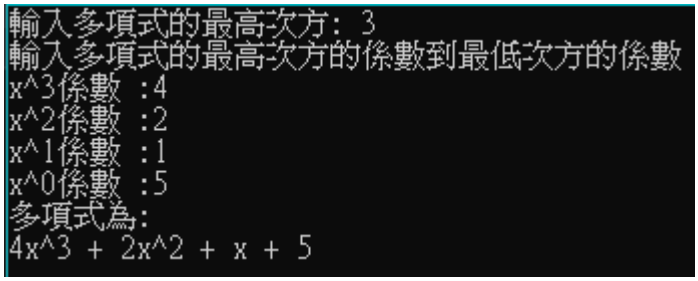
```
class Polynomial {
private:
    int* coef;
    int degree;

public:
    Polynomial() : coef(nullptr), degree(0) {}//預先將coef指標指向null以及最高次方預設為0
    ~Polynomial() {
        delete[] coef;
    }

    friend istream& operator>>(istream& in, Polynomial& poly) {//輸入運算子多載
        cout << "輸入多項式的最高次方: ";
        in >> poly.degree;

        delete[] poly.coef;
        poly.coef = new int[poly.degree + 1];

        cout << "輸入多項式的最高次方的係數到最低次方的係數" << endl;
        for (int i = poly.degree; i >= 0; --i) {
            cout << "x^" << i << "係數 :";
            in >> poly.coef[i];
        }
        return in;
    }
```

```
friend ostream& operator<<(ostream& out, const Polynomial& poly) {//輸出運算子多載
    bool first = true;//first是控制第一項會不會輸出 + 及 - 符號的變數
    for (int i = poly.degree; i >= 0; --i) {
        if (poly.coef[i] != 0) {

            if (!first && poly.coef[i] > 0) {
                out << " + ";//不是第一項 係數為正時要加 + 號
            }
            else if (poly.coef[i] < 0) {
                out << " - ";//係數為負時要加 - 號
            }

            if (abs(poly.coef[i]) != 1 || i == 0) {
                cout << abs(poly.coef[i]);
            }
            if (i > 0) {
                out << "x";
                if (i > 1) cout << "^" << i;
            }
            first = false;
        }
    }
    if (first) out << "0";
    return out;
}
```

```cpp
int main() {
    Polynomial p;
    cin >> p;

    cout << "多項式為:" << endl;
    cout << p << endl;

    return 0;
}
```

輸出結果:



效能分析:

時間複雜度為有多少項 O(degree)，假設 degree 為 4 代表總共有 4 項，那麼時間複雜度為 O(4)；空間複雜度為 O(1)，因為空間的配置在類別的動態記憶體配置裡就分配完了，所以不會有多出來的空間。O(1)是因為額外的變數要有空間儲存。

心得:

我發現要將人類平常就在用的表示方式，用程式寫出來是多麼複雜，平常寫可能寫一個多項式只要幾秒鐘，但在程式裡用陣列去儲存輸入的資料，然後還要使用運算子多載，以及判斷正負符號要在甚麼時候加入。