資料結構報告

王語晨

NOV 27, 2024

CONTENTS

1 解題說明 **2**

2 程式實作 **3**

3 效能分析 **4**

4 測試與驗證 **5**

5 申論及開發報告 **6**

CHAPTER 1

解題說明

以多項式實作，已知多項式加法、減法、乘法計算方式如下：

多項式的加法 ［編輯］

兩個多項式相加可以看作是對兩組單項式的和進行重組與合併同類項。通過加法結合律，可以將同類項放在一起，合併之後就得到了兩個多項式的和[1][2]。例如以下的兩個多項式：

$$P = 3X^2 - 2X + 5XY - 2$$
$$Q = -3X^2 + 3X + 4Y^2 + 8$$

它們的和是：

$$P + Q = (3X^2 - 2X + 5XY - 2) + (-3X^2 + 3X + 4Y^2 + 8)$$

化簡之後得到：

$$P + Q = X + 5XY + 4Y^2 + 6$$

多項式的減法 ［編輯］

例：$P = 36x^5 + 7x^4 + 66x^3 + 36x^2 + 66x + 6$ 、 $Q = 5x^5 - 73x^4 - 11x^3 - 11x^2 + 5x + 3$ 則

$$P - Q = (36-5)x^5 + (7+73)x^4 + (66+11)x^3 + (36+11)x^2 + (66-5)x + (6-3) = 31x^5 + 80x^4 + 77x^3 + 47x^2 + 61x + 3$$

多項式乘法 ［編輯］

例如以下的兩個多項式：

$$P = 2X + 3Y + 5$$
$$Q = 2X + 5Y + XY + 1$$

計算它們的乘積，步驟如下：

$$\begin{aligned} PQ = {} & (2X \cdot 2X) + (2X \cdot 5Y) + (2X \cdot XY) + (2X \cdot 1) \\ & + (3Y \cdot 2X) + (3Y \cdot 5Y) + (3Y \cdot XY) + (3Y \cdot 1) \\ & + (5 \cdot 2X) + (5 \cdot 5Y) + (5 \cdot XY) + (5 \cdot 1) \end{aligned}$$

化簡之後得到：

$$PQ = 4X^2 + 21XY + 2X^2Y + 12X + 15Y^2 + 3XY^2 + 28Y + 5$$

實作參見檔案 polynomial.cpp，其 polynomial 函式：

```
// 加法運算符重載
    Polynomial operator+(const Polynomial& other) const {
        return addOrSubtract(other, true); // 呼叫共用函式
    }
    // 減法運算符重載
    Polynomial operator-(const Polynomial& other) const {
        return addOrSubtract(other, false); // 呼叫共用函式
    }
    // 乘法運算符重載
    Polynomial operator*(const Polynomial& other) const {
        Polynomial result;
```

Figure 1.1: polynomial.cpp

```
    // 加法運算符重載
    Polynomial operator+(const Polynomial& other) const {
        return addOrSubtract(other, true); // 呼叫共用函式
    }

    // 減法運算符重載
    Polynomial operator-(const Polynomial& other) const {
        return addOrSubtract(other, false); // 呼叫共用函式
    }

    // 乘法運算符重載
    Polynomial operator*(const Polynomial& other) const {
        Polynomial result;
```

```
        // 逐項相乘
        for (int i = 0; i < termCount; ++i) {
            for (int j = 0; j < other.termCount; ++j) {
                int newCoeff = terms[i].coefficient * other.terms[j].coefficient;
                int newExp = terms[i].exponent + other.terms[j].exponent;

                // 合併相同指數的項
                bool merged = false;
                for (int k = 0; k < result.termCount; ++k) {
                    if (result.terms[k].exponent == newExp) {
                        result.terms[k].coefficient += newCoeff;
                        merged = true;
                        break;
                    }
                }
                if (!merged) result.addTerm(newCoeff, newExp);
            }
        }

        return result;
    }
```

```
private:
    // 共用加法和減法的處理函式
    Polynomial addOrSubtract(const Polynomial& other, bool isAddition) const {
        Polynomial result;
        int i = 0, j = 0;

        while (i < termCount && j < other.termCount) {
            if (terms[i].exponent == other.terms[j].exponent) {
                int coeff = isAddition ? terms[i].coefficient + other.terms[j].coefficient
                    : terms[i].coefficient - other.terms[j].coefficient;
                if (coeff != 0) result.addTerm(coeff, terms[i].exponent);
                i++;
                j++;
            }
            else if (terms[i].exponent > other.terms[j].exponent) {
                result.addTerm(terms[i].coefficient, terms[i].exponent);
                i++;
            }
            else {
                int coeff = isAddition ? other.terms[j].coefficient : -other.terms[j].coefficient;
                result.addTerm(coeff, other.terms[j].exponent);
                j++;
            }
        }

        while (i < termCount) result.addTerm(terms[i].coefficient, terms[i].exponent), i++;
        while (j < other.termCount) {
            int coeff = isAddition ? other.terms[j].coefficient : -other.terms[j].coefficient;
            result.addTerm(coeff, other.terms[j].exponent), j++;
        }

        return result;
    }
};
```

CHAPTER 2

程式實作

```
int main() {
    Polynomial p1, p2;
    // 輸入兩個多項式
    cout << "輸入第一個多項式:\n";
    cin >> p1;
    cout << "輸入第二個多項式:\n";
    cin >> p2;
    // 加法
    Polynomial sum = p1 + p2;
    cout << "加法結果:\n" << sum;
    // 減法
    Polynomial diff = p1 - p2;
    cout << "減法結果:\n" << diff;
    // 乘法
    Polynomial product = p1 * p2;
    cout << "乘法結果:\n" << product;
    // 讓使用者輸入評估點
    float f;
    cout << "輸入評估點 x 的值: ";
    cin >> f;
    // 評估
    cout << "第一個多項式在 x = " << f << " 的值為: " << p1.Eval(f) << endl;
    cout << "第二個多項式在 x = " << f << " 的值為: " << p2.Eval(f) << endl;
```

Figure 2.1: main.cpp

CHAPTER 3

效能分析

$$f(n) = O(n)$$

時間複雜度

1.addTerm 方法:O(1)，每次添加一項只需執行簡單的賦值操作，與已有項數無關。

2.多項式加法與減法：O(n+m) ，其中 n 與 m 是兩個多項式的項數。兩個多項式的項數按指數進行合併，類似於合併排序中的合併步驟。最多需遍歷兩個多項式的所有項，並執行 O(1)的加減操作。

3.多項式乘法：O(n×m)，兩個多項式的每項都需要與另一個多項式的每項相乘。若每次合併結果（合併相同指數的項）需要遍歷結果多項式，則合併的最壞情況為 O(k) ，其中 k 是當前結果的項數。但由於 k <= n×m，總時間複雜度仍為 O(n×m)。

4. 評估多項式: O(n)，其中 n 是多項式的項數。每一項的計算需要 O(1) 的乘冪和加法運算。總計需遍歷所有 n 項。

總結：加法與減法：O(n+m)，乘法：O(n×m)

空間複雜度

1. 靜態陣列

每個多項式的靜態陣列大小固定為 O(100) 。即使實際項數較少，空間仍為常數O(100)。若多項式操作產生的項數超過靜態陣列大小，則會出現溢出的風險（程式中未處理）。

2. 多項式加法和減法: O(n + m)，其中n和m是兩個多項式的項數。儲存結果多項式的臨時陣列，最多包含 n + m項。

3. 多項式乘法: O(n×m)，乘法結果的最大項數可能達到 n×m。結果多項式的臨時陣列最多包含這麼多項。

4. 評估多項式:O(1)，僅需要一個變數來累積結果。

總結：靜態陣列存儲多項式，固定空間需求為 O（100）。

CHAPTER 4

測試與驗證

```
$ g++ main.cpp -o main.exe && ./main.exe
加法結果：
Polynomial Details:
        Term 1:
                Coefficient:3
                Exponent:4
        Term 2:
                Coefficient:2
                Exponent:3
減法結果：
Polynomial Details:
        Term 1:
                Coefficient:3
                Exponent:4
        Term 2:
                Coefficient:2
                Exponent:3
乘法結果：
Polynomial Details:
        Term 1:
                Coefficient:6
                Exponent:7
        Term 2:
                Coefficient:2
                Exponent:3
輸入評估點 x 的值：2
第一個多項式在 x＝2 的值為:16
第二個多項式在 x＝2 的值為:48
```

Figure 4.1: shell command

驗證

在主函式 main（ ）中，先輸入第一個多項式，再輸入第二個多項式，程式執行多項式加法、減法及乘法，然後輸出多項式加法、減法及乘法結果，接下來讓使用者輸入評估點，則輸出評估點在第一個及第二個多項式的值。根據多項式函數的定義，指數相同則係數做相加或相減，因此第一個多項式係數 2 指數 3 和係數 3 和指數 4，第二個多項式係數 4 指數 3 和係數 3 和指數 4，輸出相加結果為係數 6 指數 6，相減結果為係數-2 指數 3。而相乘則是每個項相乘，並合併相同指數項，結果為係數 6 指數 7，最後評估值輸入 2 則第一個多項式回傳 16，第二個多項式回傳 48。

CHAPTER 5

申論與開發報告

申論

這題要求設計一個處理多項式的程式，包括基本的多項式運算（加法、減法、乘法）以及評估多項式值的功能。

- 程式設計細節

(1) 資料結構:定義了 Term 結構，包含兩個成員：coefficient (係數)和 exponent (指數)，使用靜態陣列 terms[100]儲存多項式最多 100 項，並用 termCount 追蹤項目數。

(2) 輸入與輸出操作：輸入 (operator>>)：使用者可逐項輸入「係數」與「指數」，輸入 -1 -1 終止。輸出 (operator<<)：輸出多項式的每一項。

(3) 運算操作

1. 加法與減法：透過共用函式 addOrSubtract 實現，對指數相同的項合併係數。若指數不同，直接加入結果中。

2. 乘法：對兩多項式進行逐項相乘，結果的係數相乘，指數相加。同時合併相同指數的項。

3. 測試與驗證

輸入與輸出測試結果如下：

1. 多項式輸入：第一個多項式：(2x^3 + 3x^4)，第二個多項式：(4x^3 + 3x^4)

2. 加法結果：正確結果為 (6x^3 + 6x^4)。

3. 減法結果：正確結果為 (-2x^3)。

4. 乘法結果：正確結果為(6x^7)。

5. 評估結果：正確結果為輸入評估點 x 的值：2，第一個多項式在 x＝2 的值為:16，第二個多項式在 x＝2 的值為:48。

開發報告

根據題目了解到本題的主要任務包括多項式表示、多項式運算、多項式評估，最後做輸入輸出。多項式表示是利用結構 Term 表示多項式的每一項，並將其存儲於靜態陣列中，而多項式運算則是有加法與減法還有乘法，加法與減法是根據多項式的指數合併係數，乘法則是根據計算項與項的乘積，並合併相同指數的項，接下來是多項式評估，透過代入變數值(x) 計算多項式在該點的值，最後做輸入輸出，透過鍵盤輸入多項式，並顯示結果。