資料結構 第二份 作業

姓名:林國善

8/6/2024

1 題目

Problems

1. **Implement the Polynomial class its ADT and private data members are shown in Figure 1 and 2, respectively.**
2. **Write C++ functions to input and output polynomials represented as Figure 2. Your functions should overload the << and >> operators.**

2 演算法設計與實作

想法: 這個程式用於實現多項式加法，將兩個多項式的各項係數和指數分別存儲，然後進行相應的加法運算。

1. 創建一個 Term 類別來表示多項式的一項，包括係數和指數。

```
class Term {
    friend class Polynomial;
public:
    float coef; // 係數
    int exp;    // 指數
};
```

2. 創建一個 Polynomial 類別，用來存儲多項式的所有項目和解構程式。（紅色部分）

3. 在 Polynomial 類別中實現多項式的加法輸入和輸出功能。（藍色部分）

```
class Polynomial {
private:
    Term* termArray; // 非零項的數組
    int capacity;    // termArray 的大小
    int terms;       // 非零項的數量

public:
    // 構造函數
    Polynomial(int cap = 10) : capacity(cap), terms(0) {
        termArray = new Term[capacity];
    }

    // 解構函數
    ~Polynomial() {
        delete[] termArray;
    }

    // 重載 >> 運算子，用於輸入多項式
    friend istream& operator>>(istream& input, Polynomial& p);

    // 重載 << 運算子，用於輸出多項式
    friend ostream& operator<<(ostream& output, const Polynomial& p);

    // 多項式加法
    Polynomial operator+(const Polynomial& other) const;

    // 增加一個項
    void NewTerm(float theCoeff, int theExp);
};
```

實作 ” >> ” input

```
// 重載 >> 運算子
istream& operator>>(istream& input, Polynomial& p) {
    int numTerms;
    cout << "輸入項數: ";
    input >> numTerms;

    for (int i = 0; i < numTerms; ++i) {
        float coef;
        int exp;
        cout << "輸入係數和指數: ";
        input >> coef >> exp;
        p.NewTerm(coef, exp);
    }

    // 按照指數降序排序
    sort(p.termArray, p.termArray + p.terms, [](const Term& a, const Term& b) {
        return a.exp > b.exp;
        });

    return input;
}
```

Output “<<”

```
// 重載 << 運算子
ostream& operator<<(ostream& output, const Polynomial& p) {
    for (int i = 0; i < p.terms; ++i) {
        output << p.termArray[i].coef << "x^" << p.termArray[i].exp;
        if (i != p.terms - 1) output << " + ";
    }
    return output;
}
```

4. 使用動態數組來存儲多項式的項目，以便在需要時擴展容量。

```
// 增加一個項，並合併同指數項
void Polynomial::NewTerm(float theCoeff, int theExp) {
    for (int i = 0; i < terms; ++i) {
        if (termArray[i].exp == theExp) {
            termArray[i].coef += theCoeff;
            return;
        }
    }
    if (terms == capacity) {
        capacity *= 2;
        Term* newArray = new Term[capacity];
        copy(termArray, termArray + terms, newArray);
        delete[] termArray;
        termArray = newArray;
    }
    termArray[terms].coef = theCoeff;
    termArray[terms++].exp = theExp;
}
```

5. 在主函數中，輸入兩個多項式，進行加法運算並輸出結果。

```
// 多項式加法
Polynomial Polynomial::operator+(const Polynomial& other) const {
    Polynomial c;
    int aPos = 0, bPos = 0;
    while (aPos < terms && bPos < other.terms) {
        if (termArray[aPos].exp == other.termArray[bPos].exp) {
            float t = termArray[aPos].coef + other.termArray[bPos].coef;
            if (t) c.NewTerm(t, termArray[aPos].exp);
            aPos++;
            bPos++;
        }
        else if (termArray[aPos].exp < other.termArray[bPos].exp) {
            c.NewTerm(other.termArray[bPos].coef, other.termArray[bPos].exp);
            bPos++;
        }
        else {
            c.NewTerm(termArray[aPos].coef, termArray[aPos].exp);
            aPos++;
        }
    }
    for (; aPos < terms; aPos++)
        c.NewTerm(termArray[aPos].coef, termArray[aPos].exp);
    for (; bPos < other.terms; bPos++)
        c.NewTerm(other.termArray[bPos].coef, other.termArray[bPos].exp);
    return c;
}
```

3 效能分析

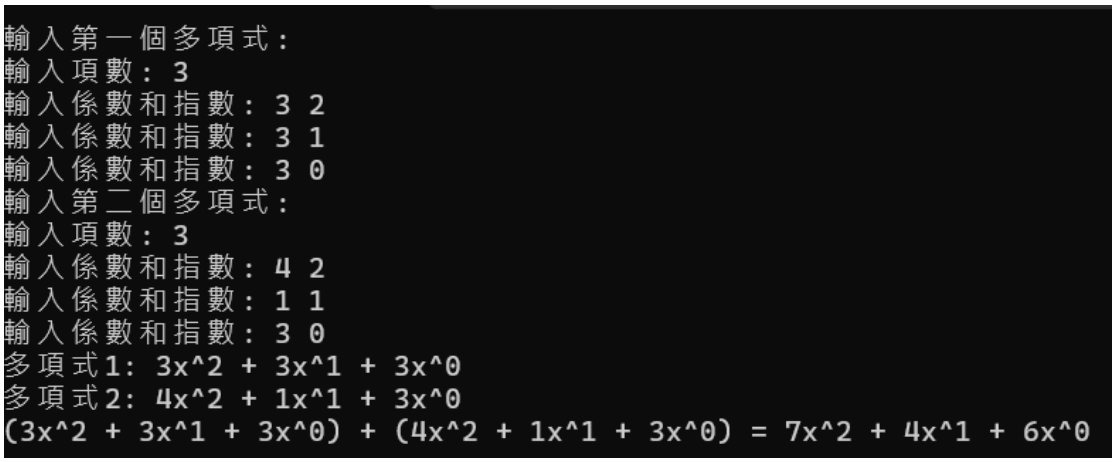
時間複雜度 :

1. 多項式輸入的時間複雜度是 O(n)，其中 n 是多項式的項數。
2. 排序的時間複雜度是 O(n log n)。
3. 多項式加法的時間複雜度是 O(n)，其中 n 是較長多項式的項數。

空間複雜度:

1. 使用動態數組來存儲多項式的項目，其空間複雜度為 O(n)，其中 n 是多項式的最大項數。

4 測試與過程



5 心得討論

這個程式實現了多項式的基本加法運算，通過動態數組管理多項式的項目。使用排序保證了多項式的有序性，使加法運算更為簡單明了。複習到了許多一年級的知識，提高了靈活我對這些的重載和要怎麼處理這些資料的基本概念。