Homework 9

日期:2024/11/18
撰寫者:喻慈恩

題目:

在第三題中，你應該寫了不少類別、函數、常數等等。

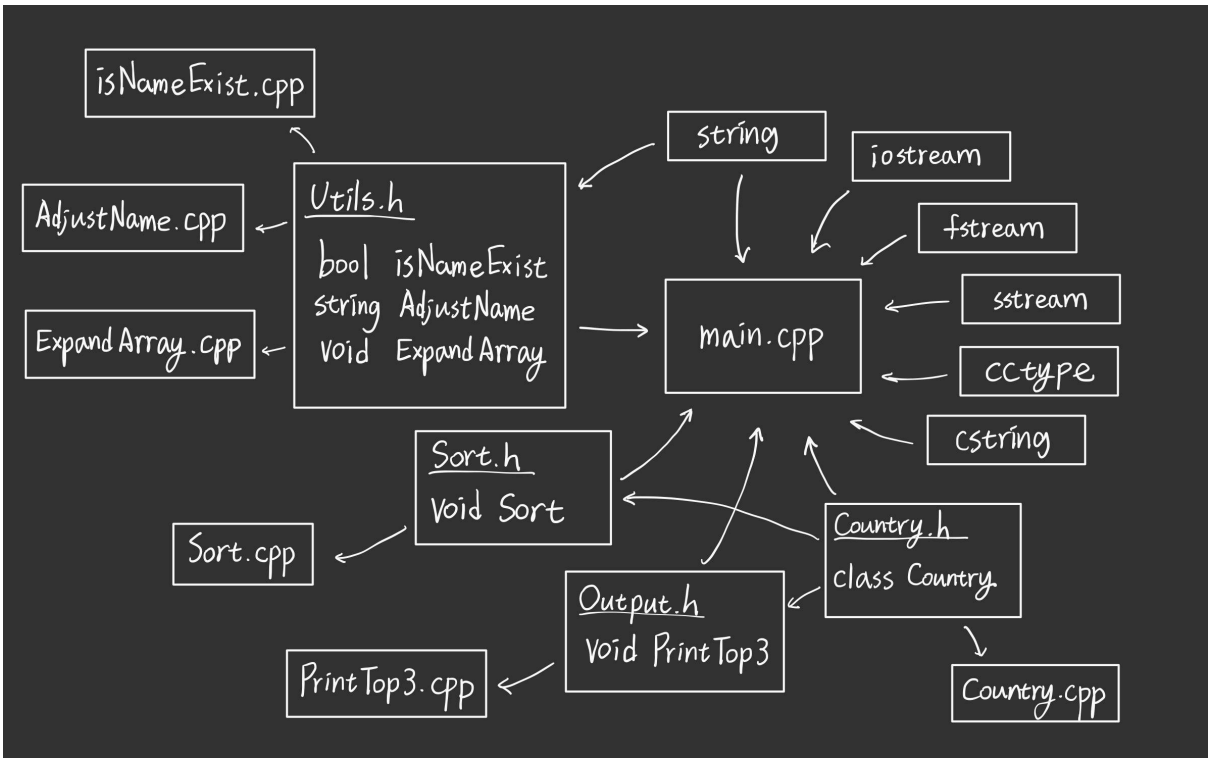
請寫一個或數個自己的.h檔，把你認為合適的程式碼放進去，再把剩下的程式碼放在一個或數個.cpp檔。
請確保你分割完程式碼之後，這些程式碼在你的開發環境中是可以被正確編譯和執行的。

請畫一張圖說明你的程式架構，用一個框框代表一個檔案或一個C++的標準函式庫（例如iostream那些），並且用A→B表示B有include A。在每一個代表你寫的檔案的框框裡面或旁邊寫下該檔案裡面宣告的類別、函數、常數（只要寫在「宣告」該類別、函數、常數的地方就好，不用寫在「定義」該類別、函數、常數的地方。

最後，請在你的架構圖後面用成段落的文字說明你設計這樣的程式架構（將這些類別、函數、常數這樣分割到不同檔案）的設計理念。

回答:

架構圖



設計理念

1. 職責分離
將程式分割成多個.h檔案是為了實現單一責任原則(Single Responsibility Principle, SRP)。每個檔案專注於一個特定功能模組，讓程式更易於理解和維護:

•Country.h：負責國家資訊的資料結構，包含屬性、操作方法及運算子重載。
•Utils.h：通用工具函數(如名稱調整、檢查是否存在、擴充陣列等)，這些函數與業務輯無直接關係，但經常會被重複使用。
•Sort.h：專注於排序功能，與國家類別及其資料排序密切相關，但與其他模組無關。
•Output.h：處理與輸出相關的功能，確保輸出的邏輯集中管理，易於修改格式或行為。

2. 高內聚、低耦合
每個.h檔案的內容與其相關的功能緊密聯繫(高內聚)，但不同模組間的依賴盡量減少(低耦合)。例如，Utils.h提供通用工具函數，不依賴於Country 的內部實現，只需傳入必要參數即可。

3. 提升可維護性與可擴展性
模組化設計讓程式易於擴展:
• 如果需要改進排序功能，只需修改 Sort.h 及其對應的 .cpp 檔案，無需動其他模組。
•若未來新增輸出格式(如寫入文件)，只需修改或擴展 Output.h。

4. 提升程式碼的可重用性
分離工具函數和特定模組的類別定義，讓這些功能能夠在其他專案中重用。例如，expandArray 和 AdjustName 可直接應用於其他需要動態陣列或名稱處理的專案。

透過這樣的設計，程式不僅結構更清晰，也有助於團隊開發中的模組分工和版本控制。