

資料結構報告

學號:41243147

姓名:楊承哲

1 解題說明	2
2 實作	3
3 效能分析	4
4 測試與過程	5

1. 解題說明

這題要使用陣列做一個多項式的計算包括了乘法、加法。

```
class Term {  
    friend class Polynomial;  
    friend ostream& operator<<(ostream&, const Polynomial&);  
    friend istream& operator>>(istream&, Polynomial&);  
public:  
    Term(float c = 0.0f, int e = 0) : coef(c), exp(e) {}  
private:  
    float coef;  
    int exp;  
};
```

這是多項式的內容。

```
Polynomial Polynomial::Add(const Polynomial& poly) const {  
    Polynomial result;  
    int i = 0, j = 0;  
    while (i < termCount && j < poly.termCount) {  
        if (terms[i].exp == poly.terms[j].exp) {  
            float sumCoef = terms[i].coef + poly.terms[j].coef;  
            if (sumCoef != 0 && result.termCount < MAX_TERMS) {  
                result.terms[result.termCount++] = Term(sumCoef, terms[i].exp);  
            }  
            i++; j++;  
        }  
        else if (terms[i].exp > poly.terms[j].exp) {  
            result.terms[result.termCount++] = terms[i++];  
        }  
        else {  
            result.terms[result.termCount++] = poly.terms[j++];  
        }  
    }  
    while (i < termCount) result.terms[result.termCount++] = terms[i++];  
    while (j < poly.termCount) result.terms[result.termCount++] = poly.terms[j++];  
    result.sortTerms();  
    return result;  
}
```

加法實作

```
Polynomial Polynomial::Mult(const Polynomial& poly) const {  
    Polynomial result;  
    for (int i = 0; i < termCount; ++i) {  
        for (int j = 0; j < poly.termCount; ++j) {  
            float prodCoef = terms[i].coef * poly.terms[j].coef;  
            int prodExp = terms[i].exp + poly.terms[j].exp;  
            bool found = false;  
            for (int k = 0; k < result.termCount; ++k) {  
                if (result.terms[k].exp == prodExp) {  
                    result.terms[k].coef += prodCoef;  
                    found = true;  
                    break;  
                }  
            }  
            if (!found && result.termCount < MAX_TERMS) {  
                result.terms[result.termCount++] = Term(prodCoef, prodExp);  
            }  
        }  
    }  
    result.sortTerms();  
    return result;  
}
```

乘法實作

```
float Polynomial::Eval(float x) const {  
    float result = 0;  
    for (int i = 0; i < termCount; ++i) {  
        result += terms[i].coef * pow(x, terms[i].exp);  
    }  
    return result;  
}
```

帶入值實作

2.程式實作

```
int main() {
    Polynomial p1, p2, sum, product;

    cout << "輸入第一個多項式:\n";
    cin >> p1;
    cout << "輸入第二個多項式:\n";
    cin >> p2;

    sum = p1.Add(p2);
    product = p1.Mult(p2);

    cout << "第一個多項式: " << p1 << endl;
    cout << "第二個多項式: " << p2 << endl;
    cout << "加法結果: " << sum << endl;
    cout << "乘法結果: " << product << endl;

    float x;
    cout << "輸入一個值帶入: ";
    cin >> x;
    cout << "P1(" << x << ") = " << p1.Eval(x) << endl;
    cout << "P2(" << x << ") = " << p2.Eval(x) << endl;
    return 0;
}
```

輸入第一個多項式有幾項，然後輸入第一項的第一項的係數與指數輸入至N項，換輸入第二個多項是有幾項然後一樣輸入至N項，程式就會輸出加法結果與乘法的結果，並可以由使用者自行輸入要帶入的數字帶到兩式之中。

效能分析

1. 時間複雜度：

- 加法 `Add`： $O(n + m)$ ， n 為第一個多項式項數， m 為第二個多項式項數。
- 乘法 `Mult`： $O(n * m)$ ，每項兩兩相乘。
- 求值 `Eval`： $O(n)$ ，遍歷所有項。

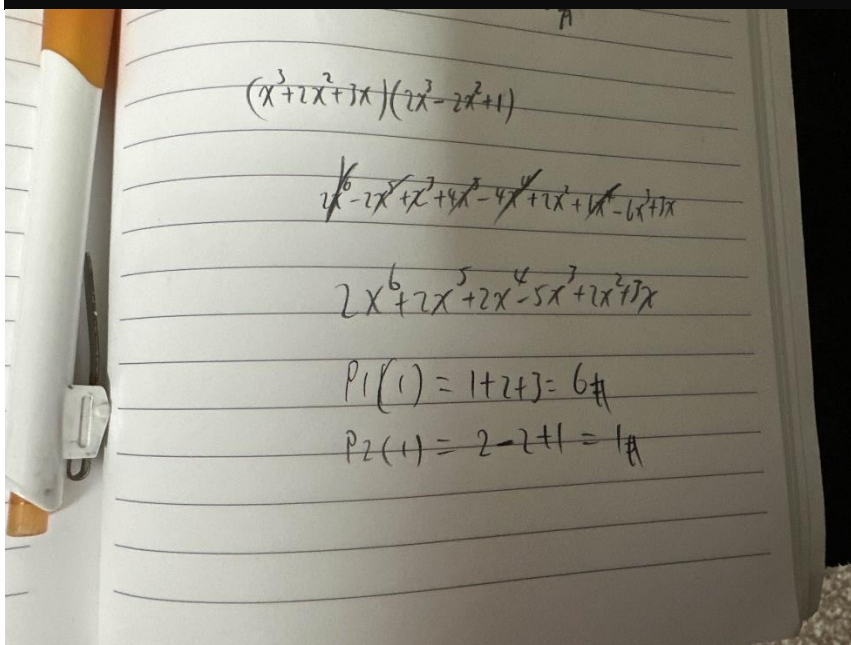
2. 空間複雜度：

- 靜態陣列存儲 `MAX_TERMS`，空間複雜度為 $O(1)$ 。

測試

1.

輸入第一個多項式：
 有幾項：3
 輸入第 1 項的係數與指數：1 3
 輸入第 2 項的係數與指數：2 2
 輸入第 3 項的係數與指數：3 1
 輸入第二個多項式：
 有幾項：3
 輸入第 1 項的係數與指數：2 3
 輸入第 2 項的係數與指數：-2 2
 輸入第 3 項的係數與指數：1 0
 第一個多項式： $1x^3 + 2x^2 + 3x^1$
 第二個多項式： $2x^3 - 2x^2 + 1x^0$
 加法結果： $3x^3 + 3x^1 + 1x^0$
 乘法結果： $2x^6 + 2x^5 + 2x^4 - 5x^3 + 2x^2 + 3x^1$
 輸入一個值帶入：1
 $P1(1) = 6$
 $P2(1) = 1$



```
輸入第一個多項式：
有幾項：2
輸入第 1 項的係數與指數：3 2
輸入第 2 項的係數與指數：5 0
輸入第二個多項式：
有幾項：2
輸入第 1 項的係數與指數：2 1
輸入第 2 項的係數與指數：4 0
第一個多項式：3x^2 +5x^0
第二個多項式：2x^1 +4x^0
加法結果：3x^2 +2x^1 +9x^0
乘法結果：6x^3 +12x^2 +10x^1 +20x^0
輸入一個值帶入：2
P1(2) = 17
P2(2) = 8
```

2.

Handwritten work on lined paper:

$$(3x^2 + 5)(2x + 4)$$
$$6x^3 + 12x^2 + 10x + 20$$
$$P_1(2) = 12 + 5 = 17$$
$$P_2(2) = 4 + 4 = 8$$

經過我手算的結果可以發現結果是非常正確的！

心得：

在實作這個程式的時候，比我想像的還要複雜一點，也具有挑戰性，讓我對類別使用更加熟悉。