

解題說明：

成員數據結構：

多項式的項目（**term**）使用

`std::vector<std::pair<float, int>> termArray` 表示，其中每對 `<float, int>` 分別表示項目的係數和指數。

建構函數：

建構函數初始化多項式為  $p(x) = 0$ 。

加法函數 **Add**：

兩個多項式相加時，我們遍歷它們的項目。如果兩項的指數相同，我們將它們的係數相加。如果指數不同，我們將具有更小指數的項添加到結果中。

乘法函數 **Mult**：

乘法涉及將兩個多項式的每一對項相乘，並將結果的項相加。我們使用嵌套循環遍歷兩個多項式的所有項目，計算它們的乘積，然後將結果添加到結果多項式中。

評估函數 **Eval**：

評估函數通過將給定的  $x$  值代入多項式，計算每一項的值，然後將所有項的值相加。

輸入和輸出運算符重載：

`operator>>` 從用戶獲取多項式的項數和每個項的係數和指數。

`operator<<` 將多項式的每一項以 "`coef * x^exp`" 的形式輸出。

Algorithm Design & Programming：

```
1  #include <iostream>
2  #include <vector>
3  #include <chrono>
4
5  class Polynomial {
6  private:
7      std::vector<std::pair<float, int>> termArray;
8
9  public:
10     Polynomial();
11     Polynomial Add(const Polynomial& poly) const;
12     Polynomial Mult(const Polynomial& poly) const;
13     float Eval(float x) const;
14
15     friend std::ostream& operator<<(std::ostream& os, const Polynomial& poly);
16     friend std::istream& operator>>(std::istream& is, Polynomial& poly);
17 };
18
19 Polynomial::Polynomial() {
20
21     termArray.push_back({ 0.0, 0 });
22 }
23
24 Polynomial Polynomial::Add(const Polynomial& poly) const {
25     Polynomial result;
26
27     auto it1 = termArray.begin();
28     auto it2 = poly.termArray.begin();
29
30     while (it1 != termArray.end() && it2 != poly.termArray.end()) {
31         if (it1->second == it2->second) {
32             result.termArray.push_back({ it1->first + it2->first, it1->second });
33             ++it1;
34             ++it2;
35         }
36         else if (it1->second < it2->second) {
37             result.termArray.push_back(*it1);
38             ++it1;
39         }
40         else {
41             result.termArray.push_back(*it2);
42             ++it2;
43         }
44     }
45
46     while (it1 != termArray.end()) {
47         result.termArray.push_back(*it1);
48         ++it1;
49     }
50
51     while (it2 != poly.termArray.end()) {
52         result.termArray.push_back(*it2);
```

```

53     ++it2;
54 }
55
56     return result;
57 }
58
59 Polynomial Polynomial::Mult(const Polynomial& poly) const {
60     Polynomial result;
61
62     for (const auto& term1 : termArray) {
63         for (const auto& term2 : poly.termArray) {
64             result.termArray.push_back({ term1.first * term2.first, term1.second + term2.second });
65         }
66     }
67
68     for (auto it = result.termArray.begin(); it != result.termArray.end(); ++it) {
69         for (auto it2 = it + 1; it2 != result.termArray.end(); ++it2) {
70             if (it->second == it2->second) {
71                 it->first += it2->first;
72                 it2 = result.termArray.erase(it2);
73             }
74             else {
75                 ++it2;
76             }
77         }
78     }
79
80     return result;
81 }
82
83 float Polynomial::Eval(float x) const {
84     float result = 0.0;
85     for (const auto& term : termArray) {
86         result += term.first * pow(x, term.second);
87     }
88     return result;
89 }
90
91 std::ostream& operator<<(std::ostream& os, const Polynomial& poly) {
92     for (const auto& term : poly.termArray) {
93         os << term.first << "x^" << term.second << " + ";
94     }
95     return os;
96 }
97
98 std::istream& operator>>(std::istream& is, Polynomial& poly) {
99     std::cout << "輸入多項式的項數: ";
100     int numTerms;
101     is >> numTerms;
102
103     poly.termArray.clear();

```

```

104
105     for (int i = 0; i < numTerms; ++i) {
106         float coefficient;
107         int exponent;
108         std::cout << "輸入多項式的第 " << i + 1 << "個係數跟次方: ";
109         is >> coefficient >> exponent;
110         poly.termArray.push_back({ coefficient, exponent });
111     }
112
113     return is;
114 }
115
116 int main() {
117     Polynomial p1, p2;
118
119     std::cout << "第一個多項式:\n";
120     std::cin >> p1;
121
122     std::cout << "第二個多項式:\n";
123     std::cin >> p2;
124
125     Polynomial sum = p1.Add(p2);
126     Polynomial product = p1.Mult(p2);
127
128     std::cout << "多項式相加:: " << sum << std::endl;
129     std::cout << "多項式相乘: " << product << std::endl;
130
131     float x;
132
133     std::cout << "輸入x值計算多項式: ";
134     std::cin >> x;
135
136     std::cout << "多項式相加後代入x值: " << sum.Eval(x) << std::endl;
137     std::cout << "多項式相乘後代入x值: " << product.Eval(x) << std::endl;
138
139     auto startAdd = std::chrono::high_resolution_clock::now();
140     Polynomial sumPerformance = p1.Add(p2);
141     auto endAdd = std::chrono::high_resolution_clock::now();
142     std::chrono::duration<double> durationAdd = endAdd - startAdd;
143
144     std::cout << "Add operation took " << durationAdd.count() << " seconds.\n";
145
146     auto startMult = std::chrono::high_resolution_clock::now();
147     Polynomial productPerformance = p1.Mult(p2);
148     auto endMult = std::chrono::high_resolution_clock::now();
149     std::chrono::duration<double> durationMult = endMult - startMult;
150
151     std::cout << "Mult operation took " << durationMult.count() << " seconds.\n";
152
153     return 0;
154 }

```

## 效能分析(Analysis)：

### 空間複雜度：

這個實現使用了 `std::vector<std::pair<float, int>>` 來表示每個多項式的項目。在 `Mult` 函數中，生成的項目可能會比兩個多項式的項目總和更多。因此，空間複雜度取決於生成的多項式的項目數。

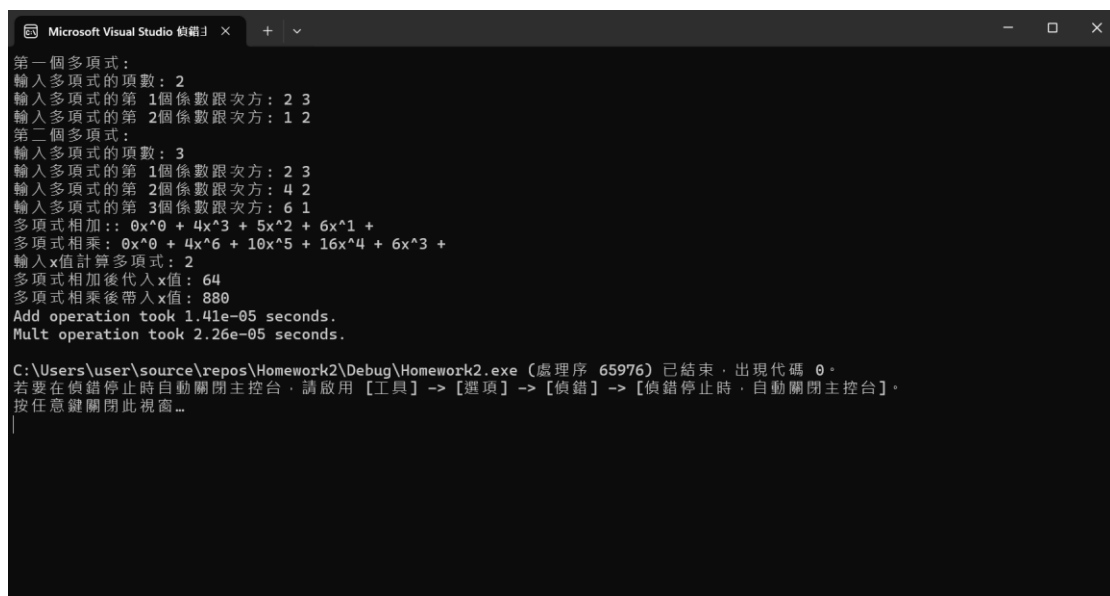
時間複雜度：

加法操 32 作 (Add) 的時間複雜度是  $O(N+M)$ ，其中  $N$  和  $M$  分別是兩個多項式的項數。

乘法操作 (Mult) 的時間複雜度是  $O(N * M)$ ，其中  $N$  和  $M$  分別是兩個多項式的項數。

評估操作 (Eval) 的時間複雜度是  $O(N)$ ，其中  $N$  是多項式的項數。

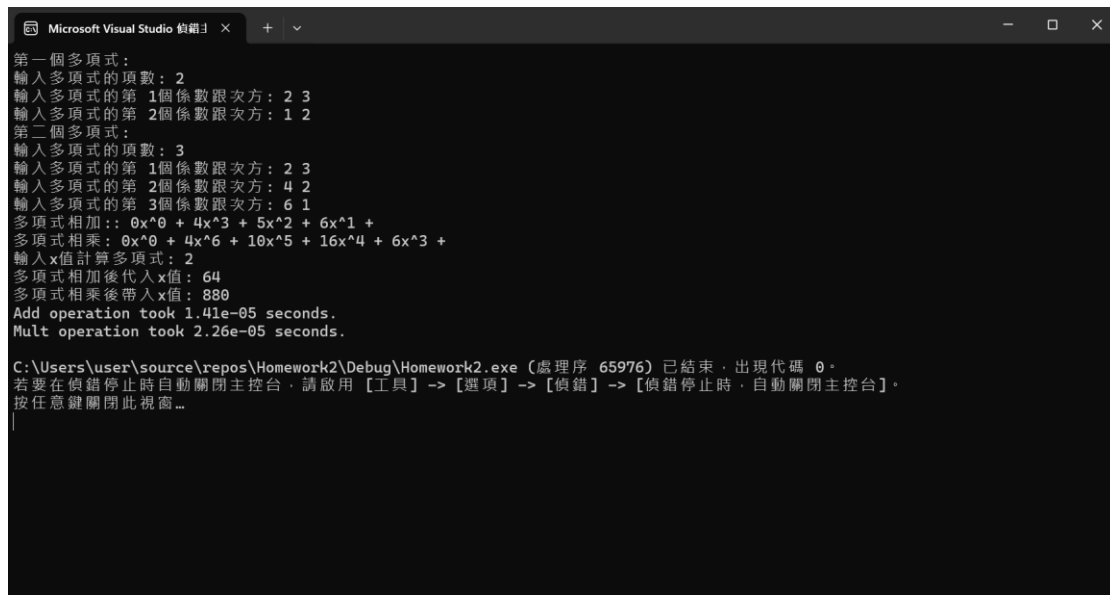
測試與驗證(Testing and Proving)：



```
Microsoft Visual Studio 偵錯器 x + v
第一個多項式：
輸入多項式的項數：2
輸入多項式的第 1 個係數跟次方：2 3
輸入多項式的第 2 個係數跟次方：1 2
第二個多項式：
輸入多項式的項數：3
輸入多項式的第 1 個係數跟次方：2 3
輸入多項式的第 2 個係數跟次方：4 2
輸入多項式的第 3 個係數跟次方：6 1
多項式相加:: 0x^0 + 4x^3 + 5x^2 + 6x^1 +
多項式相乘： 0x^0 + 4x^6 + 10x^5 + 16x^4 + 6x^3 +
輸入x值計算多項式：2
多項式相加後代入x值：64
多項式相乘後帶入x值：880
Add operation took 1.41e-05 seconds.
Mult operation took 2.26e-05 seconds.

C:\Users\user\source\repos\Homework2\Debug\Homework2.exe (處理序 65976) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用【工具】->【選項】->【偵錯】->【偵錯停止時，自動關閉主控台】。
按任意鍵關閉此視窗...
```

## 效能量測 (Measuring)：



```
Microsoft Visual Studio 偵錯器 X + -
第一個多項式：
輸入多項式的項數：2
輸入多項式的第 1 個係數跟次方：2 3
輸入多項式的第 2 個係數跟次方：1 2
第二個多項式：
輸入多項式的項數：3
輸入多項式的第 1 個係數跟次方：2 3
輸入多項式的第 2 個係數跟次方：4 2
輸入多項式的第 3 個係數跟次方：6 1
多項式相加：0x^0 + 4x^3 + 5x^2 + 6x^1 +
多項式相乘：0x^0 + 4x^6 + 10x^5 + 16x^4 + 6x^3 +
輸入x值計算多項式：2
多項式相加後代入x值：64
多項式相乘後代入x值：880
Add operation took 1.41e-05 seconds.
Mult operation took 2.26e-05 seconds.

C:\Users\user\source\repos\Homework2\Debug\Homework2.exe (處理序 65976) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用【工具】->【選項】->【偵錯】->【偵錯停止時，自動關閉主控台】。
按任意鍵關閉此視窗...
```

## 心得討論：

在實現多項式類的過程中，我學到如何使用 C++ 類來抽象數學概念。測試效能時，`chrono` 庫提供了方便的計時工具。理解加法、乘法和評估的實作讓我更深入地了解多項式操作。對於效能測試，不僅要注意正確性還需考慮效能優化，以確保在大規模操作下仍保持高效。這次經驗豐富，強化了我在 C++ 中的物件導向和數學應用的能力。