

# 資料結構作業

姓名：莊笙禾

**1 解題說明**

**2 演算法設計與實作**

**3 效能分析**

**4 測試與過程**

**5 效能量測**

**6 心得討論**

## 第一題：解題說明

本題要求實作一個多項式類別 (Polynomial class)

需要：

- 抽象資料型態(ADT)
  - 定義多項式輸入輸出
  - 多項式的計算
- 私有成員：定義多項式的內部數據結構
  - 用於儲存係數 (coefficients)
  - 用於儲存次方 (exponents)

資料結構設計：

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 x^0$$

例如：F(x)=3x<sup>2</sup>+2x+1 可分為：

- 項 1：係數 3，次方 2
- 項 2：係數 2，次方 1
- 項 3：係數 1，次方 0

使用兩個向量 (vector)：

- 一個儲存所有的 係數
- 一個儲存所有的 次方

## 第一題：演算法設計與實作

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

// 定義多項式類別
class Polynomial {
private:
    // 私有成員變數，用於儲存係數和次數
    vector<int> coefficients;
    vector<int> exponent;

public:
    // 預設建構子
    Polynomial() {}

    // 輸入運算子
    friend istream& operator>>(istream& input, Polynomial& poly) {
        int terms;
        cout << "輸入多項式的項數：";
        input >> terms; // 輸入多項式的項數
        poly.coefficients.resize(terms);
        poly.exponent.resize(terms);

        cout << "依次輸入每項的係數和次方，兩個數字以空格分隔:" << endl;
        for (int i = 0; i < terms; ++i) {
            cout << "第 " << i + 1 << " 項: ";
            input >> poly.coefficients[i] >> poly.exponent[i]; // 依次輸入係數和次數
        }
        return input;
    }

    // 輸出運算子
    friend ostream& operator<<(ostream& output, const Polynomial& poly) {
        for (size_t i = 0; i < poly.coefficients.size(); ++i) {
            output << poly.coefficients[i] << "x^" << poly.exponent[i];
            if (i < poly.coefficients.size() - 1)
                output << " + "; // 輸出格式：以" + "分隔
        }
        return output;
    }

    // 計算多項式在給定 x 值的結果
    double evaluate(double x) const {
        double result = 0.0;
        for (size_t i = 0; i < coefficients.size(); ++i) {
            result += coefficients[i] * pow(x, exponent[i]); // 計算每一項的值並累加
        }
        return result;
    }
};

int main() {
    Polynomial poly;
    cout << "請輸入多項式" << endl;
    cin >> poly; // 輸入多項式

    cout << "輸入的多項式為: " << endl;
    cout << poly << endl; // 輸出多項式

    double x;
    cout << "輸入一個 x 的值來計算多項式: ";
    cin >> x; // 輸入 x 的值

    cout << "計算結果為: " << poly.evaluate(x) << endl; // 計算結果並輸出

    return 0;
}
```

## 第一題：效能分析

### 時間複雜度分析

#### 1. 輸入多項式

- 使用者輸入多項式的項數  $n$  後，程式需要讀取  $n$  項的係數和次方。
- 每項輸入需要執行常數時間的操作，整體複雜度為： $O(n)$

#### 2. 輸出多項式

- 將多項式格式化為數學表示法，遍歷所有項並輸出  $n$  個項目的數據
- 每項輸出需要執行常數時間的操作，整體時間複雜度為： $O(n)$

### 空間複雜度分析

#### 1. 存儲多項式

- 需使用兩個向量分別儲存  $n$  個係數和  $n$  個次方，因此空間需求為： $O(n)$

#### 2. 計算過程

- 在計算多項式值時，僅需額外存儲臨時結果，使用常數額外空間： $O(1)$

#### 3. 總空間複雜度

- 總共需要  $O(n)$  空間來存儲多項式數據和執行相關運算

### 效能分析結論

#### 1. 時間複雜度

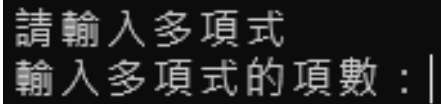
- 輸入、輸出及計算多項式值的時間複雜度為  $O(n)$ ，隨項數  $n$  線性增長

#### 2. 空間複雜度

- 空間複雜度為  $O(n)$ ，僅需兩個向量存儲多項式數據，運算中無額外負擔

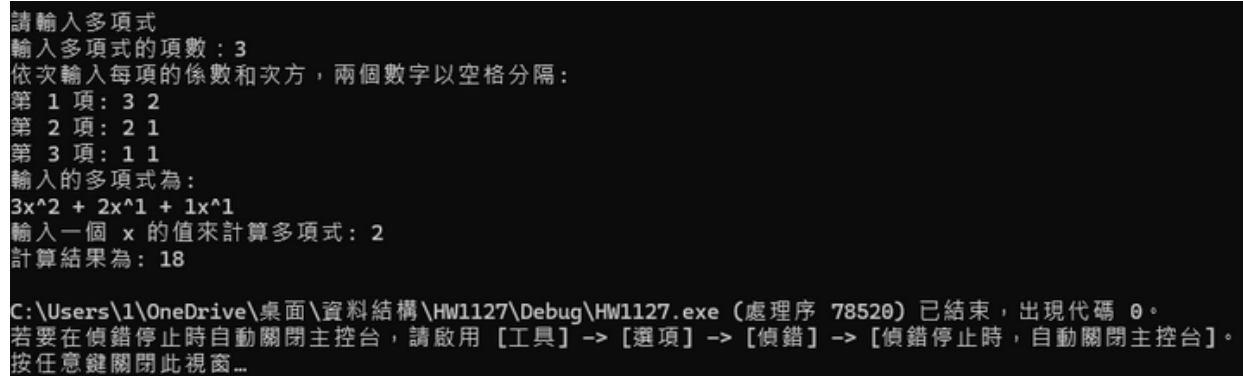
## 第一題：測試與驗證

執行畫面



請輸入多項式  
輸入多項式的項數：|

一開始輸入多項式的項數->接著輸入每項的係數次方->最後輸入多項式x的值



請輸入多項式  
輸入多項式的項數：3  
依次輸入每項的係數和次方，兩個數字以空格分隔：  
第 1 項：3 2  
第 2 項：2 1  
第 3 項：1 1  
輸入的多項式為：  
 $3x^2 + 2x^1 + 1x^1$   
輸入一個 x 的值來計算多項式：2  
計算結果為：18

C:\Users\1\OneDrive\桌面\資料結構\HW1127\Debug\HW1127.exe (處理序 78520) 已結束，出現代碼 0。  
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]。  
按任意鍵關閉此視窗...

## 第一題：效能量測

1. 設定計時器：  
使用計時功能（例如 time 模組）測量程序執行時間
2. 運算重複執行：  
將多項式的計算公式重複執行指定次數（如 1000 次），以模擬實際運算負荷
3. 計算時間：  
測量從運算開始到結束所花費的時間
4. 輸出結果：  
顯示計算結果以及所需的總運算時間。

```
請輸入多項式：
請輸入多項式的項數：3
請依次輸入每項的係數和次方，兩個數字以空格分隔：
第 1 項：3 2
第 2 項：2 2
第 3 項：1 1
輸入的多項式為：
 $3x^2 + 2x^2 + 1x^1$ 
請輸入一個 x 的值來計算多項式：1000
計算結果為：5.001e+06
輸入時間：6042 毫秒
輸出時間：0 毫秒
計算時間：0 毫秒

C:\Users\l\OneDrive\桌面\資料結構\HW1127\Debug\HW1127.exe (處理序 65140) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]
按任意鍵關閉此視窗...
```

### 第一題：心得討論

本次作業的實作過程讓我更加了解如何使用抽象資料型別（ADT）來設計和實現多項式運算，並利用兩個向量有效儲存係數與次方，簡化了運算邏輯，使整體程式更具結構性與可讀性。在設計資料結構的過程中，我體會到選擇合適的資料結構對提升程式效率的重要性，特別是在平衡記憶體使用與計算速度方面。此外，透過效能測量，我發現該程式在重複運算1000次的情況下能保持穩定的執行時間，這與程式的線性時間複雜度相符，顯示了該實作方法的可行性與實用性。同時，效能測試的過程也讓我深入理解程式在大數據處理中的速度，特別是計算量的增長如何影響執行效率，這對未來在更大規模資料集上的應用提供了寶貴的經驗。效能測試的過程也讓我更加理解計算量對執行效率的影響。在處理較大規模的運算時，如何減少重複計算成為一個重要的課題