資料結構 HW3

解題說明

此程式旨在實作一個多項式運算系統，使用循環鏈結串列來儲存與操作多項式。多項式的運算包括加法、減法、乘法，以及計算給定 x 值的多項式結果。

設計重點為：

1. 運用節點來儲存多項式的係數和指數。
2. 實作多項式的基本操作（輸入、輸出、加、減、乘）。
3. 支援多項式的深層複製與記憶體釋放。

程式的需求涵蓋以下功能：

- 輸入與輸出多項式。
- 對多項式進行加法、減法與乘法運算。
- 計算多項式在指定變數值下的結果。
- 使用深層拷貝與賦值操作確保正確的物件管理。

程式實作

1. 節點結構

- 每個節點儲存一個多項式項目，包括係數和指數。
- 使用 next 指標構成循環鏈結串列。

2. 多項式類別

- **建構子與解構子**：
  - 預設建構子初始化空多項式。
  - 複製建構子和賦值運算符提供深層拷貝功能。
  - 解構子釋放節點記憶體以避免洩漏。
- **運算符重載**：
  - Operator +：將兩個多項式相加。
  - Operator -：執行多項式減法。
  - Operator *：執行多項式乘法。
  - Operator >> 和 operator << ：用於多項式的輸入與輸出。
- **輔助函數**：
  - addNode：插入新節點至多項式。

- Evaluate：計算多項式值。

3. 主程式邏輯

- 接收使用者輸入多項式，進行基本操作。
- 驗證複製建構子與賦值操作。
- 執行加、減、乘運算，並計算給定變數值的結果。
- 使用解構子測試物件的正確釋放。

完整程式碼如本文所附。

效能分析

時間複雜度

- **多項式加法與減法**：
  - 若兩個多項式分別有 n 和 m 項，則加法與減法的時間複雜度為 O(n+m)。
- **多項式乘法**：
  - 兩個多項式的乘法需要計算每一項的組合，時間複雜度為 O(n*m)。
- **多項式評估**：
  - 評估多項式需要遍歷所有節點，時間複雜度為 O(n)。

空間複雜度

- 使用循環鏈結串列，節點的額外空間開銷為每個節點的指標大小。
- 複製建構子和運算符可能會分配額外記憶體，但結構緊湊，適合處理動態項數的多項式。

測試與驗證

測試用例

1. **基本操作測試**：
   - 輸入多項式 3 2 3 -4 2 3 0，驗證輸出格式是否正確：

     輸入多項式 A: 2x^3 - 4x^2 + 3

輸出多項式 A: 2x^3 - 4x^2 + 3 ``` 2. **加法測試**：

- 多項式 A: 2x^3 - 4x^2 + 3，多項式 B: x^2 - 1。
- 結果：2x^3 - 3x^2 + 2。

3. **減法測試**：
    - 多項式 A: 2x^3 - 4x^2 + 3，多項式 B: x^2 - 1。
    - 結果：2x^3 - 5x^2 + 4。
4. **乘法測試**：
    - 多項式 A: x + 1，多項式 B: x - 1。
    - 結果：x^2 - 1。
5. **評估測試**：
    - 多項式 A: 2x^3 - 4x^2 + 3，當 時，計算結果是否正確。

另有多貼一張驗證結果在 github 上，以示測試之公平。

驗證結果

- 運算符的正確性與多項式的輸出格式均通過測試。
- 釋放記憶體後（解構子）多項式輸出為 "Empty Polynomial"，驗證物件安全釋放。

申論及開發報告

成就

1. 使用循環鏈結串列有效處理多項式項目。
2. 實現深層複製與記憶體釋放，確保類別的正確性與穩定性。
3. 支援多種運算符重載，提升使用者體驗。

改進空間

1. **效率優化**：
    - 在乘法運算中，若多項式項目經常變動，可考慮使用跳表或排序後的線性結構來提升查找效率。
2. **輸入驗證**：
    - 增加輸入的格式檢查，避免不合法數據導致程式崩潰。
3. **減少冗餘節點**：
    - 在加減運算後，過濾系數為 0 的節點以簡化多項式結構。

開發經驗

在實作過程中，循環鏈結串列的使用展示了如何高效處理動態資料結構，同時記憶體管理為穩定的程式提供了保障。未來可以將此多項式類別擴展為泛型類別，支援更多數值型別與操作。

結論

本程式成功實現了一個功能完整的多項式運算系統，經過測試與驗證，能正確執行基本運算與記憶體管理。未來的改進方向包括效率優化與使用者介面友善化。