很多的，說明其原因。所謂「程序式程式設計」（procedural programming）就是只使用 `if`、`else`、`for`、`while`、`array`、`function` 這些語法，整個程式的主體是演算法；「物件導向程式設計」（object-oriented programming）則是再加上使用 `structure`、`class` 等語法，整個程式的主體是物件之間的互動；要敘述一個「程式開發的任務」，則是要告訴大家所要完成的程式要有哪些功能、能做到哪些事。舉例來說，一個「行事曆」可能就是個好例子，因為一個使用者的行事曆中會有一堆日期和時間，如果完全不使用 `structure` 或 `class`，程式的可讀性、可維護性、可擴充性大概都不會太好，開發上也比較困難。

在本題中，我們強烈建議大家問 AI 工具這個問題，並且透過多次互動得到令人滿意、合適、夠完整的答案。之所以這樣建議，是因為 AI 工具以及各種生成式 AI 其實很能幫助大家學習程式設計。每當我們教一個新觀念（例如 `class`、動態記憶體配置、`constants`、`copy constructor` 等等），很多時候我們的困擾都是「為什麼要這麼做」，此時如果有很多例子，就可能有一些例子幫助我們理解，而能讓每個人理解的例子可能都不一樣，一個老師能舉的例子也有限，那麼請 AI 工具幫我們多舉幾個例子，讀完沒感覺的話就再一個，就很可能幫助大家瞭解這些新觀念要解決的問題、原理、適合使用的場景，以及使用的方法了！

以下是老師跟某個 AI 工具互動的過程，供大家參考：

- 老師：請敘述一個 C++ 程式開發的任務，是「用物件導向程式設計做會比用程序式程式設計」好很多的，並且說明原因。
- 某 AI 工具：One example where Object-Oriented Programming (OOP) in C++ can be more beneficial than Procedural Programming is in the development of a Graphical User Interface (GUI) for an application.
  - OOP allows for the creation of objects (like buttons, text boxes, etc.) that can be reused throughout the application, reducing code redundancy.
  - The encapsulation feature of OOP ensures that the data and methods within an object are safe from accidental alteration.
  - Inheritance allows new objects to be created from existing ones, promoting code reusability and efficiency.
  - Polymorphism allows objects to take on many forms depending on the situation, increasing flexibility. Can you think of how these features might be applied in a GUI application?
- 老師：請給我另一個例子，不要視窗，也不要繼承和多型
- 某 AI 工具：Sure, let's consider the development of a complex simulation system, such as a traffic simulation.
  - In a traffic simulation, each car, pedestrian, or traffic light can be represented as an object. Each object can have its own properties (like speed, location) and behaviors (like move, stop).
  - Using OOP, you can encapsulate these properties and behaviors within each object. This makes the code easier to understand and maintain, as each object's data and methods are grouped together.

Even without inheritance and polymorphism, the ability to create multiple instances of an object (like multiple cars or pedestrians) can greatly simplify the code and make it more efficient. Can you see how OOP would be beneficial in this scenario?

- 老師：這個很好！但可以告訴我這樣的模擬系統可能有什麼實務應用嗎？為什麼要開發一個模擬系統？
- 某 AI 工具：Traffic simulation systems have a wide range of practical applications:
  - Urban Planning: They can help city planners understand the impact of new roads, traffic lights, or other infrastructure before they are built.
  - Traffic Management: They can assist in optimizing traffic light timings to reduce congestion.
  - Emergency Planning: They can model the impact of road closures or other disruptions, helping authorities plan for emergencies.
  - Autonomous Vehicles: They can be used to test the behavior of autonomous vehicles in various traffic scenarios.

Developing a simulation system allows us to test and predict outcomes in a controlled environment before implementing changes in the real world, which can save time, resources, and potentially lives. Can you think of other potential applications?

- 老師：請把你的前兩個敘述合併起來。先說明我們現在要開發一個程式去最佳化某個路口的紅綠燈秒數，因此我們要開發一個交通模擬系統。接著說明為什麼這樣的一個系統適合用 OOP 的精神去開發，比用程序式的精神更好
- 某 AI 工具: We're tasked with developing a program to optimize traffic light timings at an intersection, for which we're creating a traffic simulation system. Here's why Object-Oriented Programming (OOP) is a suitable approach:
  - Encapsulation: In OOP, we can encapsulate properties (like speed, location) and behaviors (like move, stop) within each object (like car, pedestrian, traffic light). This makes the code easier to understand and maintain, as each object's data and methods are grouped together.
  - Multiple Instances: OOP allows us to create multiple instances of an object (like multiple cars or pedestrians), simplifying the code and making it more efficient.

In contrast, procedural programming might require us to keep track of each entity and their behaviors separately, which can become complex and difficult to manage as the system grows. Can you see how OOP would be beneficial in this scenario?

綜合以上，本題一個還可以的答案是：

- 假設今天我們要開發一個程式，用於優化交叉路口的紅綠燈時間長度。為此我們需要建立一個交通模擬系統，這個系統中會有一個十字路口，許多車和行人會隨機地出現並且有隨機的移動路線，但都會遵守紅綠燈的規範。這個系統要讓我們可以在指定一組紅綠燈秒數後，透過系統大量地模擬每輛車和每個行人的等待時間，進而計算出總平均等待時間，以便我們最終可以求取能最小化總平均等待時間的紅綠燈秒數。

物件導向程式設計（OOP）是一個適合的方法，這是因為：

1. 封裝：在 OOP 中，我們可以在每個物件（如汽車、行人、紅綠燈）內封裝屬性（如速度、位置）和行為（如移動、停止）。這使得程式碼更容易被理解和維護，因為每個物件的屬性和行為都被組織在一起。
2. 方便建立複數個物件實體：OOP 允許我們創建一個類別的多個物件實體（如多輛汽車或行人），再讓他們互動，這會方便我們維持程式碼的整潔與可讀性。

相較之下，在程序式程式設計中，每個實體的屬性都會獨立存在，我們要對一個實體做更新（例如更改一個行人的位置）時都得要去存取複數個彼此獨立的資料結構（例如陣列），所有讓實體發生行為的函數也會全部都是無法被結構化管理的全域函數（global function）。隨著系統的擴大，程式可能會變得複雜且難以管理。

大家可以發現這個解答是老師把這個 AI 工具提供的解答改一改之後得到的，所以這個 AI 工具確實幫上忙了。但大家應該也有發現老師確實有換句話說，並且按照自己的認知完成答案，這也是大家得要做的（善用生成式 AI，而不是完全依賴）。最後，上面的例子只是個簡單的範例，你的答案應該要更完整一點，包含任務敘述和 OOP 的好處。

你舉的例子和你列舉的原因可以包含目前沒教過的（例如繼承和多型），但你的敘述必須要正確並且讓我們相信你確實理解。與其寫你不確定的東西，我們建議大家還是寫你確定的就好了。