資料結構報告 HW2

楊道恩

August 6, 2024

CONTENTS

解題說明

Implement Polynomial class from Figure 1.1 and Figure 1.2, and write C++ function to input and output polynomials represented as figure 1.2.

```
class Polynomial {
// p(x) = a_0 x^{e_0} + ··· + a_n x^{e_n}; a set of ordered pairs of <e_i, a_i>,
// 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 1.1 The abstract class Polynomial

```
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 1.2 The private data members of Polynomial class

設計與實作

The implementation of each private public class member of Polynomial is shown in Figure 2.1~2.5, in polynomial.cpp

```
class polynomial; //forward decleartion

class Term{
    friend polynomial;
private:
    float coef;
    int exp;
public:
    Term(float co = 0, int ex = 0):exp(ex), coef(co){}
};

class polynomial {
private:
    Term* termArray;
    int capacity;
    int terms;
public:
    polynomial():termArray(new Term[2]),capacity(2),terms(0) {}
    //forward decleartions
    polynomial Add(polynomial b);
    polynomial Mult(polynomial b);
    float Eval(float f);
    void NewTerm(const float theCoeff, const int theExp);
    void display();
};
```

Figure 2.1 The classes Term and polynomial

```
polynomial polynomial::Add(polynomial b) {
    polynomial c;
    int aPos = 0, bPos = 0;
    while((aPos<terms)&&(bPos<b.terms))
        if(termArray[aPos].exp==b.termArray[bPos].exp){
            float t = termArray[aPos].coef + b.termArray[bPos].coef;
            if (t)c.NewTerm(t, termArray[aPos].exp);
            aPos++; bPos++;
        }
        else if (termArray[aPos].exp < b.termArray[bPos].exp) {
            c.NewTerm(b.termArray[bPos].coef, b.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 < b.terms; bPos++)
        c.NewTerm(b.termArray[bPos].coef, b.termArray[bPos].exp);
    return c;
}
```

Figure 2.2 Implementation of Add function

```
polynomial polynomial::Mult(polynomial b) {
    polynomial c;
    float* prods = new float[terms + b.terms + 1];
    fill(prods, prods + terms + b.terms + 1, 0); //initialize
    for (int i = 0; i < terms; i++) {
        for (int j = 0; j < b.terms; j++) {
            //Multiply the term of polynomial A to
            //every term of polynomial B
            float t = termArray[i].coef * b.termArray[j].coef;
            int e = termArray[i].exp + b.termArray[j].exp;
            prods[e] += t;
        }
    }
    //Add terms to polynomial c if the element of prods array is > 0
    for (int k = (terms + b.terms); k >= 0; k--) {
        if (prods[k] != 0)
            c.NewTerm(prods[k], k);
    }
    return c;
}
```

Figure 2.3 Implementation of Mult function

```
float polynomial::Eval(float f) {
    float result = 0;
    for (int i = 0; i < terms; i++) {
        result += termArray[i].coef * pow(f, termArray[i].exp);
    }
    return result;
}

void polynomial::NewTerm(const float theCoeff, const int theExp) {
    if (terms == capacity) {
        capacity *= 2;
        Term* temp = new Term[capacity];
        copy(termArray, termArray + terms, temp);
        delete[] termArray;
        termArray = temp;
    }
    termArray[terms].coef = theCoeff;
    termArray[terms++].exp = theExp;
}
```

Figure 2.4 Implementation of functions Eval and NewTerm

```
void polynomial::display() {
    int i;
    for (i = 0; i < terms; i++) {
        if (termArray[i].exp > 1) {
            cout << termArray[i].coef << "x^" << termArray[i].exp;
        }
        else if (termArray[i].exp == 1) {
            cout << termArray[i].coef << "x";
        }
        else {
            cout << termArray[i].coef;
        }
        if(i<terms-1){ cout << "+"; }
    }
    cout << '\n';
}
```

Figure 2.5 The function to display (output) polynomial

The main() inputs 2 polynomials A and B. **Polynomial C** is the result of *adding* polynomial A and B, and **is the polynomial to output**. Functions *Mult* and *Eval* tests are included in the comments.

```
int main() {
    int numTerms;
    polynomial a, b;
    cout << "Enter terms of A(x): \n";
    cin >> numTerms;
    cout << "Enter polynomial A coefficients, exponents:\n";
    for(int i = 0; i < numTerms; i++){
        float coA;
        int exA;
        cin >> coA >> exA;
        a.NewTerm(coA, exA);
    }
    cout << "Enter terms of B(x): \n";
    cin >> numTerms;
    cout << "Enter polynomial B coefficients, exponents:\n";
    for (int i = 0; i < numTerms; i++) {
        float coB;
        int exB;
        cin >> coB >> exB;
        b.NewTerm(coB, exB);
    }
    polynomial C;
    C = a.Add(b); //Add
    //C = a.Mult(b); //Mult
    cout << "C(x) = A(x) + B(x) = ";
    //cout << "C(x) = A(x) * B(x) = ";
    C.display();
    //cout << C.Eval(2) << '\n'; //Eval

    system("pause");
    return 0;
}
```

Figure 2.6 The main section of polynomial.cpp

效能分析

時間複雜度

$$\text{Add: } T(P) = O(m+n)$$

共執行m+n次迴圈

$$\text{Mult: } T(P) = O(mn*(m+n))$$

迴圈執行m*n次，然後將m+n個項次加入積的多項式

$$\text{Eval: } T(P) = O(n)$$

迴圈執行n次

空間複雜度

$$\text{Add: } S(P) = O(0)$$

$$\text{Mult: } S(P) = O(m+n)$$

積的陣列空間為m+n

$$\text{Eval: } S(P) = O(0)$$

測試與過程

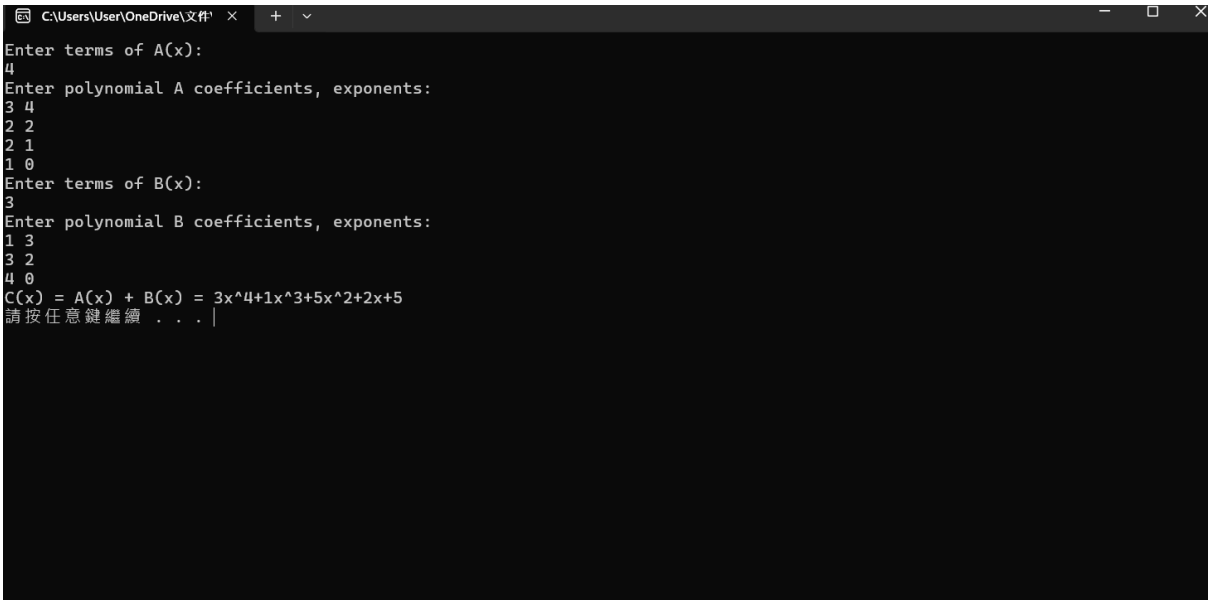

Figure 4.1 Terminal

驗證

$(3x^4 + 2x^2 + 2x + 1) + (1x^3 + 3x^2 + 4)$

$=3x^4 + 1x^3 + 2x^2 + 3x^2 + 2x + 1 + 4$

$=3x^4 + 1x^3 + 5x^2 + 2x + 5$