Homework 7

日期:2024/11/3
撰寫者:喻慈恩

題目:

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。

回答:

假設今天我們要開發一個程式，用於自助點餐系統，讓顧客可以自行選擇餐點並建立訂單，最終計算出總金額並結帳。我們需要建立一個自助點餐模擬系統，這個系統中會有菜單項目(例如漢堡、薯條、飲料等)，顧客可以選擇不同數量的菜單項目加入訂單，並可以看到訂單的詳細項目和總金額。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 MenuItem 表示菜單項目、OrderItem 表示訂單中的項目、Order 表示顧客訂單)內封裝屬性(如菜單項目的名稱、價格，訂單項目的數量)和行為(例如計算訂單金額、顯示訂單詳情)。這使得程式碼更容易理解和維護，因為每個物件的屬性和行為都被組織在一起。例如，OrderItem 封裝了單一菜單項目和數量的組合，Order 則包含所有 OrderItem，並具備計算總金額的行為。

2.方便建立複數個物件實體

OOP 允許我們基於類別來創建多個物件實體(如多個菜單項目、訂單項目)，並讓這些物件實體互相作用。舉例來說，每一份訂單包含了不同的訂單項目，且每一個訂單項目都代表不同數量的菜單項目。透過物件的組合，我們能更清晰地組織每個訂單中的細節，同時可以有效地管理程式碼結構，提高可讀性。

3.擴充性

當我們需要擴展系統功能時，OOP 的結構更便於調整。假設未來我們需要加入優惠折扣功能或不同支付方式，這些功能可以作為訂單的額外方法或屬性而新增，而不需要大幅更動現有程式碼。

相較之下，在程序式程式設計中，我們可能需要以多個分開的資料結構來儲存菜單項目、訂單項目和訂單的細節，例如使用多個陣列分別儲存菜單項目名稱、價格和數量。當我們要對訂單進行更新(例如更改某項目的數量)時，就必須手動管理多個資料結構，導致程式碼雜亂且難以維護。此外，所有與訂單、菜單操作相關的函數會成為無法被結構化管理的全域函數(global functions)，這樣會讓程式隨著系統擴大而變得複雜且難以管理。

總結來說，透過物件導向設計，自助點餐系統的每個功能都可以模組化，讓整個系統的設計和維護更具效率，並且在擴展新功能時不會影響原有的程式碼結構。