解題說明(大部分在註解):

按照題目要求做相加 相乘及帶入 x 後的值，以及 term 類別紀錄係數根指數，相乘跟相加可能使陣列變大，所以做一個 newsize*2 倍。

輸入(istream): 3x^3+2x^2+1(舉例)

讀入:3 x(判斷是不是'\0') ^(ignore) 3(指數)+(判斷+，-)2 x(判斷是不是'\0') ^(ignore) 2(指數)+(判斷+，-)+1(判斷是'\0') 結束

輸出(ostream): 3x^3+2^2+1(舉例)

顯示:3(判斷是不是第一個或常數)x^3(exp)+2(判斷是不是第一個或常數)x^2+1

Add，Mult 寫在註解

```
class Term{
    friend polynomial;
    friend ostream& operator<<(ostream& os,const polynomial& poly);
private:
    float coef;  //係數
    int exp;     //指數
};
```

```
class polynomial{
    friend ostream& operator<<(ostream& os,const polynomial& poly);
    friend istream& operator>>(istream& is ,polynomial& poly);

public:
    polynomial(); //建構多項式p(x)0

    polynomial Add(polynomial poly);//相加

    polynomial Mult(polynomial poly);//相乘

    float Eval(float f);//帶入X的數值

    void newSize(const float newcoef,const int newexp);//陣列不夠大時使用
private:
    Term *termArray;// 陣列
    int capacity;// termArray 大小
    int terms;// 非零數字

};
```

```
polynomial::polynomial():capacity(2),terms(0){

    termArray = new Term[capacity];//初始化多項式
}
```

建構子，建立多項式

```
void polynomial::newSize(const float newcoef,const int newexp){
    if(this->terms==this->capacity){//如果當前項數等於陣列容量，代表需要擴充
        this->capacity*=2;//空間不足時*2倍
        Term *temp= new Term[this->capacity];//創一個新的2倍大陣列
        copy(this->termArray,this->termArray+terms,temp);//將舊的存到2倍大的陣列
        delete[] this->termArray;//刪舊的
        this->termArray=temp;//將新的陣列指向termArray
    }
    this->termArray[this->terms].coef=newcoef;//設定係數
    this->termArray[this->terms++].exp=newexp;//設定次方
}
```

建立大小，註解說明

```
//相加
polynomial polynomial::Add(polynomial poly) {
    polynomial result; //用於存儲加法結果
    int *loc; //記錄次方
    loc = new int[poly.terms+this->terms]; //為loc分配兩個多項式項數相加的總和
    float *data;//紀錄係數
    data = new float[poly.terms+this->terms];
    int use_len=0;//紀錄有效長度

    for(int i=0;i<this->terms;i++){//跑目前每一項
        int t=-1;//判斷使否相同次方
        for(int j=0;j<use_len;j++){
            if(this->termArray[i].exp==loc[j]){//如果已存在loc位置
                t=j;//紀錄位置
                continue;
            }
        }
        if(t==-1){//不在loc裡面
            loc[use_len]=this->termArray[i].exp;//新增次方
            data[use_len++]=this->termArray[i].coef;//新增系數
        }
        else
            data[t]+=this->termArray[i].coef;//已存在->相加
    }
    for(int i=0;i<poly.terms;i++){//跑傳入多項式每一項
        int t=-1;
        for(int j=0;j<use_len;j++){
            if(poly.termArray[i].exp==loc[j]){
                t=j;
            continue;
            }
        }

        if(t==-1){
            loc[use_len]= poly.termArray[i].exp;
            data[use_len++]=poly.termArray[i].coef;
        }
        else
            data[t]+=poly.termArray[i].coef;
    }
    for(int i=0;i<use_len;i++){// 將 loc 和 data 中的值加入結果多項式
        result.newSize(data[i],loc[i]);// 新增項目到結果多項式
    }
    return result;
}
```

```
//相乘
polynomial polynomial::Mult(polynomial poly){
    polynomial result;
    int *loc;
    loc=new int[poly.terms*this->terms];//項數相乘總和
    float *data;
    data = new float[poly.terms*this->terms];//系數箱乘總和
    int use_len=0;//有效長度
    for(int i=0;i<this->terms;i++){//跑當前多項是每一個
        for(int j=0;j<poly.terms;j++){//跑輸入多項式每一個
            float t_coef = this->termArray[i].coef*poly.termArray[j].coef;//計算項數相乘
            int t_exp = this->termArray[i].exp+poly.termArray[j].exp;//計算係數相加
            int t=-1;//找細數
            for(int k=0;k<use_len;k++){//跑細數位置
                if(t_exp==loc[k]){ // 如過細數相同
                t=k;//紀錄當前位置
                continue;}
            }
        if(t==-1){//代表該系數裡面沒有值
            loc[use_len]= t_exp;//記錄新的系數
            data[use_len++]= t_coef;
        }
        else
            data[t]+=t_coef;//如果已存在，相加
        }


    }
    for(int i=0;i<use_len;i++)
        result.newSize(data[i],loc[i]);
    return result;
}
```

```
float polynomial::Eval(float x){//帶入x求值
    float result = 0.0f;//初值為0
    for(int i=0;i<this->terms;i++){
        float temp = this->termArray[i].coef;//獲得當前係數
        for(int j=0;j<this->termArray[i].exp;j++){//算x次方
            temp *=x ;//乘以冪次

        }
        result+=temp;//加總
    }
    return result;//回傳
}
```

```
istream& operator>>(istream& is,polynomial& poly){
    float t_coef;
    int t_exp;
    char tmp;
    bool plus = true;
    while(1){
        is>>t_coef; //讀入係數
        if(!plus){
            t_coef*=-1;
            plus = true;
        }
        is.get(tmp);//讀下個字元，可能是'x'`'\0'
        if(tmp=='\n'){//如果是\0，表示常數
            poly.newSize(t_coef,0);//新增指數0的Term
            break;//沒下一個值
        }
        is.ignore(1);//忽略'^'
        is>>t_exp;//讀指數
        poly.newSize(t_coef,t_exp);//新增項目
        is.get(tmp);//讀下一個
        if(tmp== '\n') break;//判斷是不是結尾
        else if(tmp=='-')plus = false;//如果是'-'代表負號
    }
    return is;
}
```

```
ostream& operator<<(ostream& os,const polynomial& poly){
    for(int i=0;i<poly.terms;i++){
        if(poly.termArray[i].exp==0){//如果是常數，只輸出數字
            os<<(poly.termArray[i].coef<0?"":"+")<<poly.termArray[i].coef;
            continue;
        }
        if(i==0)//ax^n
            os << poly.termArray[i].coef << "x^" << poly.termArray[i].exp;
        else//+ax^n
            os<<(poly.termArray[i].coef<0?"":"+")<<poly.termArray[i].coef<<"x^"<<poly.termArray[i].exp;
    }
    return os;
}
```

```
int main(){
    clock_t start,finish;
    cout<<"輸入格式 ax^n2+bx^n1+c (常數不用^n0)"<<endl;
    polynomial p1,p2;
    cout<<"p1: ";
    cin>>p1;
    cout<<"p2: ";
    cin>>p2;
    cout<<endl;
    cout << "Polynomial的Eval()\n";
    float f = 0.0f;
    cout << "請輸入f值：";
    cin >> f;
    cout << "多項式：" << p1 << endl;
    cout << "結果 = " << p1.Eval(f) << endl;
    cout<<endl<<"polynomial的Add()\n";
    cout<<"p1= "<<p1<<endl;
    cout<<"p2= "<<p2<<endl;
    start=clock();
    cout<<"("<<p1<<")+("<<p2<<")="<<p1.Add(p2)<<endl;
    finish=clock();
    cout<<"Add()需時"<<(double)(finish-start)/CLOCKS_PER_SEC<<"s"<<endl<<endl;
    cout<<"polynomial的Mult()\n";
    cout<<"p1= "<<p1<<endl;
    cout<<"p2= "<<p2<<endl;
    start=clock();
    cout<<"("<<p1<<")*("<<p2<<")="<<p1.Mult(p2)<<endl;
    finish=clock();
    cout<<"Mult()需時: "<<(double)(finish-start)/CLOCKS_PER_SEC<<"s"<<endl;
    return 0;
}
```

總程式

```cpp
#include<iostream>
#include<cmath>
#include<string>
#include<ctime>
#include<exception>
using namespace std;
class polynomial;


class Term{
    friend polynomial;
    friend ostream& operator<<(ostream& os,const polynomial& poly);
private:
    float coef;  //係數
    int exp;    //指數
};


class polynomial{
    friend ostream& operator<<(ostream& os,const polynomial& poly);
    friend istream& operator>>(istream& is ,polynomial& poly);

public:
    polynomial(); //建構多項式p(x)0

    polynomial Add(polynomial poly);//相加

    polynomial Mult(polynomial poly);//相乘

    float Eval(float f);//帶入X的數值

    void newSize(const float newcoef,const int newexp);//陣列不夠大時使用
private:
    Term *termArray;// 陣列
    int capacity;// termArray 大小
    int terms;// 非零數字

};



polynomial::polynomial():capacity(2),terms(0){

    termArray = new Term[capacity];//初始化多項式
}
```

```
void polynomial::newSize(const float newcoef,const int newexp){
    if(this->terms==this->capacity){//如果當前項數等於陣列容量，代表需要擴充
        this->capacity*=2;//空間不足時*2倍
        Term *temp= new Term[this->capacity];//創一個新的2倍大陣列
        copy(this->termArray,this->termArray+terms,temp);//將舊的存到2倍大的陣列
        delete[] this->termArray;//刪舊的
        this->termArray=temp;//將新的陣列指向termArray
    }
    this->termArray[this->terms].coef=newcoef;//設定係數
    this->termArray[this->terms++].exp=newexp;//設定次方
}


//相加
polynomial polynomial::Add(polynomial poly) {
    polynomial result; //用於存儲加法結果
    int *loc; //記錄次方
    loc = new int[poly.terms+this->terms]; //為loc分配兩個多項式項數相加的總和
    float *data;//紀錄係數
    data = new float[poly.terms+this->terms];
    int use_len=0;//紀錄有效長度

    for(int i=0;i<this->terms;i++){//跑目前每一項
        int t=-1;//判斷使否相同次方
        for(int j=0;j<use_len;j++){
            if(this->termArray[i].exp==loc[j]){//如果已存在loc位置
                t=j;//紀錄位置
                continue;
            }
        }
        if(t==-1){//不在loc裡面
            loc[use_len]=this->termArray[i].exp;//新增次方
            data[use_len++]=this->termArray[i].coef;//新增系數
        }
        else
            data[t]+=this->termArray[i].coef;//已存在->相加
    }
    for(int i=0;i<poly.terms;i++){//跑傳入多項式每一項
        int t=-1;
        for(int j=0;j<use_len;j++){
            if(poly.termArray[i].exp==loc[j]){
                t=j;
            continue;
```

```
            }
        }

        if(t==-1){
            loc[use_len]= poly.termArray[i].exp;
            data[use_len++]=poly.termArray[i].coef;
        }
        else
            data[t]+=poly.termArray[i].coef;
    }
    for(int i=0;i<use_len;i++){// 將 loc 和 data 中的值加入結果多項式
        result.newSize(data[i],loc[i]);// 新增項目到結果多項式
    }
    return result;
}

//相乘
polynomial polynomial::Mult(polynomial poly){
    polynomial result;
    int *loc;
    loc=new int[poly.terms*this->terms];//項數相乘總和
    float *data;
    data = new float[poly.terms*this->terms];//系數箱乘總和
    int use_len=0;//有效長度
    for(int i=0;i<this->terms;i++){//跑當前多項是每一個
        for(int j=0;j<poly.terms;j++){//跑輸入多項式每一個
            float t_coef = this->termArray[i].coef*poly.termArray[j].coef;//計算項數相乘
            int t_exp = this->termArray[i].exp+poly.termArray[j].exp;//計算係數相加
            int t=-1;//找細數
            for(int k=0;k<use_len;k++){//跑細數位置
                if(t_exp==loc[k]){ // 如過細數相同
                t=k;//紀錄當前位置
                continue;}
            }
        if(t==-1){//代表該系數裡面沒有值
            loc[use_len]= t_exp;//記錄新的系數
            data[use_len++]= t_coef;
        }
        else
            data[t]+=t_coef;//如果已存在，相加
        }
```

```
    }
    for(int i=0;i<use_len;i++)
        result.newSize(data[i],loc[i]);
    return result;
}


float polynomial::Eval(float x){//帶入x求值
    float result = 0.0f;//初值為0
    for(int i=0;i<this->terms;i++){
        float temp = this->termArray[i].coef;//獲得當前係數
        for(int j=0;j<this->termArray[i].exp;j++){//算x次方
            temp *=x ;//乘以冪次

        }
        result+=temp;//加總
    }
    return result;//回傳
}

istream& operator>>(istream& is,polynomial& poly){
    float t_coef;
    int t_exp;
    char tmp;
    bool plus = true;
    while(1){
        is>>t_coef; //讀入係數
        if(!plus){
            t_coef*=-1;
            plus = true;
        }
        is.get(tmp);//讀下個字元，可能是'x'``'\0'
        if(tmp=='\n'){//如果是\0，表示常數
            poly.newSize(t_coef,0);//新增指數0的Term
            break;//沒下一個值
        }
        is.ignore(1);//忽略'^'
        is>>t_exp;//讀指數
        poly.newSize(t_coef,t_exp);//新增項目
        is.get(tmp);//讀下一個
        if(tmp== '\n') break;//判斷是不是結尾
        else if(tmp=='-')plus = false;//如果是'-'代表負號
    }
    return is;
}
```

```cpp
ostream& operator<<(ostream& os,const polynomial& poly){
    for(int i=0;i<poly.terms;i++){
        if(poly.termArray[i].exp==0){//如果是常數，只輸出數字
            os<<(poly.termArray[i].coef<0?"":"+")<<poly.termArray[i].coef;
            continue;
        }
        if(i==0)//ax^n
            os << poly.termArray[i].coef << "x^" << poly.termArray[i].exp;
        else//+ax^n
            os<<(poly.termArray[i].coef<0?"":"+")<<poly.termArray[i].coef<<"x^"<<poly.termArray[i].exp;
    }
    return os;
}


int main(){
    clock_t start,finish;
    cout<<"輸入格式 ax^n2+bx^n1+c (常數不用^n0)"<<endl;
    polynomial p1,p2;
    cout<<"p1: ";
    cin>>p1;
    cout<<"p2: ";
    cin>>p2;
    cout<<endl;
    cout << "Polynomial的Eval()\n";
    float f = 0.0f;
    cout << "請輸入f值：";
    cin >> f;
    cout << "多項式：" << p1 << endl;
    cout << "結果 = " << p1.Eval(f) << endl;
    cout<<endl<<"polynomial的Add()\n";
    cout<<"p1= "<<p1<<endl;
    cout<<"p2= "<<p2<<endl;
    start=clock();
    cout<<"("<<p1<<")+("<<p2<<")="<<p1.Add(p2)<<endl;
    finish=clock();
    cout<<"Add()需時"<<(double)(finish-start)/CLOCKS_PER_SEC<<"s"<<endl<<endl;
    cout<<"polynomial的Mult()\n";
    cout<<"p1= "<<p1<<endl;
    cout<<"p2= "<<p2<<endl;
    start=clock();
    cout<<"("<<p1<<")*("<<p2<<")="<<p1.Mult(p2)<<endl;
    finish=clock();
    cout<<"Mult()需時: "<<(double)(finish-start)/CLOCKS_PER_SEC<<"s"<<endl;

    return 0;
}
```

效能分析:

這個 Add()函式時間複雜度為 O(n+m)；空間複雜度為 O(4(n+m)+12)。

Mult()函式時間複雜度是 O(n*m)；空間複雜度是 O(2(n+m+nm)+13)。

測試與驗證:

F:\polynomial.exe

```
輸入格式 ax^n2+bx^n1+c (常數不用^n0)
p1: 3x^3+5x^1+1
p2: 4x^2+5

Polynomial的Eval()
請輸入f值：2
多項式：3x^3+5x^1+1
結果 = 35

polynomial的Add()
p1= 3x^3+5x^1+1
p2= 4x^2+5
(3x^3+5x^1+1)+(4x^2+5)=3x^3+5x^1+6+4x^2
Add()需時0.003s

polynomial的Mult()
p1= 3x^3+5x^1+1
p2= 4x^2+5
(3x^3+5x^1+1)*(4x^2+5)=12x^5+35x^3+25x^1+4x^2+5
Mult()需時: 0.003s
```

相加 耗時 0.003 秒

$$\begin{array}{r} 3x^3 + 5x + 1 \\ + \quad 4x^2 + 5 \\ \hline 3x^3 + 4x^2 + 5x + 6 \end{array}$$

F=2

$$f=2$$

$$3 \cdot 2^3 + 5 \cdot 2 + 1 = 24 + 10 + 1 = 35$$

相乘 耗時 0.003 秒

$$(3x^3+5x+1)(4x^2+5)$$
$$= 12x^5+15x^3+20x^3+25x+4x^2+5$$
$$= 12x^5+35x^3+4x^2+25x+5$$

心得:

這次是做多項式類別的計算，讓我更加深對於物件導向的觀念，還擴加矩陣也用到資料結構上學的根據長度增加 2 倍大的矩陣，在節省時間的情況下盡量做到不浪費記憶體空間，這份作業也讓我花費大量的時間在尋找資料及實作，實不相瞞，我光是 DeBug 就做了好幾個小時，做完只覺得暫時不想看到電腦了。