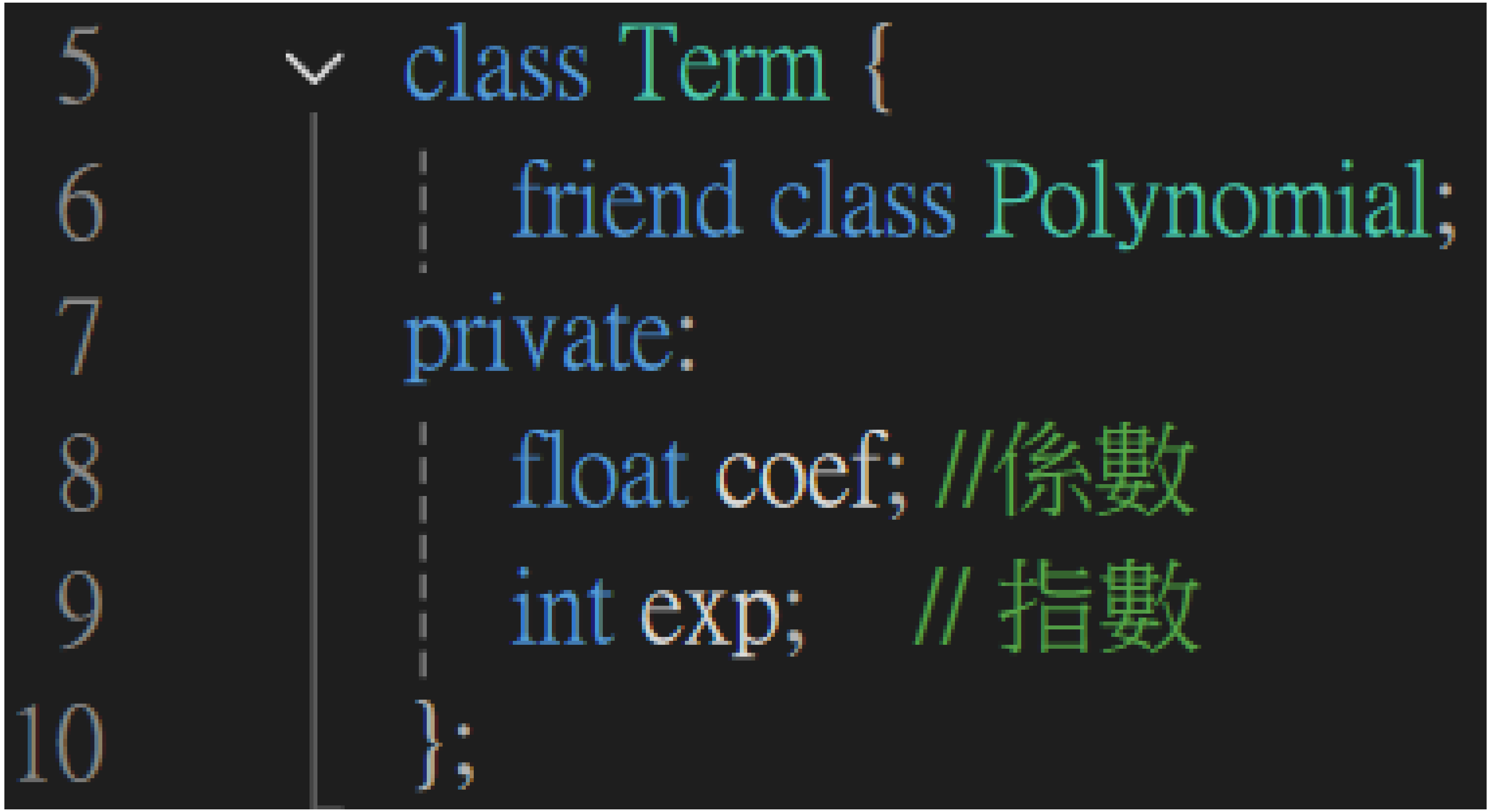
HW02多項式

11227105羅冠穎

想法

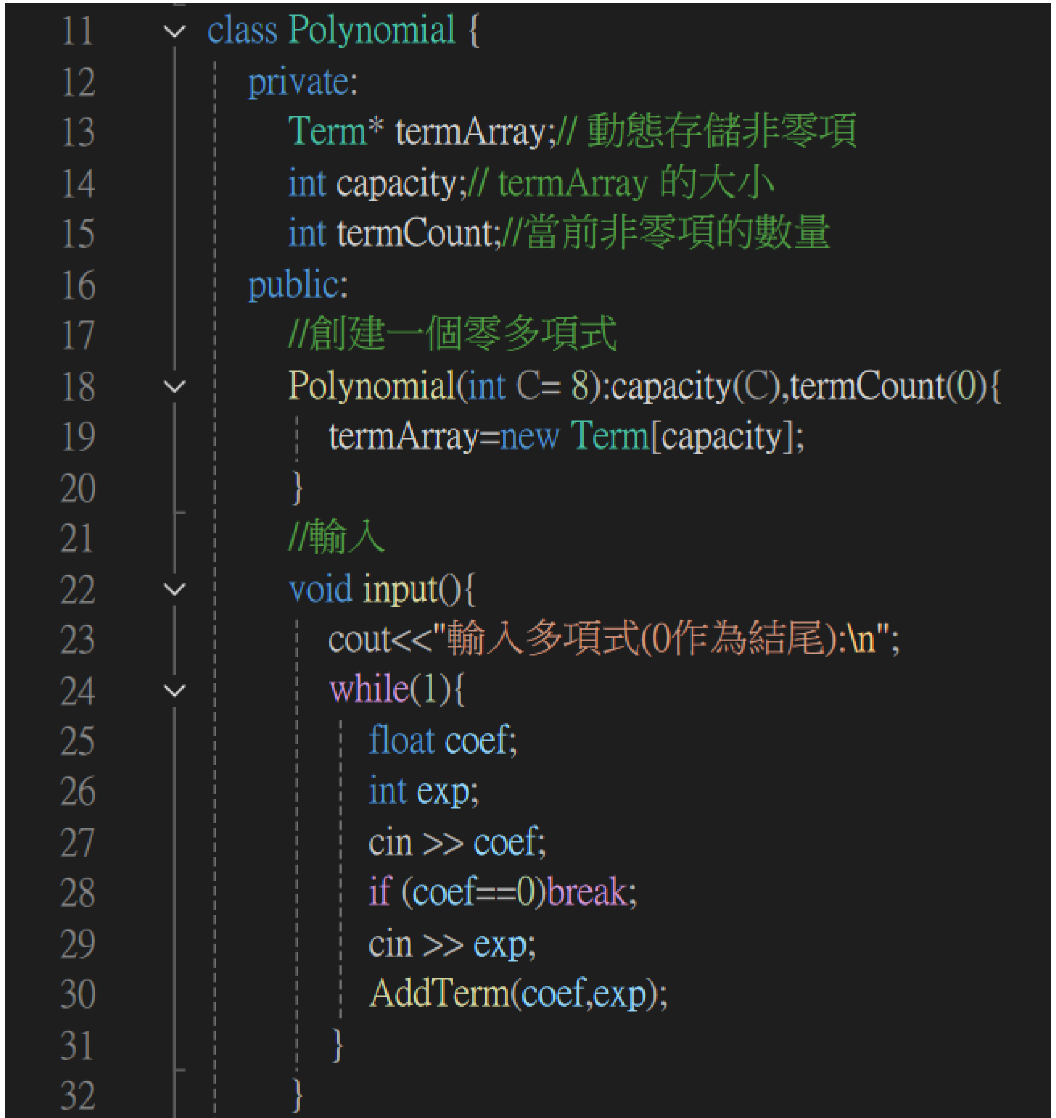
邊翻書邊寫,走一步算一步

設計&實作:Term



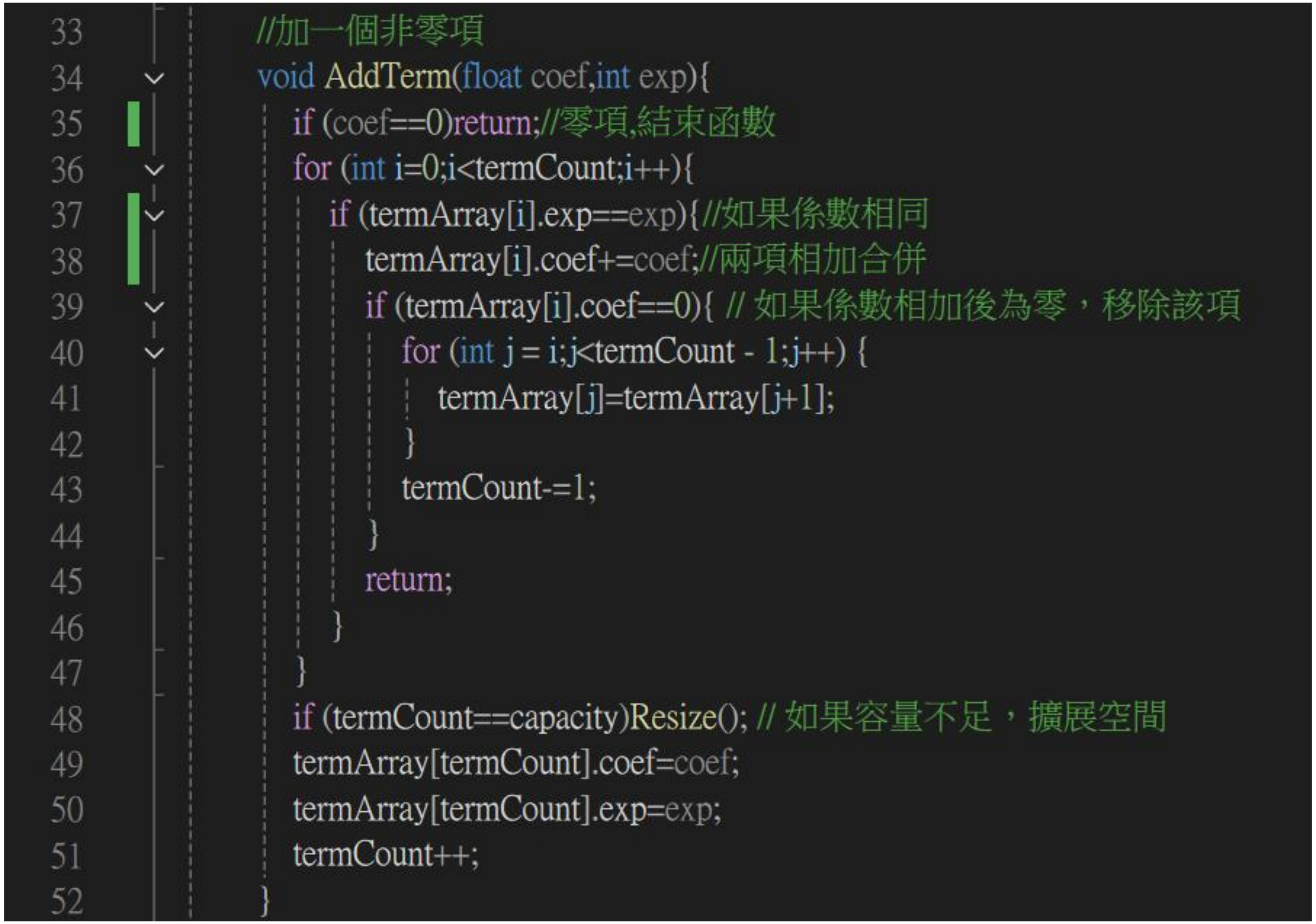
```
class Term {
    friend class Polynomial;
private:
    float coef; //係數
    int exp;    // 指數
};
```

設計&實作:polynomial_01



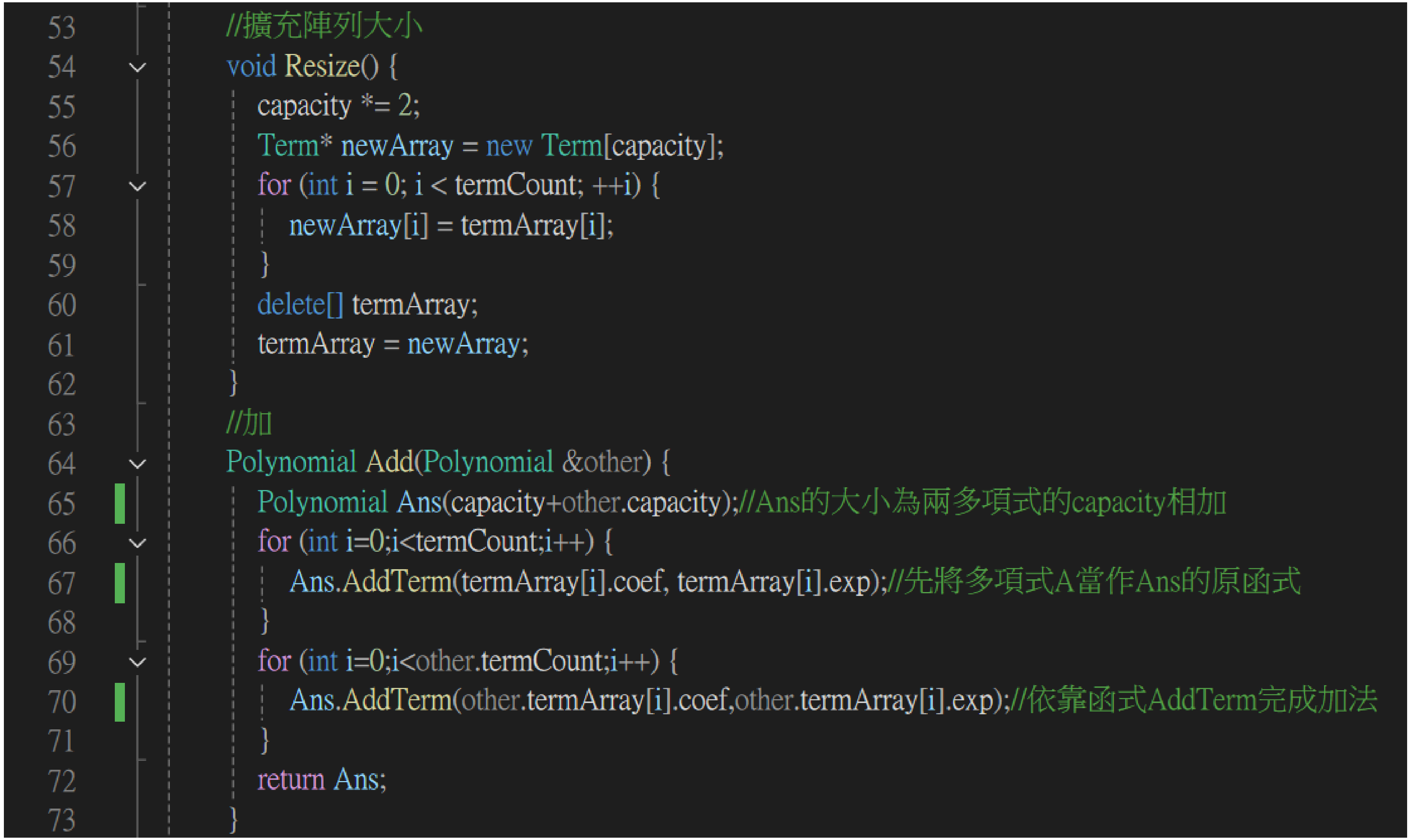
```cpp
class Polynomial {
    private:
        Term* termArray;// 動態存儲非零項
        int capacity;// termArray 的大小
        int termCount;//當前非零項的數量
    public:
        //創建一個零多項式
        Polynomial(int C= 8):capacity(C),termCount(0){
            termArray=new Term[capacity];
        }
        //輸入
        void input(){
            cout<<"輸入多項式(0作為結尾):\n";
            while(1){
                float coef;
                int exp;
                cin >> coef;
                if (coef==0)break;
                cin >> exp;
                AddTerm(coef,exp);
            }
        }
```

設計&實作:polynomial_02



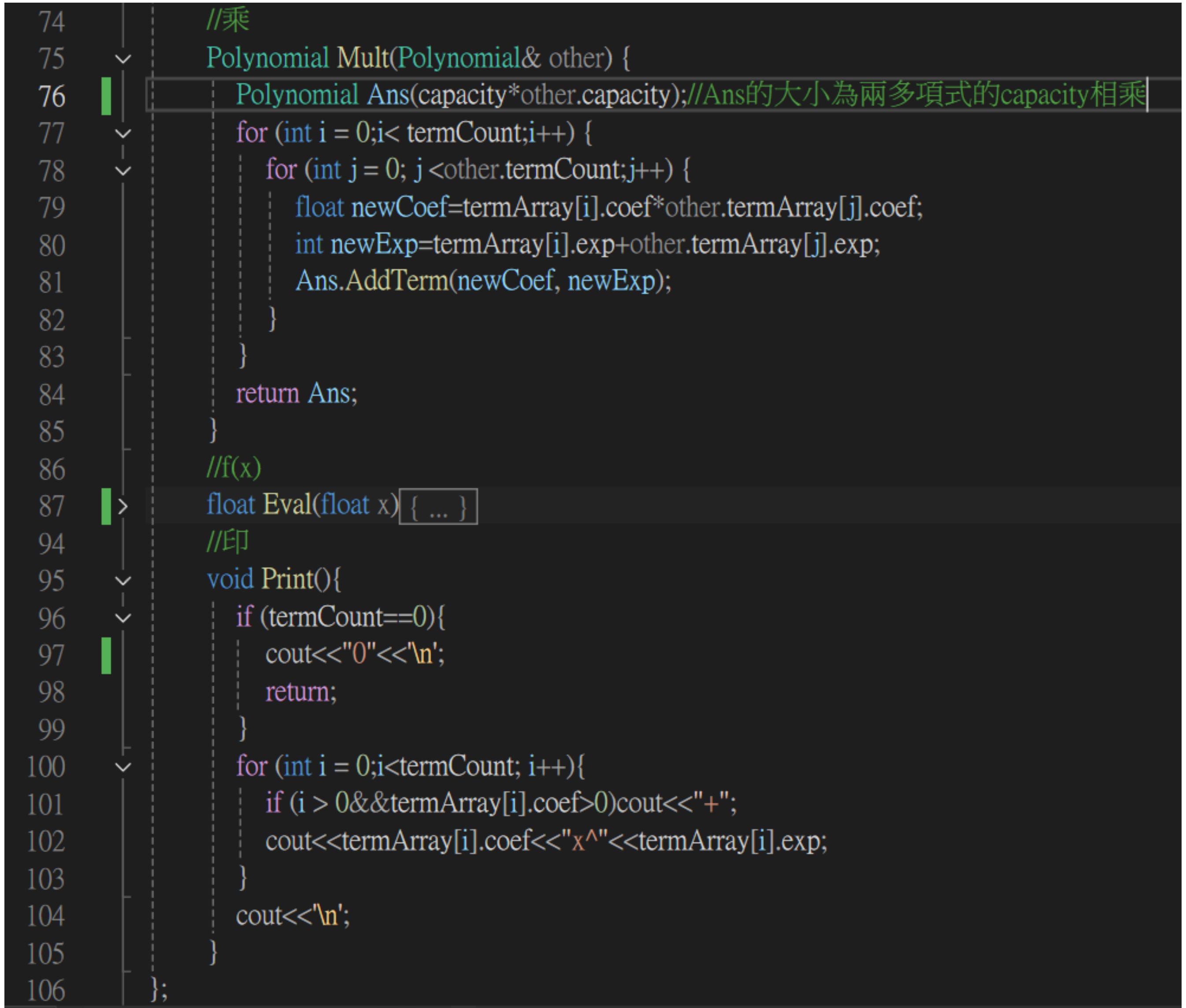
```
//加一個非零項
void AddTerm(float coef,int exp){
    if (coef==0)return;//零項,結束函數
    for (int i=0;i<termCount;i++){
        if (termArray[i].exp==exp){//如果係數相同
            termArray[i].coef+=coef;//兩項相加合併
            if (termArray[i].coef==0){ // 如果係數相加後為零，移除該項
                for (int j = i;j<termCount - 1;j++) {
                    termArray[j]=termArray[j+1];
                }
                termCount-=1;
            }
            return;
        }
    }
    if (termCount==capacity)Resize(); // 如果容量不足，擴展空間
    termArray[termCount].coef=coef;
    termArray[termCount].exp=exp;
    termCount++;
}
```

設計&實作:polynomial_03



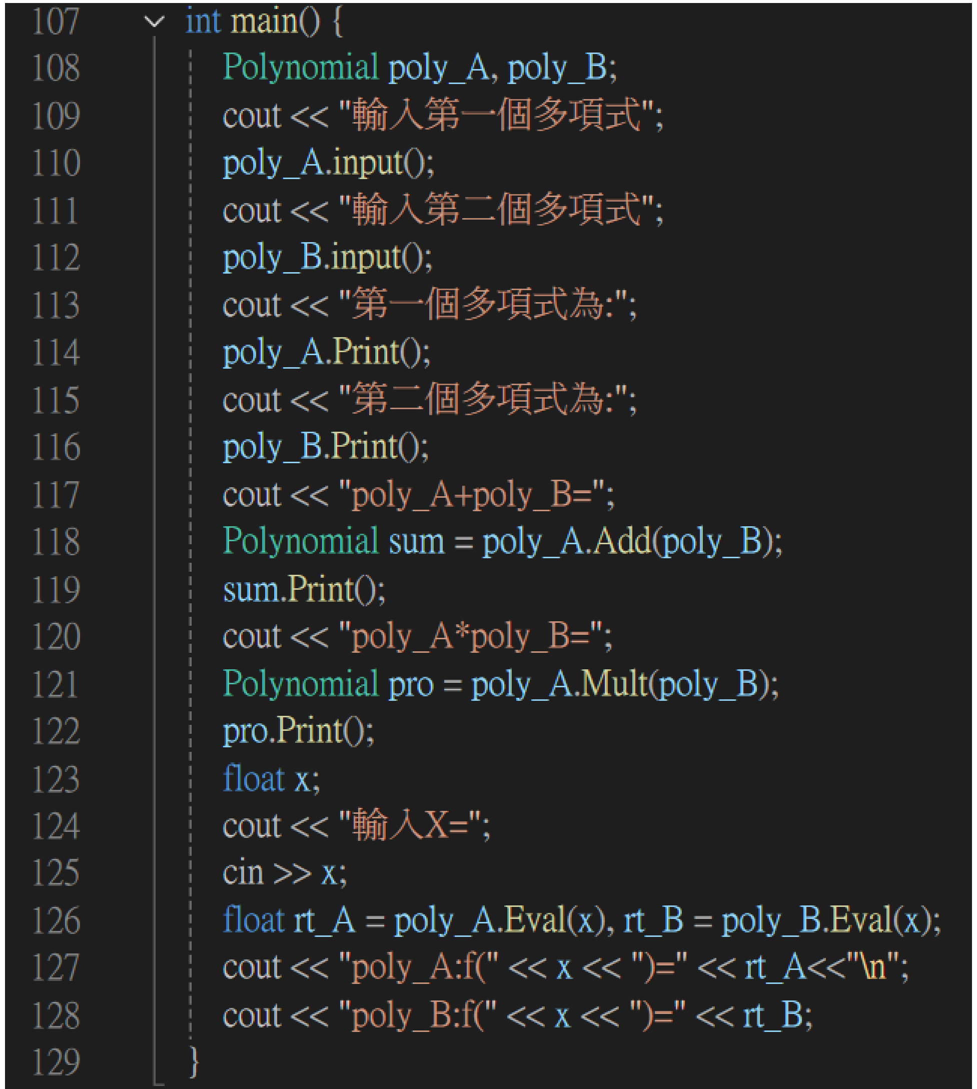
```
    //擴充陣列大小
    void Resize() {
        capacity *= 2;
        Term* newArray = new Term[capacity];
        for (int i = 0; i < termCount; ++i) {
            newArray[i] = termArray[i];
        }
        delete[] termArray;
        termArray = newArray;
    }
    //加
    Polynomial Add(Polynomial &other) {
        Polynomial Ans(capacity+other.capacity);//Ans的大小為兩多項式的capacity相加
        for (int i=0;i<termCount;i++) {
            Ans.AddTerm(termArray[i].coef, termArray[i].exp);//先將多項式A當作Ans的原函式
        }
        for (int i=0;i<other.termCount;i++) {
            Ans.AddTerm(other.termArray[i].coef,other.termArray[i].exp);//依靠函式AddTerm完成加法
        }
        return Ans;
    }
```

設計&實作:polynomial_04



```
    //乘
    Polynomial Mult(Polynomial& other) {
        Polynomial Ans(capacity*other.capacity);//Ans的大小為兩多項式的capacity相乘
        for (int i = 0;i< termCount;i++) {
            for (int j = 0; j <other.termCount;j++) {
                float newCoef=termArray[i].coef*other.termArray[j].coef;
                int newExp=termArray[i].exp+other.termArray[j].exp;
                Ans.AddTerm(newCoef, newExp);
            }
        }
        return Ans;
    }
    //f(x)
    float Eval(float x){ ... }
    //印
    void Print(){
        if (termCount==0){
            cout<<"0"<<'\n';
            return;
        }
        for (int i = 0;i<termCount; i++){
            if (i > 0&&termArray[i].coef>0)cout<<"+";
            cout<<termArray[i].coef<<"x^"<<termArray[i].exp;
        }
        cout<<'\n';
    }
};
```

設計&實作:main



```
int main() {
    Polynomial poly_A, poly_B;
    cout << "輸入第一個多項式";
    poly_A.input();
    cout << "輸入第二個多項式";
    poly_B.input();
    cout << "第一個多項式為:";
    poly_A.Print();
    cout << "第二個多項式為:";
    poly_B.Print();
    cout << "poly_A+poly_B=";
    Polynomial sum = poly_A.Add(poly_B);
    sum.Print();
    cout << "poly_A*poly_B=";
    Polynomial pro = poly_A.Mult(poly_B);
    pro.Print();
    float x;
    cout << "輸入X=";
    cin >> x;
    float rt_A = poly_A.Eval(x), rt_B = poly_B.Eval(x);
    cout << "poly_A:f(" << x << ")=" << rt_A<<"\n";
    cout << "poly_B:f(" << x << ")=" << rt_B;
}
```

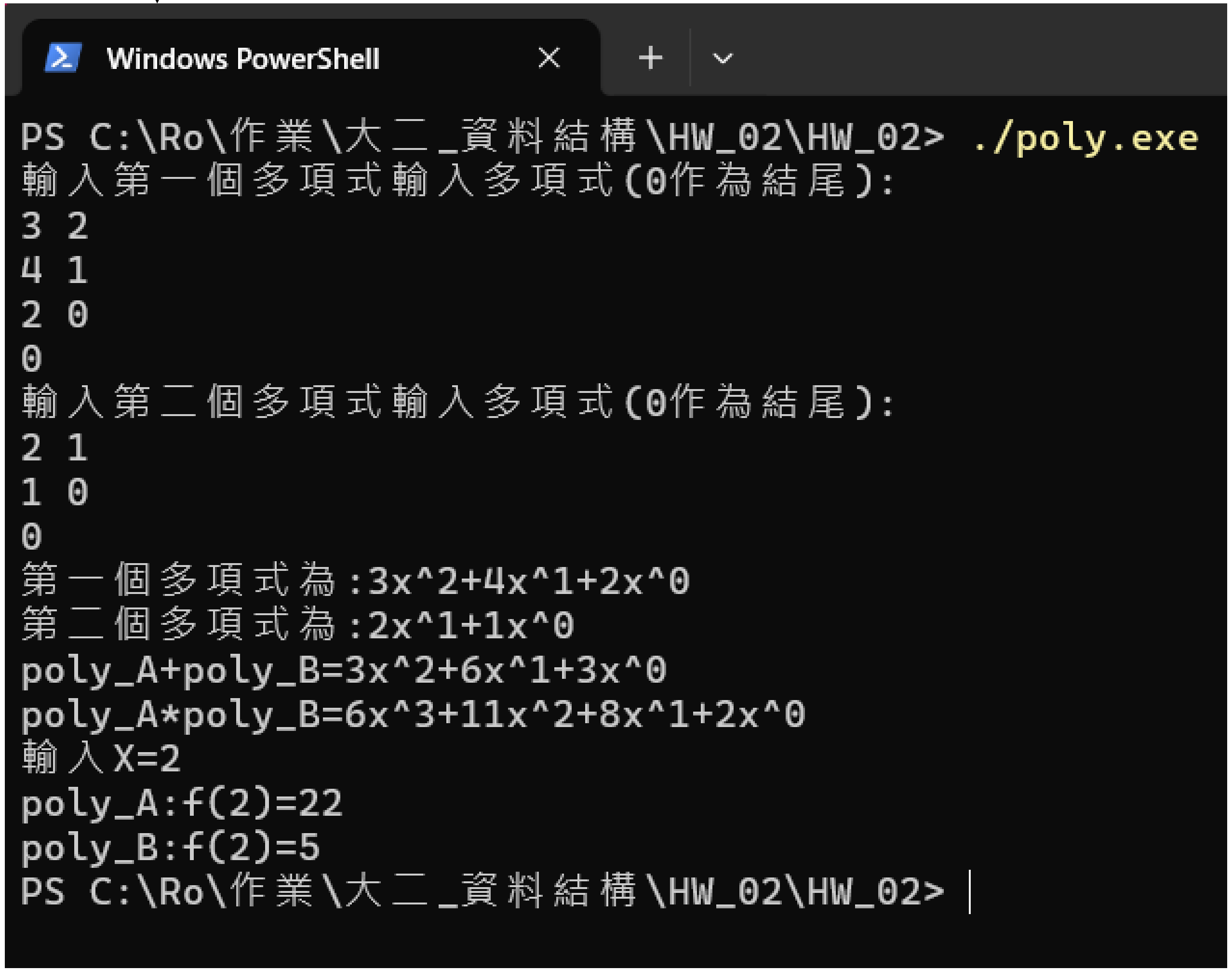
效能分析_時間複雜度

- Add():
  - 遍歷兩個多項式的所有非零項，各執行一次AddTerm。對於 A 和 B 的非零項數量分別為n1 和 2，時間複雜度為：
  - O(n1*n1)+O(n2*(n1+n2))=O((n1^2)+n1*n2+n2^2)
- Mult():
  - 對於A和B的非零項，兩兩相乘後執行 AddTerm。n1 和n 2為A和B的非零項數，總時間複雜度為：
  - O(n1*n2*(n1*n2))=O((n1^2)*(n2^2))
- Eval():
  - 對於每個非零項，計算 coef*X^exp，時間複雜度為O(n) ，n是非零項數量

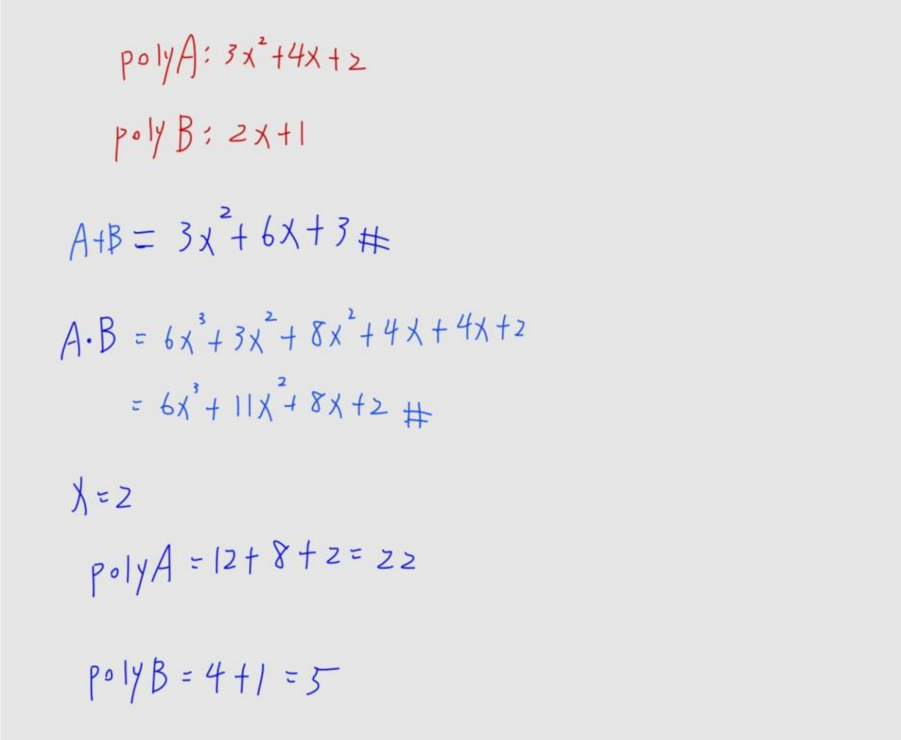
效能分析_空間複雜度

- Add():O(n1+n2)
- Mult():O(n1*n2)
- Eval():O(n),n為非零項數量

測試&驗證



```
PS C:\Ro\作業\大二_資料結構\HW_02\HW_02> ./poly.exe
輸入第一個多項式輸入多項式(0作為結尾):
3 2
4 1
2 0
0
輸入第二個多項式輸入多項式(0作為結尾):
2 1
1 0
0
第一個多項式為:3x^2+4x^1+2x^0
第二個多項式為:2x^1+1x^0
poly_A+poly_B=3x^2+6x^1+3x^0
poly_A*poly_B=6x^3+11x^2+8x^1+2x^0
輸入X=2
poly_A:f(2)=22
poly_B:f(2)=5
PS C:\Ro\作業\大二_資料結構\HW_02\HW_02>
```



polyA: $3x^2+4x+2$

polyB: $2x+1$

$A+B = 3x^2+6x+3$ #

$A\cdot B = 6x^3+3x^2+8x^2+4x+4x+2$

$= 6x^3+11x^2+8x+2$ #

$x=2$

polyA $=12+8+2=22$

polyB $=4+1=5$

心得

在原本的學校老師上課考試全都是同一個模板,全部成員放public,所以這算是我第一次真正的接觸到物件導向,這個作業可以說是不斷的在“翻書>coding>chat gpt debug”中不斷loop,永遠寫不完,當下我感覺自己和那個推巨石的老兄差不多