

第一題：實現 Polynomial 類別的 ADT

解題說明

1. 多項式需要儲存多個非零項，因此使用 Term 類別來表示單個項，包含係數 (coef) 與指數 (exp)。
 2. 使用動態陣列 (termArray) 來存放多項式的所有非零項，並設計變數來管理大小與容量。
 3. 提供多項式的基本操作方法，包括加法、乘法與在指定點的值計算。
 4. 支援擴充性：動態調整儲存空間以處理更多項數。
-

程式碼說明

1. 成員函數：
 - Polynomial::Add: 將兩個多項式的對應項進行合併，返回結果。
 - Polynomial::Mult: 計算兩個多項式的乘積，並返回新多項式。
 - Polynomial::Eval: 將變數值代入並計算多項式的結果。

2. 內部管理：

- Resize: 當需要存放的項超過當前容量時，調整動態陣列大小。
-

效能分析

1. 時間複雜度：

- 加法 (Add) : $O(n + m)$ ，其中 n 和 m 是兩個多項式的項數。
- 乘法 (Mult) : $O(n * m)$ ，需要計算所有項的乘積。
- 評估 (Eval) : $O(n)$ ，依次計算每項的值。

2. 空間複雜度：

- 動態陣列的使用增加了存儲靈活性，但可能導致額外的內存分配成本。
-

測試與驗證

1. 測試案例：

- 測試空多項式的初始化。

- 測試兩個簡單多項式的加法與乘法。
- 測試多項式在特定點的計算。

2. 驗證方法：

- 手動計算結果並與程式執行結果比較。
-

效能量測

- 測試大規模多項式（例如，1000 項）進行加法和乘法。
 - 使用計時器（如 `std::chrono`）測量操作所需時間。
-

第二題：實現運算子重載以支持多項式的輸入與輸出

解題說明

1. 透過重載 `<<` 和 `>>` 運算子，讓多項式的輸入與輸出更符合 C++ 標準的操作風格。
2. 輸入多項式時，讀取非零項的數量和每項的係數與指數。

3. 輸出多項式時，需按照數學格式顯示，並處理特殊情況（如只有常數項）。
-

程式碼說明

1. 輸入運算子 (>>) :

- 提供一個 `istream` 的介面，讓用戶輸入多項式的每個項（係數與指數）。
- 更新 `termArray`。

2. 輸出運算子 (<<) :

- 將多項式轉換為格式化的字串（如 $3x^2 + 2x + 1$ ）。
 - 處理符號（+ 或 -）與特殊情況（如 0）。
-

效能分析

1. 時間複雜度：

- 輸入： $O(n)$ ，需要處理每一項。
- 輸出： $O(n)$ ，格式化每一項。

2. 空間複雜度：

- 輸出操作可能需要臨時的字串儲存。
-

測試與驗證

1. 測試案例：

- 輸入並輸出單項與多項多項式。
- 測試格式化輸出的正確性，特別是符號與係數的處理。

2. 驗證方法：

- 檢查輸出格式是否符合數學規範。
 - 使用多種不同的輸入，確保操作的穩定性。
-

效能量測

- 使用包含大量項的多項式測試輸入和輸出的速度。
 - 比較運算子重載前後的執行效率差異。
-

申論及開發報告

一、問題背景與需求

本題要求設計並實現一個多項式類別，支持以下功能：

1. 多項式的加法、乘法及求值，用於數值計算和符號運算。
 2. 多項式的輸入與輸出運算子重載，便於用戶直觀操作。
-

二、系統設計

1. 結構設計

- Term 類別：儲存每項的係數與指數。
- Polynomial 類別：包含多項式的主要操作和一個動態陣列儲存非零項。

2. 功能實現

- 加法：合併兩個多項式的非零項。
- 乘法：計算每對項的積並合併同類項。
- 求值：代入變數值，逐項計算結果。

3. 記憶體管理

- 動態陣列初始化後自動擴充容量，確保靈活性。

4. 運算子重載

- >>：支持格式化輸入多項式。
- <<：以數學表達式格式輸出多項式。