

資料結構作業 2 報告

41243128 徐聖硯

一 · 題目

```
class Polynomial {  
    //  $p(x) = a_0x^{e_0} + \cdots + a_nx^{e_n}$ ; a set of ordered pairs of  $\langle e_i, a_i \rangle$ ,  
    // where  $a_i$  is a nonzero float coefficient and  $e_i$  is a non-negative integer exponent.  
    public:  
        Polynomial();  
        // Construct the polynomial  $p(x) = 0$ .  
  
        Polynomial Add(Polynomial poly);  
        // Return the sum of the polynomials *this and poly.  
  
        Polynomial Mult(Polynomial poly);  
        // Return the product of the polynomials *this and poly.  
  
        float Eval(float f);  
        // Evaluate the polynomial *this at f and return the result.  
};
```

Figure 1. Abstract data type of *Polynomial* class

```
class Polynomial ; // forward declaration
```

```
class Term {  
    friend Polynomial;  
    private:  
        float coef; // coefficient  
        int exp;    // exponent  
};
```

The private data members of *Polynomial* are defined as follows:

```
private:  
    Term *termArray; // array of nonzero terms  
    int capacity;      // size of termArray  
    int terms;         // number of nonzero terms
```

Figure 2. The private data members of *Polynomial* class

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.

二・解題說明

`Polynomial` 裡面塞好塞滿用來表示整個多項式

`Term` 來表示多項式的項目

`Add()` `Mult()` `Eval()`是多項式的加法乘法&值

如果空間不夠會呼叫 `Copy` 跟 `addTerm` 來擴大容量

`operator<< & operator>>` 多載

來處理輸入的每一項 另一個則是處理輸出的每一項

效能量測的部分 使用 `chrono` 函式庫來計時加法和乘法的執行時間並使用

`high_resolution_clock::now()`記錄起始和結束時間並使用 `duration_cast<nanoseconds>`轉換為毫秒來顯示時間

三 · Algorithm Design & Programming

```
// 運算子 >> 重載，用於輸入多項式
istream& operator>>(istream& input, Polynomial& p) {
    float t_coef;
    int t_exp;
    char tmp;
    bool isNegative = false;

    while (true) {
        input >> t_coef; // 讀入係數
        if (isNegative) {
            t_coef *= -1; // 處理負號
            isNegative = false;
        }

        input.get(tmp); // 讀取下一個字元，可能是 'x' 或 '\n'
        if (tmp == '\n') {
            p.AddTerm(t_coef, 0); // 若為換行符，代表此項為常數
            break;
        }

        if (tmp == 'x') {
            input.get(tmp); // 讀取 '^' 或 '\n'
            if (tmp == '^') {
                input >> t_exp; // 讀取指數
            }
            else {
                t_exp = 1; // 若沒有 '^'，指數為 1
                if (tmp == '\n') break;
            }
        }
        else {
            t_exp = 0; // 若沒有 'x'，指數為 0
        }

        p.AddTerm(t_coef, t_exp); // 新增項目

        input.get(tmp); // 讀取下一個項目的分隔符號
        if (tmp == '\n') break; // 若為換行符，結束輸入
        else if (tmp == '-') isNegative = true; // 若為負號，處理下一項
    }

    return input;
}
```

Operator>>實作

輸入進來可能有+-或是空格及 $x^$ 還有數字，

所以判斷輸入進來的東西並讀取換行符號結束輸入；

```
// 運算子 << 多載，用於將多項式輸出
ostream& operator<<(ostream& output, const Polynomial& p) {
    if (p.terms == 0) {
        output << "0"; // 若多項式無項目，輸出 0
        return output;
    }
    for (int i = 0; i < p.terms; i++) {
        if (i > 0 && p.termArray[i].coef > 0) output << " + "; // 若不是第一項且係數為正數，輸出加號
        output << p.termArray[i].coef;
        if (p.termArray[i].exp != 0) output << "x^" << p.termArray[i].exp; // 輸出指數
    }
    return output;
}
```

Operator<<實作

處理輸出，它會根據多項式的項目逐一輸出每一項，並處理符號（如正號和負號）及指數的顯示。

```
// 新增項目
void AddTerm(float coef, int exp) {
    if (coef == 0) return; // 若係數為 0，不加入
    if (terms >= capacity) { // 若容量不足，擴展容量
        Copy(capacity * 2);
    }
    termArray[terms].coef = coef;
    termArray[terms].exp = exp;
    terms++; // 增加項數
}

// 擴展容量
void Copy(int newCapacity) {
    Term* newArray = new Term[newCapacity]; // 創建新的陣列
    copy(termArray, termArray + terms, newArray); // 將原陣列的資料複製到新陣列
    delete[] termArray; // 釋放舊陣列的記憶體
    termArray = newArray; // 將新陣列賦值給 termArray
    capacity = newCapacity; // 更新容量
}
```

這邊是如果容量不夠就擴大的部分

```

// 預設建構子
Polynomial() : termArray(nullptr), capacity(0), terms(0) {}

// 帶容量的建構子
Polynomial(int capacity) {
    this->capacity = capacity;
    terms = 0;
    termArray = new Term[capacity]; // 用來儲存多項式的項目
}

```

建構子

```

// 計算多項式的值，傳入某個 x 的值
double Eval(float f) {
    double result = 0.0;
    for (int i = 0; i < terms; i++) {
        result += termArray[i].coef * pow(f, termArray[i].exp); // 根據每項的係數與指數計算值
    }
    return result; // 返回計算結果
}

```

計算多項式的值

```

// 多項式相乘
Polynomial Mult(const Polynomial& poly) {
    Polynomial result(terms * poly.terms); // 創建一個結果多項式，容量為兩多項式項數的乘積
    for (int i = 0; i < terms; i++) { // 遍歷 p1 的每一項
        for (int j = 0; j < poly.terms; j++) { // 遍歷 p2 的每一項
            float newCoef = termArray[i].coef * poly.termArray[j].coef; // 係數相乘
            int newExp = termArray[i].exp + poly.termArray[j].exp; // 指數相加
            result.AddTermToResult(newCoef, newExp); // 加入結果多項式
        }
    }
    return result; // 返回乘法結果
}

```

多項式的相乘

```

// 多項式相加
Polynomial Add(const Polynomial& poly) {
    Polynomial result(max(capacity, poly.capacity)); // 創建一個結果多項式
    int i = 0, j = 0;
    while (i < terms && j < poly.terms) { // 遍歷兩個多項式的項目
        if (termArray[i].exp == poly.termArray[j].exp) {
            float sumCoef = termArray[i].coef + poly.termArray[j].coef;
            if (sumCoef != 0) result.AddTerm(sumCoef, termArray[i].exp); // 若係數和不為 0，則加入結果
            i++;
            j++;
        }
        else if (termArray[i].exp > poly.termArray[j].exp) { // 若 p1 的指數較大
            result.AddTerm(termArray[i].coef, termArray[i].exp); // 將 p1 的項目加入結果
            i++;
        }
        else { // 若 p2 的指數較大
            result.AddTerm(poly.termArray[j].coef, poly.termArray[j].exp); // 將 p2 的項目加入結果
            j++;
        }
    }
    // 如果 p1 還有剩餘項目，加入結果
    while (i < terms) {
        result.AddTerm(termArray[i].coef, termArray[i].exp);
        i++;
    }
    // 如果 p2 還有剩餘項目，加入結果
    while (j < poly.terms) {
        result.AddTerm(poly.termArray[j].coef, poly.termArray[j].exp);
        j++;
    }
    return result; // 返回加法結果
}

```

多項式的相加

```

// Term 類別用來表示多項式的一項
class Term {
    friend class Polynomial; // 讓 Polynomial 類別可以訪問 Term 類別的私有成員
    friend ostream& operator<<(ostream& output, const Polynomial& p); // 友元函式，用於輸出 Polynomial 類別
private:
    float coef; // 係數
    int exp;    // 指數
};

```

Term


```

int main() {
    Polynomial p1(10), p2(10); // 創建兩個多項式

    // 輸入多項式
    cout << "輸入第一個多項式 (格式: 3x^2 -4x +5):";
    cin >> p1;
    cout << "輸入第二個多項式 (格式: 2x^3 +x -7):";
    cin >> p2;

    //輸出多項式
    cout << "多項式 P1 = " << p1 << endl;
    cout << "多項式 P2 = " << p2 << endl;

    // 多項式相加
    auto start_add = high_resolution_clock::now();
    Polynomial sum = p1.Add(p2);
    cout << "P1 + P2 = " << sum << endl;
    auto end_add = high_resolution_clock::now();
    auto add_duration = duration_cast<nanoseconds>(end_add - start_add).count(); // 計時
    cout << "Add() 需時: " << add_duration / 1e6 << " ms" << endl; // 轉換為毫秒並輸出

    // 多項式相乘
    auto start_mult = high_resolution_clock::now();
    Polynomial product = p1.Mult(p2);
    cout << "P1 * P2 = " << product << endl;
    auto end_mult = high_resolution_clock::now();
    auto mult_duration = duration_cast<nanoseconds>(end_mult - start_mult).count(); // 計時
    cout << "Mult() 需時: " << mult_duration / 1e6 << " ms" << endl; // 轉換為毫秒並輸出

    //計算多項式的值
    float x;
    cout << "輸入 x 的值來計算 P1(x): ";
    cin >> x;
    cout << "P1(x=" << x << ") 的值 = " << p1.Eval(x) << endl;
    return 0;
}

```

這是主程式他創建了兩個多項式

並呼叫 Add() Mult() Eval() 並輸出其結果與需時

四・效能分析

時間複雜度:

Add(): $O(n + m)$ ， n 跟 m 為兩個多項式的項數

Mult(): $O(n * m)$ ， n 跟 m 為兩個多項式的項數

Eval(): $O(n)$ ， n 為多項式的項數

空間複雜度:

為 $O(n)$ ， n 為多項式的項數

五・測試與驗證&效能量測

```
Microsoft Visual Studio 偵錯主控台
輸入第一個多項式 (格式: 3x^2 -4x +5): 3x^2 +5x +1
輸入第二個多項式 (格式: 2x^3 +x -7): -3x^2 -5x -1
多項式 P1 = 3x^2 + 5x^1 + 1
多項式 P2 = -3x^2 -5x^1 -1
P1 + P2 = 0
Add() 需時: 0.116 ms
P1 * P2 = -9x^4 -30x^3 -31x^2 -10x^1 -1
Mult() 需時: 0.6803 ms
輸入 x 的值來計算 P1(x): 2
P1(x=2) 的值 = 23

C:\Users\saint\source\repos\Project3\Debug\Project3.exe (處理序 20520) 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...
```

$$P1 = 3x^2 + 5x + 1$$

$$P2 = -3x^2 - 5x - 1$$

$$P1 + P2 = 3x^2 + 5x + 1 - 3x^2 - 5x - 1 = 0$$

$$\begin{aligned} P1 \times P2 &= (3x^2 + 5x + 1)(-3x^2 - 5x - 1) \\ &= -9x^4 - 15x^3 - 15x^3 - 25x^2 - 3x^2 - 3x^2 - 5x - 5x - 1 \\ &= -9x^4 - 30x^3 - 31x^2 - 10x - 1 \end{aligned}$$

$$x=2 \quad P1 = 3 \times 2^2 + 5 \times 2 + 1 = 23$$

