解題說明：

利用這段程式可以算出多項式的加法跟乘法。

效能分析：

空間複雜度：單個多項式的儲存空間：O(n

運算時的額外空間：

- 加法：O(n1+n2)
- 乘法：O(n1×n2)

時間複雜度：

- 乘法和加法：O(n1×n2)
- 新增項次：O(k) k 為當前項次

結果展示：

```
輸入格式 ax^n2+bx^n1+cx^n0+d（若為常數，可省略 x^0）
p1：3x^3+2x^2+3x+4
p2：5x+1
p1 = 3x^3+2x^2+3x^1+4
p2 = 5x^1+1
------------------------------------------------------------
新增項目
請輸入項次的係數：2
請輸入項次的指數：4
新增前的多項式 = 3x^3+2x^2+3x^1+4
新增後的多項式 = 3x^3+2x^2+3x^1+4+2x^4
------------------------------------------------------------
求值
請輸入要帶入的值：1
多項式：3x^3+2x^2+3x^1+4+2x^4
結果 = 14
------------------------------------------------------------
相加測試
(3x^3+2x^2+3x^1+4+2x^4) + (5x^1+1) = 3x^3+2x^2+8x^1+5+2x^4
------------------------------------------------------------
相乘測試
(3x^3+2x^2+3x^1+4+2x^4) * (5x^1+1) = 17x^4+13x^3+17x^2+23x^1+4+10x^5
```

心得：

這次的作業需要用到很多的類別，而我在這部分還不夠熟悉，導致我花費了需多時間在這次的作業上。

申論及開發報告：

1. 類別 Term：
    - 用於表示多項式中的單項式，包含兩個成員：係數 (coef) 和指數 (exp)。
    - 為了支援函式操作，該類別宣告了 Polynomial 和輸出運算子 (ostream) 為 friend。
2. 類別 Polynomial：
    - 表示一個多項式，包含非零項陣列 (termArray)、陣列容量 (capacity)、非零項數量 (terms)。
    - 主要功能：
        1. 新增項目 (NewTerm)： 添加新的多項式項。
        2. 多項式相加 (Add)： 合併兩個多項式。
        3. 多項式相乘 (Mult)： 計算兩多項式的積。
        4. 多項式求值 (Eval)： 對多項式進行求值。
        5. 輸入與輸出運算子重載 (operator<< 和 operator>>)： 使程式與使用者的交互更自然。
3. 技術實現
    1. 陣列動態擴展：
        - 當 termArray 容量不足時，透過 new 動態分配更大陣列，並將舊數據拷貝進新陣列。

2. 多項式合併操作：
    - 相加與相乘時，使用兩個陣列 (loc 和 data) 分別存儲指數和累加的係數，避免直接操作多項式陣列，提升運算效率與程式可讀性。
3. 求值運算的次方處理：
    - 使用循環模擬次方運算，確保程式執行的跨平台穩定性。
4. 輸入解析：
    - 使用 istream 重載，處理多項式表達式中正負號與冪次的變化。

程式片段：

```
#include <iostream>
#include <string>
using namespace std;

class Polynomial;

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

class Polynomial { // 多項式類別
    friend ostream& operator<<(ostream& os, const Polynomial& p);
    friend istream& operator>>(istream& input, Polynomial& p);
private:
    Term* termArray; // 儲存非零項的陣列
    int capacity;    // 陣列容量
    int terms;       // 非零項數量

public:
    Polynomial(); // 預設建構函式：p(x) = 0
    Polynomial Add(Polynomial poly);  // 多項式相加
    Polynomial Mult(Polynomial poly); // 多項式相乘
    float Eval(float f);              // 多項式求值
    void NewTerm(const float newCoef, const int newExp); // 新增項目
};
```

```cpp
Polynomial::Polynomial() : capacity(2), terms(0) {
    this->termArray = new Term[capacity];
}

// 新增項目函式
void Polynomial::NewTerm(const float newCoef, const int newExp) {
    if (this->terms == this->capacity) { // 空間不足時擴展容量
        this->capacity *= 2;
        Term* temp = new Term[this->capacity];
        copy(this->termArray, this->termArray + terms, temp);
        delete[] this->termArray;
        this->termArray = temp;
    }
    this->termArray[this->terms].coef = newCoef; // 設定係數
    this->termArray[this->terms++].exp = newExp; // 設定指數
}

// 多項式相加
Polynomial Polynomial::Add(Polynomial poly) {
    Polynomial res;
    int* loc = new int[poly.terms + this->terms]; // 儲存指數
    float* data = new float[poly.terms + this->terms]; // 儲存係數
    int use_len = 0; // 使用長度

    // 將當前多項式的項加入陣列
    for (int i = 0; i < this->terms; i++) {
        int t = -1; // -1 表示未找到
        for (int j = 0; j < use_len; j++) {
            if (this->termArray[i].exp == loc[j]) {
                t = j;
                break;
            }
```

```
        }
        if (t == -1) { // 新的指數
            loc[use_len] = this->termArray[i].exp;
            data[use_len++] = this->termArray[i].coef;
        }
        else {
            data[t] += this->termArray[i].coef; // 相同指數合併係數
        }
    }

    // 處理第二個多項式
    for (int i = 0; i < poly.terms; i++) {
        int t = -1;
        for (int j = 0; j < use_len; j++) {
            if (poly.termArray[i].exp == loc[j]) {
                t = j;
                break;
            }
        }
        if (t == -1) {
            loc[use_len] = poly.termArray[i].exp;
            data[use_len++] = poly.termArray[i].coef;
        }
        else {
            data[t] += poly.termArray[i].coef;
        }
    }

    // 將結果存入 res
    for (int i = 0; i < use_len; i++)
        res.NewTerm(data[i], loc[i]);

    delete[] loc;
```

```
97      delete[] data;
98
99      return res;
100     }
101
102     // 多項式相乘
103     Polynomial Polynomial::Mult(Polynomial poly) {
104         Polynomial res;
105         int* loc = new int[poly.terms * this->terms];
106         float* data = new float[poly.terms * this->terms];
107         int use_len = 0;
108
109         for (int i = 0; i < this->terms; i++) {
110             for (int j = 0; j < poly.terms; j++) {
111                 float t_coef = this->termArray[i].coef * poly.termArray[j].coef; // 係數相乘
112                 int t_exp = this->termArray[i].exp + poly.termArray[j].exp;      // 指數相加
113
114                 int t = -1;
115                 for (int k = 0; k < use_len; k++) {
116                     if (t_exp == loc[k]) {
117                         t = k;
118                         break;
119                     }
120                 }
121                 if (t == -1) { // 新的項目
122                     loc[use_len] = t_exp;
123                     data[use_len++] = t_coef;
124                 }
```

```
125             else {
126                 data[t] += t_coef;
127             }
128         }
129     }
130
131     for (int i = 0; i < use_len; i++)
132         res.NewTerm(data[i], loc[i]);
133
134     delete[] loc;
135     delete[] data;
136
137     return res;
138 }
139
140 // 多項式求值
141 float Polynomial::Eval(float f) {
142     float res = 0.0f;
143     for (int i = 0; i < this->terms; i++) {
144         float temp = this->termArray[i].coef;
145         for (int j = 0; j < this->termArray[i].exp; j++)
146             temp *= f; // 計算次方
147         res += temp;
148     }
149     return res;
150 }
151
152 // 輸入運算子
153 istream& operator>>(istream& input, Polynomial& p) {
154     float coef = 0.0f;
155     int exp = 0;
156     char ch;
```

```
157         bool isNegative = false;
158
159         while (true) {
160             input >> coef;
161             if (isNegative) coef = -coef;
162             isNegative = false;
163
164             input.get(ch);
165             if (ch == 'x') {
166                 input.get(ch);
167                 if (ch == '^') {
168                     input >> exp;
169                 }
170                 else {
171                     exp = 1;
172                     input.unget();
173                 }
174             }
175             else {
176                 exp = 0;
177                 input.unget();
178             }
179
180             p.NewTerm(coef, exp);
181
182             input.get(ch);
183             if (ch == '\n' || input.eof()) break;
184             if (ch == '-') {
185                 isNegative = true;
186             }
187             else if (ch != '+') {
188                 cout << "輸入格式錯誤！請使用正確的多項式格式。\n";
189                 break;
```

```
        }
    }
    return input;
}

// 輸出運算子
ostream& operator<<(ostream& output, const Polynomial& p) {
    for (int i = 0; i < p.terms; i++) {
        if (p.termArray[i].exp == 0) {
            output << (p.termArray[i].coef < 0 ? "" : "+") << p.termArray[i].coef;
        }
        else {
            if (i == 0) {
                output << p.termArray[i].coef << "x^" << p.termArray[i].exp;
            }
            else {
                output << (p.termArray[i].coef < 0 ? "" : "+") << p.termArray[i].coef << "x^" << p.termArray[i].exp;
            }
        }
    }
    return output;
}

int main() {
    cout << "輸入格式 ax^n2+bx^n1+cx^n0+d（若為常數，可省略 x^0）\n";
    Polynomial p1, p2;
    cout << "p1：";
    cin >> p1;
    cout << "p2：";
    cin >> p2;
```

```
    cout << "p1 = " << p1 << endl;
    cout << "p2 = " << p2 << endl;

    cout << "------------------------------------------------------------\n";
    cout << "新增項目\n";
    float t_coef = 0.0f;
    int t_exp = 0;
    cout << "請輸入項次的係數：";
    cin >> t_coef;
    cout << "請輸入項次的指數：";
    cin >> t_exp;
    cout << "新增前的多項式 = " << p1 << endl;
    p1.NewTerm(t_coef, t_exp);
    cout << "新增後的多項式 = " << p1 << endl;

    cout << "------------------------------------------------------------\n";
    cout << "求值\n";
    float f = 0.0f;
    cout << "請輸入要帶入的值：";
    cin >> f;
    cout << "多項式：" << p1 << endl;
    cout << "結果 = " << p1.Eval(f) << endl;

    cout << "------------------------------------------------------------\n";
    cout << "相加測試\n";
    cout << "(" << p1 << ") + (" << p2 << ") = " << p1.Add(p2) << endl;

    cout << "------------------------------------------------------------\n";
    cout << "相乘測試\n";
    cout << "(" << p1 << ") * (" << p2 << ") = " << p1.Mult(p2) << endl;

    return 0;
```