chapter 2 hw 1

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.**

作業繳交規範

- 解題說明 10%
  - 想法 (How to do?) 陳述，並舉例說明。
- Algorithm Design & Programming 40%
  - Source code + Comment
- 效能分析)Analysis) 15%
  - Time complexity & Space complexity
- 測試與驗證 (Testing and Proving) 15%
- 效能量測 (Measuring) 10%
- 心得討論 10%

完整程式碼:

```
/*  10/31 楊育哲
    題目: 多項式加減乘除
*/
#include <iostream>
#include <math.h>
using namespace std;

class Polynomial;
class Term{
friend ostream& operator<<(ostream& os, const Polynomial& p);
friend istream& operator>>(istream& is, Polynomial &p);
friend Polynomial;
private:
    float coef;
    int exp;
};
class Polynomial{
friend ostream& operator<<(ostream& os, const Polynomial& p);
friend istream& operator>>(istream& is, Polynomial &p);
private:
    int capacity;
public:
    int terms;//運算要用到
    Term *termsArray;
    Polynomial();
    Polynomial Add(Polynomial b);
    Polynomial Mult(Polynomial b);
    float Eval(float f);
    void newTerm(float coef, int exp);
};

int main(){
    Polynomial A;
    Polynomial B;
    cin>>A>>B;
    // cout<<A.Mult(B)<<"\n";
    cout<<A.Add(B)<<"\n";
    return 0;
}
Polynomial::Polynomial():capacity(2),terms(0){
    termsArray = new Term[capacity];
}
void Polynomial::newTerm(float coef, int exp){
    if(capacity==terms){
        capacity*=2;
        Term *temp = new Term[capacity];
        copy(termsArray, termsArray+terms, temp);
        delete[] termsArray;
        termsArray = temp;
    }
    termsArray[terms].coef = coef;
    termsArray[terms++].exp = exp;
}
Polynomial Polynomial::Add(Polynomial b){
    int apos=0, bpos=0;
    Polynomial c;
    while(apos<terms&&bpos<b.terms){
        if(termsArray[apos].exp==b.termsArray[bpos].exp){
            float tmp=termsArray[apos++].coef+b.termsArray[bpos++].coef;
            if(tmp) c.newTerm(tmp, termsArray[apos-1].exp);
        }else if(termsArray[apos].exp>b.termsArray[bpos].exp){
            c.newTerm(termsArray[apos].coef, termsArray[apos].exp);
```

```
            apos++;
        }else{
            c.newTerm(b.termsArray[bpos].coef, b.termsArray[bpos].exp);
            bpos++;
        }
    }
    for(int i=apos; i<terms; i++) c.newTerm(termsArray[i].coef, termsArray[i].exp);
    for(int i=bpos; i<b.terms; i++) c.newTerm(b.termsArray[i].coef, b.termsArray[i].exp);
    return c;
}
Polynomial Polynomial::Mult(Polynomial b){
    Polynomial c;
    for(int apos=0; apos<terms; apos++){
        for(int bpos=0; bpos<b.terms; bpos++){
            int Exp=termsArray[apos].exp+b.termsArray[bpos].exp, cpos=0;
            bool flag=true;
            while(flag&&cpos<c.terms){
                if(c.termsArray[cpos].exp==Exp){
                    flag=false;
                    c.termsArray[cpos].coef+=termsArray[apos].coef*b.termsArray[bpos].coef;
                }
                cpos++;
            }
            if(flag) c.newTerm(termsArray[apos].coef*b.termsArray[bpos].coef, Exp);
        }
    }
    return c;
}
float Polynomial::Eval(float f){
    float ans;
    for(int i=0; i<terms; i++){
        ans+=termsArray[i].coef*pow(f, termsArray[i].exp);
    }
    return ans;
}
ostream& operator<<(ostream& os, const Polynomial& p){
    for(int i=0; i<p.terms; i++){
        if(p.termsArray[i].exp==0){
            os<<p.termsArray[i].coef;
        }else if(i==p.terms-1) os<<p.termsArray[i].coef<<"x^"<<p.termsArray[i].exp;
        else os<<p.termsArray[i].coef<<"x^"<<p.termsArray[i].exp<<"+";
    }
    return os;
}
istream& operator>>(istream& is, Polynomial &p){
    float newCoef;
    int newExp;
    cout<<"start input coef and exp to Polynomial\n";
    while(true){
        cout<<"input coef and exp([0, 0] to end): ";
        is>>newCoef>>newExp;
        if(newCoef==0&&newExp==0) break;
        p.newTerm(newCoef, newExp);
    }
    return is;
}
```

- 解題想法:
    - 多載用算子”>>”、”<<”於輸入輸出(此寫法參考自深碗課程範例程式碼，當天是首次知道可以這樣寫)。其中輸入寫法採一直輸入coef及exp，直到輸入等於[0, 0]，在此之前每項都以

newTerm函式加進多項式裡。

- 加法實作與課本範例程式解法一樣，尋遍兩多項式，比較各個次方項，一樣則相加加進新多項式，不一樣則將大者加進新多項式，最後巡完其中一多項式後，將另一多項式剩下元素加進新多項式中。
- 乘法實作用雙層for迴圈尋訪兩多項式，將兩多項式各個元素與另一多項式各個元素相乘(常數項相乘；次方項相加)，後檢查新的多項式中有無已新增過目標次方項的元素，有則使其常數項增加目標元素(常數項相乘)，無則用newTerm函式加進新多項式。
- 求值函式如f(x)帶入x一樣，尋訪多項式將每項運算完的結果加進變數ans，最後ans即為所求。

ex.

```
int main(){
    Polynomial A;
    Polynomial B;
    cin>>A>>B;
    cout<<A.Add(B)<<"\n";
    //cout<<A.Eval(1.0);
    return 0;
}
```

input: output:

2 2 2x^3+2x^2+3x^1+2

2 1

2 0

0 0

2 3

1 1

0 0

```
start input coef and exp to Polynomial
input coef and exp([0, 0] to end): 2 2
input coef and exp([0, 0] to end): 2 1
input coef and exp([0, 0] to end): 2 0
input coef and exp([0, 0] to end): 0 0
start input coef and exp to Polynomial
input coef and exp([0, 0] to end): 2 3
input coef and exp([0, 0] to end): 1 1
input coef and exp([0, 0] to end): 0 0
2x^3+2x^2+3x^1+2
```

```
int main(){
    Polynomial A;
    Polynomial B;
    cin>>A>>B;
    cout<<A.Mult(B)<<"\n";
    //cout<<A.Eval(1.0);
    return 0;
}
```

input: output:

2 2 4x^3+8x^2+8x^1+4

2 1

2 0

0 0

2 1

2 0

0 0

```
start input coef and exp to Polynomial
input coef and exp([0, 0] to end): 2 2
input coef and exp([0, 0] to end): 2 1
input coef and exp([0, 0] to end): 2 0
input coef and exp([0, 0] to end): 0 0
start input coef and exp to Polynomial
input coef and exp([0, 0] to end): 2 1
input coef and exp([0, 0] to end): 2 0
input coef and exp([0, 0] to end): 0 0
4x^3+8x^2+8x^1+4
```

- 測試與驗證:

  以上面例子來說明，

  輸入一為加法的測資，表示(2x^2+2x^1+2) + (2x^3+x^1) = 2x^3+2x^2+3x^1+2。註解取消的輸出為: 9.0

輸入二為乘法的測資，表示(2x^2+2x^1+2) * (2x^1+2) = 4x^3+8x^2+8x^1+4。註解取消的輸出為: 24.0

- 效能分析(量測): (補充:量測示意在註解, 如// +=8, 表示往下數八項常數步數)

```
void Polynomial::newTerm(float coef, int exp){// +=8
    if(capacity==terms){
        capacity*=2;
        Term *temp = new Term[capacity];
        copy(termsArray, termsArray+terms, temp);
        delete[] termsArray;
        termsArray = temp;
    }
    termsArray[terms].coef = coef;
    termsArray[terms++].exp = exp;
}
```

←newTerm函示程式對照

S(p) = c + Sp = 6+0

T(p) = c + Tp = 8+0

f(8) = O(1)

```
Polynomial Polynomial::Add(Polynomial b){
    int apos=0, bpos=0; // ++
    Polynomial c; // ++
    while(apos<terms&&bpos<b.terms){ // += min(n, m)*9+1
        if(termsArray[apos].exp==b.termsArray[bpos].exp){
            float tmp=termsArray[apos++].coef+b.termsArray[bpos++].coef;
            if(tmp) c.newTerm(tmp, termsArray[apos-1].exp);
        }else if(termsArray[apos].exp>b.termsArray[bpos].exp){
            c.newTerm(termsArray[apos].coef, termsArray[apos].exp);
            apos++;
        }else{
            c.newTerm(b.termsArray[bpos].coef, b.termsArray[bpos].exp);
            bpos++;
        }
    }
    for(int i=apos; i<terms; i++) c.newTerm(termsArray[i].coef, termsArray[i].exp);//(max(n,m)-k)*1+1
    for(int i=bpos; i<b.terms; i++) c.newTerm(b.termsArray[i].coef, b.termsArray[i].exp); // ++
    return c;// ++
}
```

←add函式程式對照

S(p) = c + Sp = 13+2

T(p) = c + Tp = 6+10n

f(10n+6) = O(n)

```
Polynomial Polynomial::Mult(Polynomial b){
    Polynomial c; // +=4
    for(int apos=0; apos<terms; apos++){// *n
        for(int bpos=0; bpos<b.terms; bpos++){// (4n+5)*n
            int Exp=termsArray[apos].exp+b.termsArray[bpos].exp, cpos=0;
            bool flag=true;
            while(flag&&cpos<c.terms){
                if(c.termsArray[cpos].exp==Exp){
                    flag=false;
                    c.termsArray[cpos].coef+=termsArray[apos].coef*b.termsArray[bpos].coef;
                }
                cpos++;
            }
            if(flag) c.newTerm(termsArray[apos].coef*b.termsArray[bpos].coef, Exp);
        }
    }
```

←Mult函式程式對照

S(p) = c + Sp = 12+2

T(p) = c + Tp = 4+(4n^3+5n^2)

… = O(n^3)

```
    return c;
}
```

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

←Eval函式程式對照

S(p) = c + Sp = 5+1

T(p) = c + Tp = 3+n

f(n+3) = O(n)

- 心得討論:
  - 其中運算子多載是第一次接觸，是在深碗課程時看到的，提高了不少可讀性。
  - 做乘法函式的效能分析、量測前，並不覺得有什麼問題，但分析完發現big-O為n^3。我認為這程式還可以改善，如其中雙層for迴圈中的檢查程式，可以在之前建一個bool陣列，存相對位置的次方項已新增過，以空間換取時間。
  - 上資料結構課程前，寫題目或專案時都不會做效能分析，通常都用邏輯硬寫。到現在效能分析仍算我的弱項，以上面的分析為例，應該有些錯誤。之後撰寫專案時，都會做這項練習，並試著以資料結構實作，而非再以邏輯硬寫。