解題說明

1. 建立多項式
2. 新增項到多項式
3. 計算多項式和
4. 計算多項式乘積
5. 評估多項式值

Term類 包含每一項係數和指數

Polynomial 類

NewTrem 新增項的方法 如果項數已滿擴展

Add 返回兩個多項式和

Mult 返回兩個多項式乘積

Eval 計算多項式給定值處結果

**解題思路**

這段 C++ 程式碼的主要目的是實現一個表示多項式的類 Polynomial，並提供基本的多項式操作，如新增項 (NewTerm)、加法 (Add)、乘法 (Mult) 和評估 (Eval)。

**類的設計**

1. **Term 類**：表示多項式中的一項，包含兩個私有成員變數：系數 (coef) 和指數 (exp)。
2. **Polynomial 類**：
   * 私有成員變數：
     + Term \*termArray：用來存儲多項式中的非零項的數組。
     + int capacity：數組的容量。
     + int terms：當前多項式中的非零項數量。
   * 公有成員函數：
     + Polynomial(int cap = 10)：建構函數，初始化多項式。
     + ~Polynomial()：析構函數，釋放動態分配的記憶體。
     + void NewTerm(const float theCoeff, const int theExp)：新增一項到多項式中。
     + Polynomial Add(const Polynomial& poly) const：返回當前多項式和另一個多項式的和。
     + Polynomial Mult(const Polynomial& poly) const：返回當前多項式和另一個多項式的乘積。
     + float Eval(float f) const：評估多項式在某個值處的結果

**效能分析**

**NewTerm 方法**

* **時間複雜度**：在最壞情況下（當需要擴展數組容量時），時間複雜度為 O(n)O(n)O(n)，因為需要拷貝現有的項到新的數組。然而，擴展容量的操作是指數次的，平均時間複雜度接近於 O(1)O(1)O(1)。
* **空間複雜度**：需要額外的空間來存儲新的數組，最壞情況下空間複雜度為 O(n)O(n)O(n)。

**Add 方法**

* **時間複雜度**：O(n+m)O(n + m)O(n+m)，其中 nnn 和 mmm 分別是兩個多項式的項數。因為需要逐項比較並將結果存入新的多項式中。
* **空間複雜度**：O(n+m)O(n + m)O(n+m)，需要額外的空間來存儲結果多項式。

**Mult 方法**

* **時間複雜度**：O(n×m)O(n \times m)O(n×m)，其中 nnn 和 mmm 分別是兩個多項式的項數。因為每一項需要與另一多項式的每一項進行乘法運算。
* **空間複雜度**：O(n×m)O(n \times m)O(n×m)，需要額外的空間來存儲結果多項式。

**Eval 方法**

* **時間複雜度**：O(n)O(n)O(n)，其中 nnn 是多項式的項數。需要遍歷所有項並計算其值。
* **空間複雜度**：O(1)O(1)O(1)，只需要常數空間來存儲結果

心得總結

這段程式碼通過類和動態數組的方式實現了多項式的表示和基本操作，並且在確保效能的前提下，實現了動態擴展數組的功能