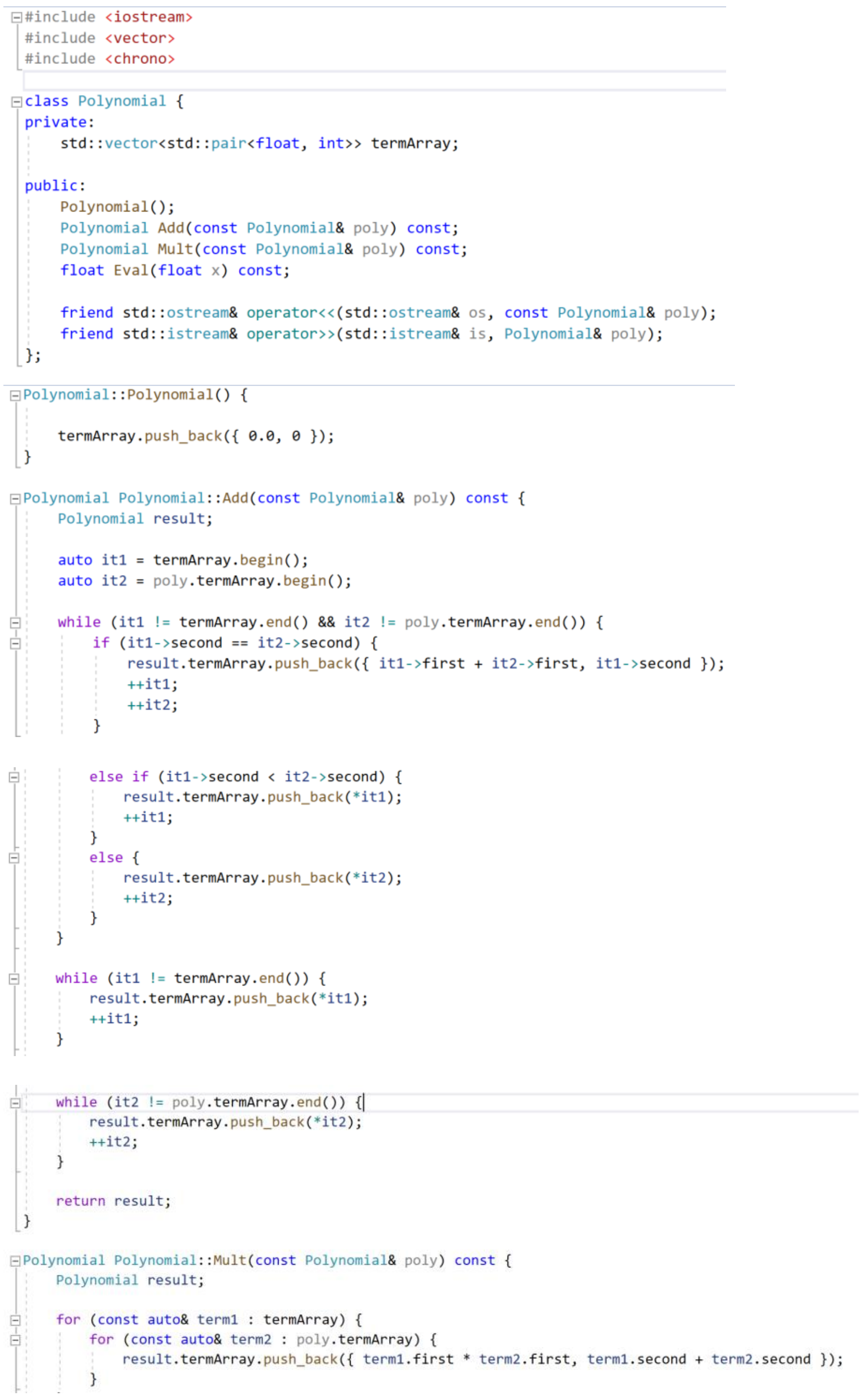
資工二乙 41143229 張大軒 HW2

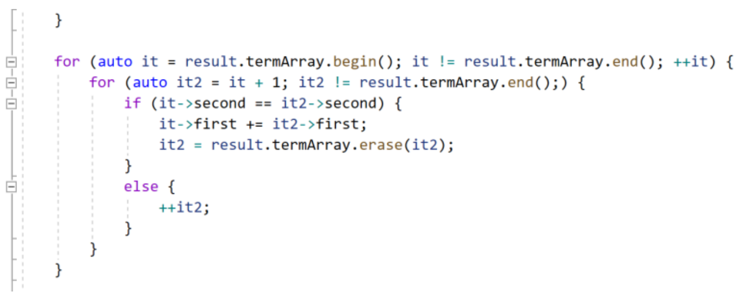
解題說明:

程式碼定義了一個多項式的類別 Polynomial，並提供了相關的操作和功能。這個類別允許創建多項式物件，並且可以進行多項式的加法、乘法以及多項式在給定點的求值。

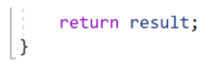
Algorithm Design & Programming:



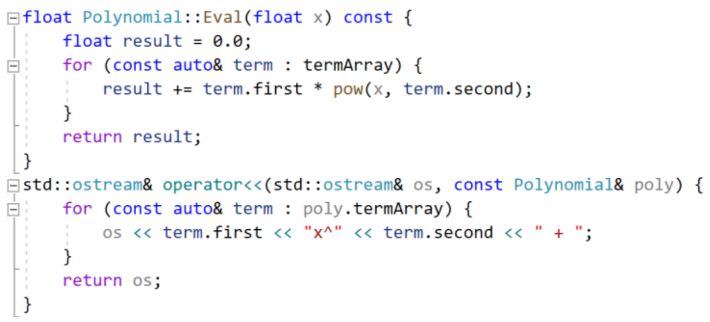
```
#include <iostream>
#include <vector>
#include <chrono>

class Polynomial {
private:
    std::vector<std::pair<float, int>> termArray;

public:
    Polynomial();
    Polynomial Add(const Polynomial& poly) const;
    Polynomial Mult(const Polynomial& poly) const;
    float Eval(float x) const;

    friend std::ostream& operator<<(std::ostream& os, const Polynomial& poly);
    friend std::istream& operator>>(std::istream& is, Polynomial& poly);
};

Polynomial::Polynomial() {

    termArray.push_back({ 0.0, 0 });
}

Polynomial Polynomial::Add(const Polynomial& poly) const {
    Polynomial result;

    auto it1 = termArray.begin();
    auto it2 = poly.termArray.begin();

    while (it1 != termArray.end() && it2 != poly.termArray.end()) {
        if (it1->second == it2->second) {
            result.termArray.push_back({ it1->first + it2->first, it1->second });
            ++it1;
            ++it2;
        }

        else if (it1->second < it2->second) {
            result.termArray.push_back(*it1);
            ++it1;
        }
        else {
            result.termArray.push_back(*it2);
            ++it2;
        }
    }

    while (it1 != termArray.end()) {
        result.termArray.push_back(*it1);
        ++it1;
    }

    while (it2 != poly.termArray.end()) {
        result.termArray.push_back(*it2);
        ++it2;
    }

    return result;
}

Polynomial Polynomial::Mult(const Polynomial& poly) const {
    Polynomial result;

    for (const auto& term1 : termArray) {
        for (const auto& term2 : poly.termArray) {
            result.termArray.push_back({ term1.first * term2.first, term1.second + term2.second });
        }
```



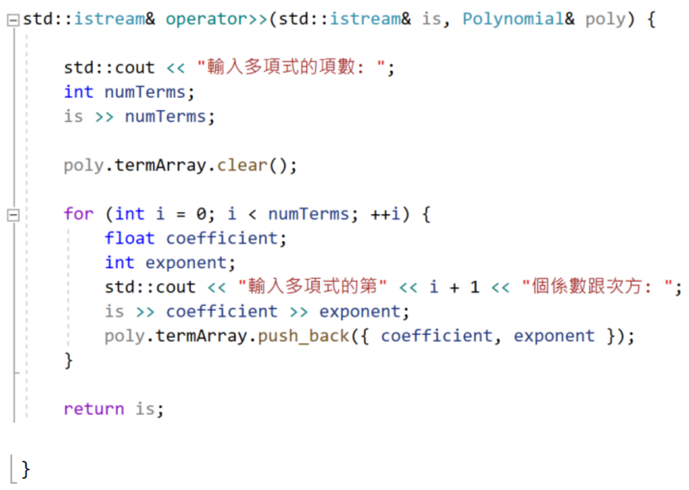
```
    }

    for (auto it = result.termArray.begin(); it != result.termArray.end(); ++it) {
        for (auto it2 = it + 1; it2 != result.termArray.end();) {
            if (it->second == it2->second) {
                it->first += it2->first;
                it2 = result.termArray.erase(it2);
            }
            else {
                ++it2;
            }
        }
    }
```



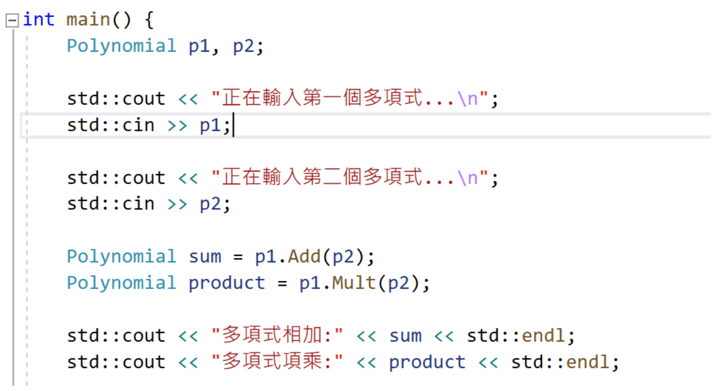
```
    return result;
}
```



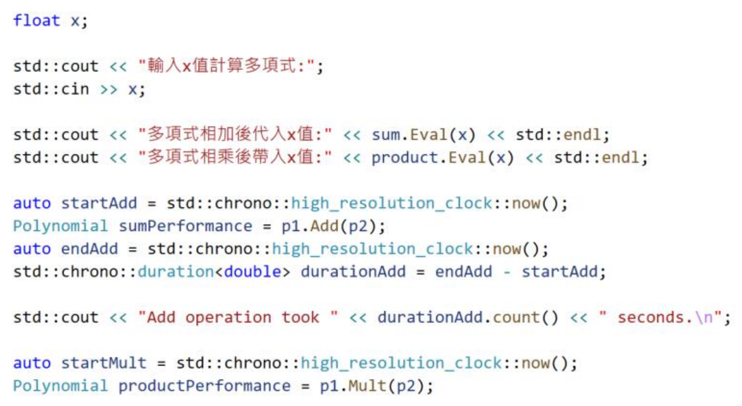
```
float Polynomial::Eval(float x) const {
    float result = 0.0;
    for (const auto& term : termArray) {
        result += term.first * pow(x, term.second);
    }
    return result;
}
std::ostream& operator<<(std::ostream& os, const Polynomial& poly) {
    for (const auto& term : poly.termArray) {
        os << term.first << "x^" << term.second << " + ";
    }
    return os;
}
```



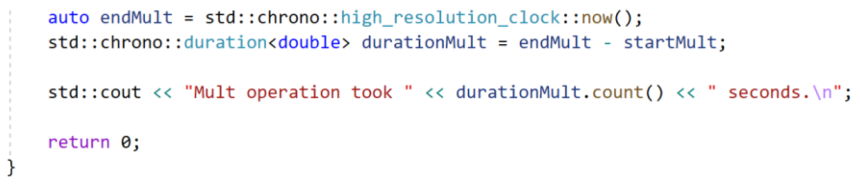
```
std::istream& operator>>(std::istream& is, Polynomial& poly) {

    std::cout << "輸入多項式的項數: ";
    int numTerms;
    is >> numTerms;

    poly.termArray.clear();

    for (int i = 0; i < numTerms; ++i) {
        float coefficient;
        int exponent;
        std::cout << "輸入多項式的第" << i + 1 << "個係數跟次方: ";
        is >> coefficient >> exponent;
        poly.termArray.push_back({ coefficient, exponent });
    }

    return is;

}
```



```
int main() {
    Polynomial p1, p2;

    std::cout << "正在輸入第一個多項式...\n";
    std::cin >> p1;

    std::cout << "正在輸入第二個多項式...\n";
    std::cin >> p2;

    Polynomial sum = p1.Add(p2);
    Polynomial product = p1.Mult(p2);

    std::cout << "多項式相加:" << sum << std::endl;
    std::cout << "多項式項乘:" << product << std::endl;
```



```
    float x;

    std::cout << "輸入x值計算多項式:";
    std::cin >> x;

    std::cout << "多項式相加後代入x值:" << sum.Eval(x) << std::endl;
    std::cout << "多項式相乘後帶入x值:" << product.Eval(x) << std::endl;

    auto startAdd = std::chrono::high_resolution_clock::now();
    Polynomial sumPerformance = p1.Add(p2);
    auto endAdd = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> durationAdd = endAdd - startAdd;

    std::cout << "Add operation took " << durationAdd.count() << " seconds.\n";

    auto startMult = std::chrono::high_resolution_clock::now();
    Polynomial productPerformance = p1.Mult(p2);
```



```
    auto endMult = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> durationMult = endMult - startMult;

    std::cout << "Mult operation took " << durationMult.count() << " seconds.\n";

    return 0;
}
```

效能分析:

加法(‘Add’)

時間複雜度:O(n+m)

空間複雜度: O(n+m)

乘法(’Mult’)

時間複雜度:O(n*m)

空間複雜度: O(n*m)

求值(‘Eval’)

時間複雜度:O(n)

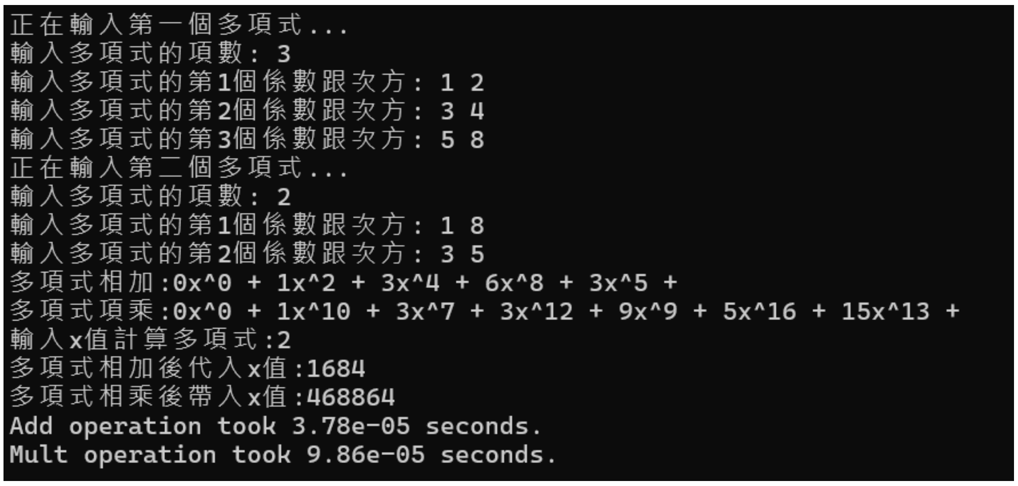
空間複雜度: O(1)

輸入(‘operator>>’)和輸出(‘operator<<’)

時間複雜度:O(n)

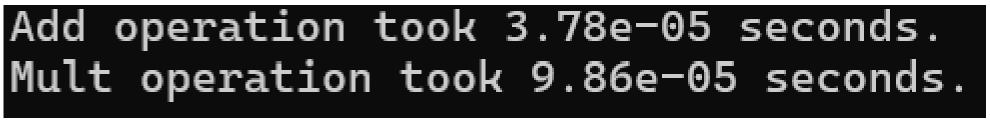
空間複雜度: O(n)

測試與驗證:



```
正在輸入第一個多項式...
輸入多項式的項數: 3
輸入多項式的第1個係數跟次方: 1 2
輸入多項式的第2個係數跟次方: 3 4
輸入多項式的第3個係數跟次方: 5 8
正在輸入第二個多項式...
輸入多項式的項數: 2
輸入多項式的第1個係數跟次方: 1 8
輸入多項式的第2個係數跟次方: 3 5
多項式相加:0x^0 + 1x^2 + 3x^4 + 6x^8 + 3x^5 +
多項式項乘:0x^0 + 1x^10 + 3x^7 + 3x^12 + 9x^9 + 5x^16 + 15x^13 +
輸入x值計算多項式:2
多項式相加後代入x值:1684
多項式相乘後帶入x值:468864
Add operation took 3.78e-05 seconds.
Mult operation took 9.86e-05 seconds.
```

效能量測:



```
Add operation took 3.78e-05 seconds.
Mult operation took 9.86e-05 seconds.
```

心得:

這次的作業，我有藉由 Chat GPT 的幫忙來完成，上課聽老師分析程式碼的細節與功用，回家上網查詢以及整理資料，一點一滴完成這項作業，讓我有了成就感，不會再對程式感到那麼畏懼及排斥。