資料結構報告

學號:41243147

姓名:楊承哲

1 解題說明 2

2 實作 3

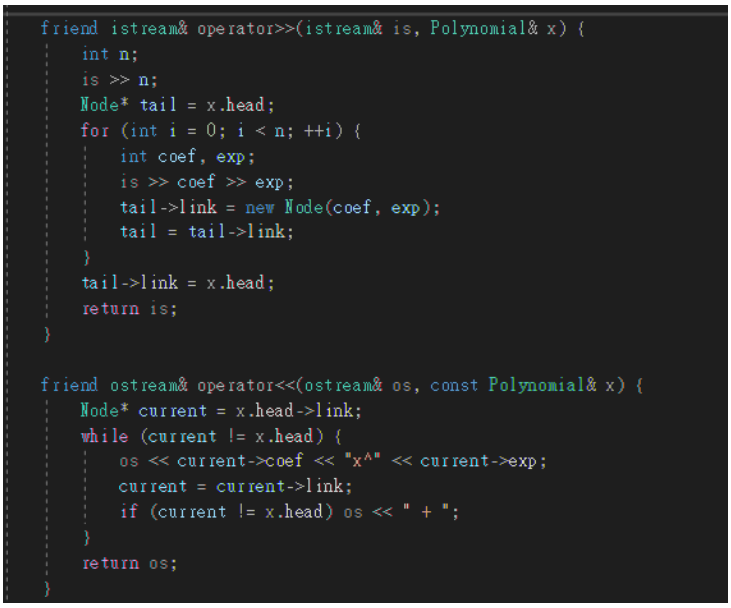
3 效能分析 4

4 測試與過程 5

1.解題說明

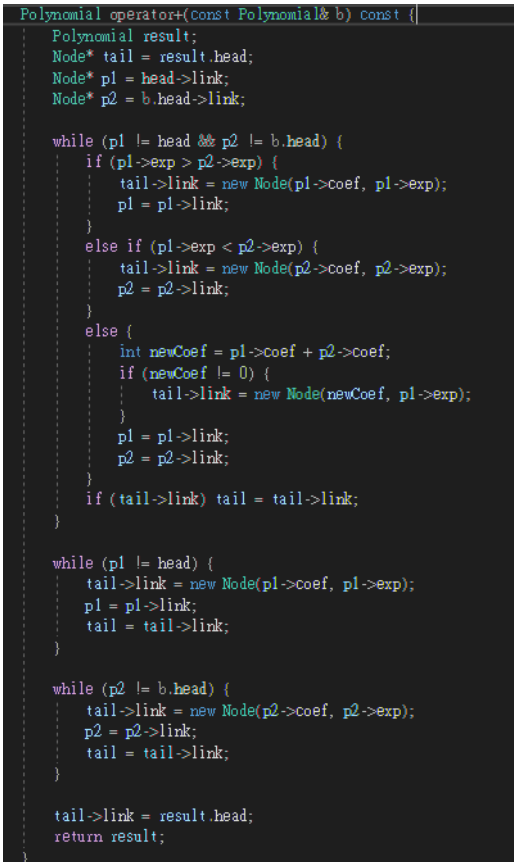
這題要使用陣列做一個多項式的計算包括了、和、差、積和帶入值。

讀取輸入的多項式，轉成鏈結串列。
然後將鏈結串列輸出



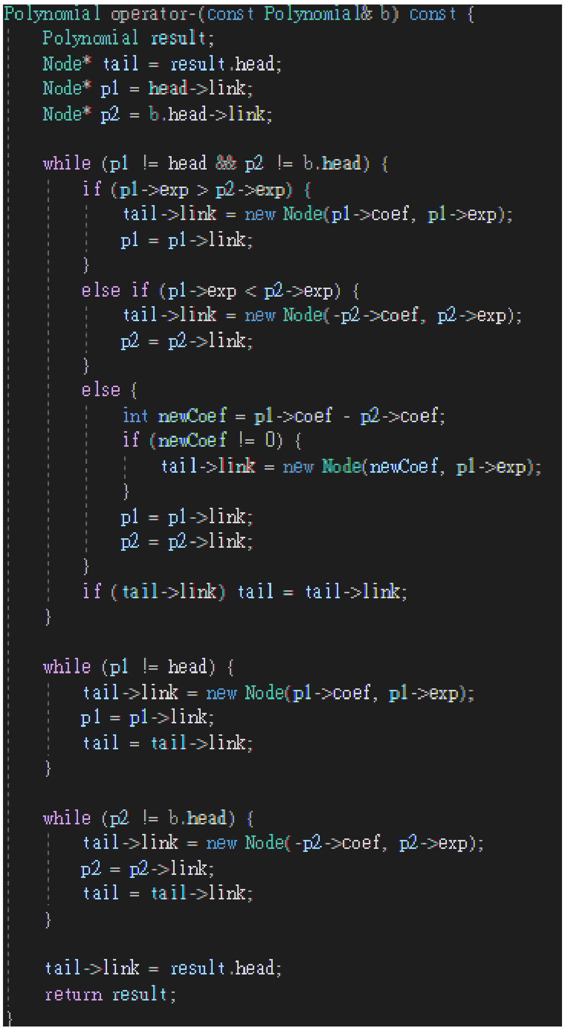
```cpp
friend istream& operator>>(istream& is, Polynomial& x) {
    int n;
    is >> n;
    Node* tail = x.head;
    for (int i = 0; i < n; ++i) {
        int coef, exp;
        is >> coef >> exp;
        tail->link = new Node(coef, exp);
        tail = tail->link;
    }
    tail->link = x.head;
    return is;
}

friend ostream& operator<<(ostream& os, const Polynomial& x) {
    Node* current = x.head->link;
    while (current != x.head) {
        os << current->coef << "x^" << current->exp;
        current = current->link;
        if (current != x.head) os << " + ";
    }
    return os;
}
```

加法實作



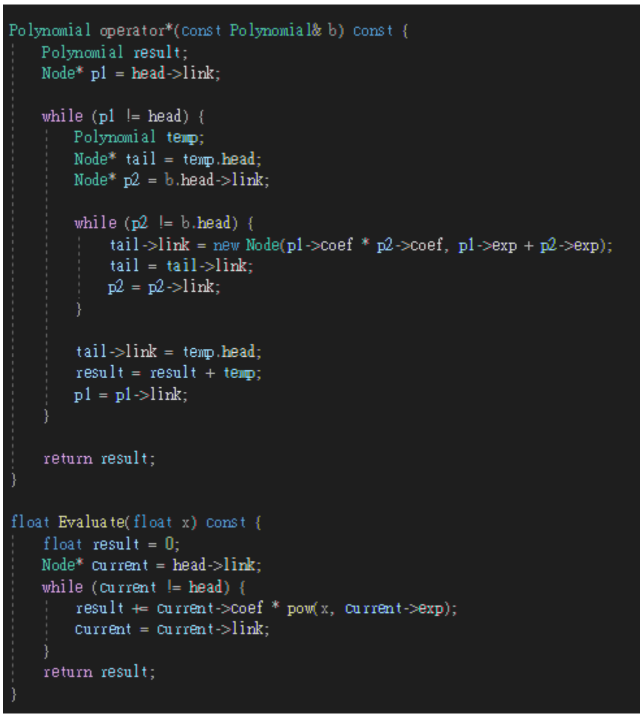
```cpp
Polynomial operator+(const Polynomial& b) const {
    Polynomial result;
    Node* tail = result.head;
    Node* p1 = head->link;
    Node* p2 = b.head->link;

    while (p1 != head && p2 != b.head) {
        if (p1->exp > p2->exp) {
            tail->link = new Node(p1->coef, p1->exp);
            p1 = p1->link;
        }
        else if (p1->exp < p2->exp) {
            tail->link = new Node(p2->coef, p2->exp);
            p2 = p2->link;
        }
        else {
            int newCoef = p1->coef + p2->coef;
            if (newCoef != 0) {
                tail->link = new Node(newCoef, p1->exp);
            }
            p1 = p1->link;
            p2 = p2->link;
        }
        if (tail->link) tail = tail->link;
    }

    while (p1 != head) {
        tail->link = new Node(p1->coef, p1->exp);
        p1 = p1->link;
        tail = tail->link;
    }

    while (p2 != b.head) {
        tail->link = new Node(p2->coef, p2->exp);
        p2 = p2->link;
        tail = tail->link;
    }

    tail->link = result.head;
    return result;
}
```

減法實作



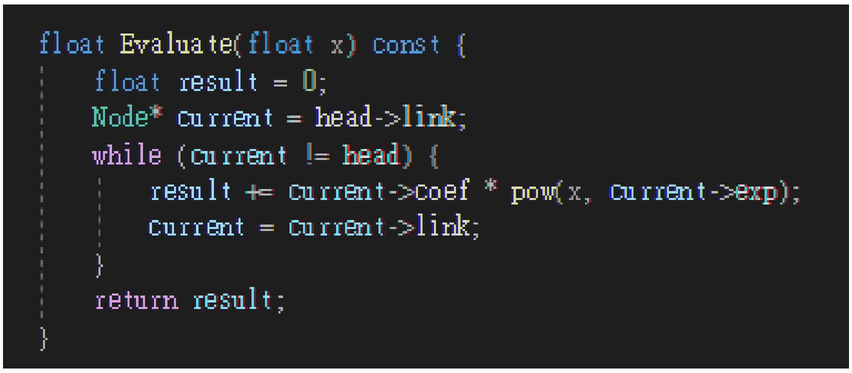
```
Polynomial operator-(const Polynomial& b) const {
    Polynomial result;
    Node* tail = result.head;
    Node* p1 = head->link;
    Node* p2 = b.head->link;

    while (p1 != head && p2 != b.head) {
        if (p1->exp > p2->exp) {
            tail->link = new Node(p1->coef, p1->exp);
            p1 = p1->link;
        }
        else if (p1->exp < p2->exp) {
            tail->link = new Node(-p2->coef, p2->exp);
            p2 = p2->link;
        }
        else {
            int newCoef = p1->coef - p2->coef;
            if (newCoef != 0) {
                tail->link = new Node(newCoef, p1->exp);
            }
            p1 = p1->link;
            p2 = p2->link;
        }
        if (tail->link) tail = tail->link;
    }

    while (p1 != head) {
        tail->link = new Node(p1->coef, p1->exp);
        p1 = p1->link;
        tail = tail->link;
    }

    while (p2 != b.head) {
        tail->link = new Node(-p2->coef, p2->exp);
        p2 = p2->link;
        tail = tail->link;
    }

    tail->link = result.head;
    return result;
}
```

乘法實作



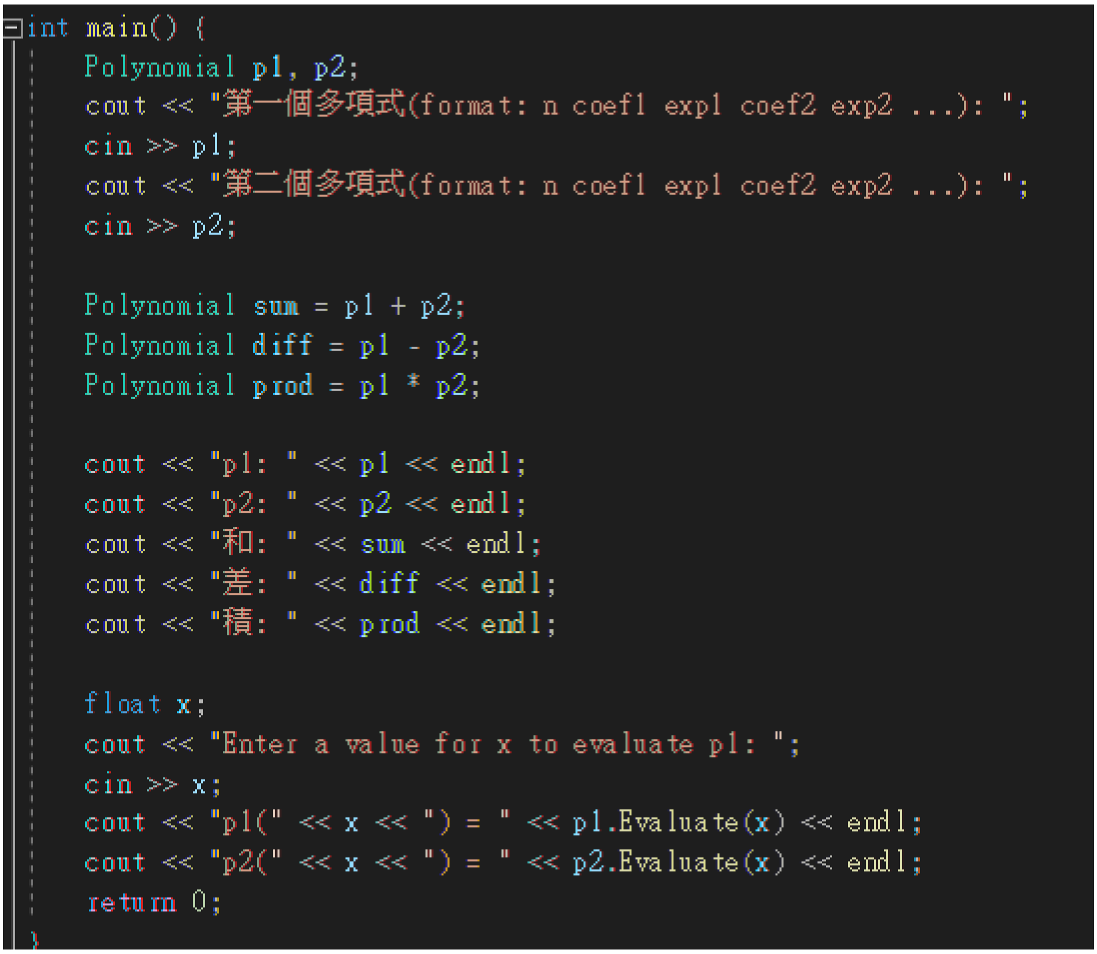
```
Polynomial operator*(const Polynomial& b) const {
    Polynomial result;
    Node* p1 = head->link;

    while (p1 != head) {
        Polynomial temp;
        Node* tail = temp.head;
        Node* p2 = b.head->link;

        while (p2 != b.head) {
            tail->link = new Node(p1->coef * p2->coef, p1->exp + p2->exp);
            tail = tail->link;
            p2 = p2->link;
        }

        tail->link = temp.head;
        result = result + temp;
        p1 = p1->link;
    }

    return result;
}

float Evaluate(float x) const {
    float result = 0;
    Node* current = head->link;
    while (current != head) {
        result += current->coef * pow(x, current->exp);
        current = current->link;
    }
    return result;
}
```

帶入值實作



```
float Evaluate(float x) const {
    float result = 0;
    Node* current = head->link;
    while (current != head) {
        result += current->coef * pow(x, current->exp);
        current = current->link;
    }
    return result;
}
```

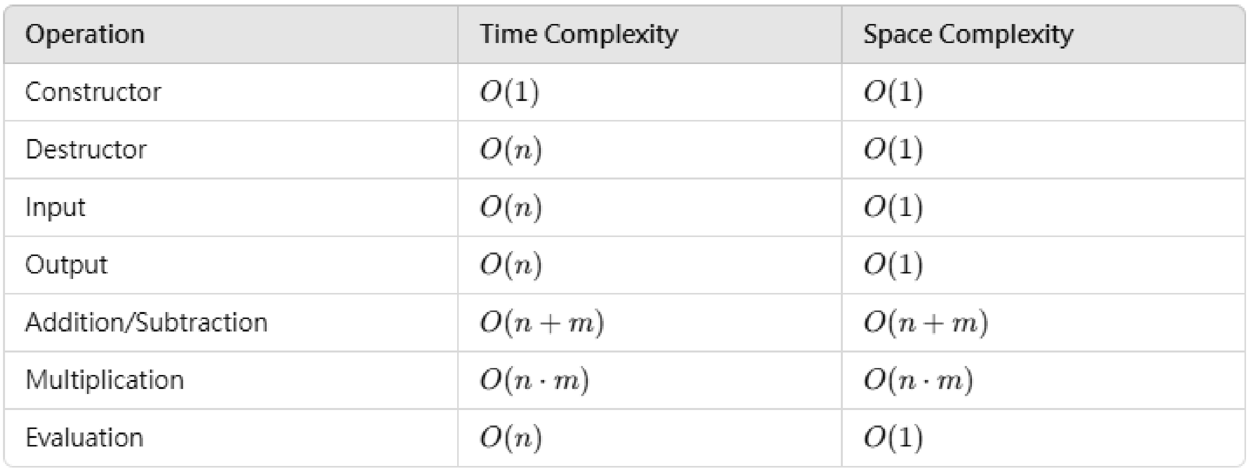
2.程式實作



```
int main() {
    Polynomial p1, p2;
    cout << "第一個多項式(format: n coef1 exp1 coef2 exp2 ...): ";
    cin >> p1;
    cout << "第二個多項式(format: n coef1 exp1 coef2 exp2 ...): ";
    cin >> p2;

    Polynomial sum = p1 + p2;
    Polynomial diff = p1 - p2;
    Polynomial prod = p1 * p2;

    cout << "p1: " << p1 << endl;
    cout << "p2: " << p2 << endl;
    cout << "和: " << sum << endl;
    cout << "差: " << diff << endl;
    cout << "積: " << prod << endl;

    float x;
    cout << "Enter a value for x to evaluate p1: ";
    cin >> x;
    cout << "p1(" << x << ") = " << p1.Evaluate(x) << endl;
    cout << "p2(" << x << ") = " << p2.Evaluate(x) << endl;
    return 0;
}
```

輸入依序是有幾項、係數1、次方1…以此類推。
在兩個多項式輸入結束後還有帶入值輸入。
輸入則是和、差、積、帶入值。

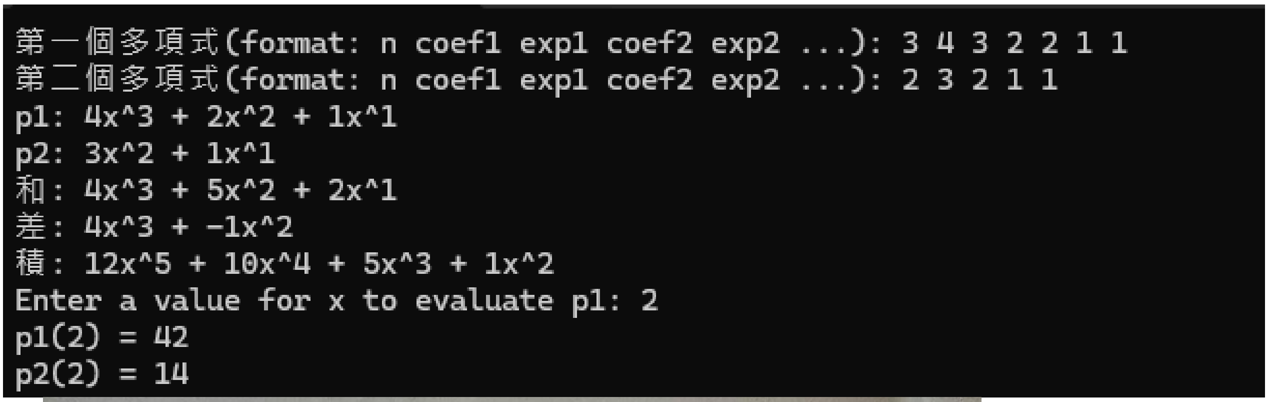
效能分析



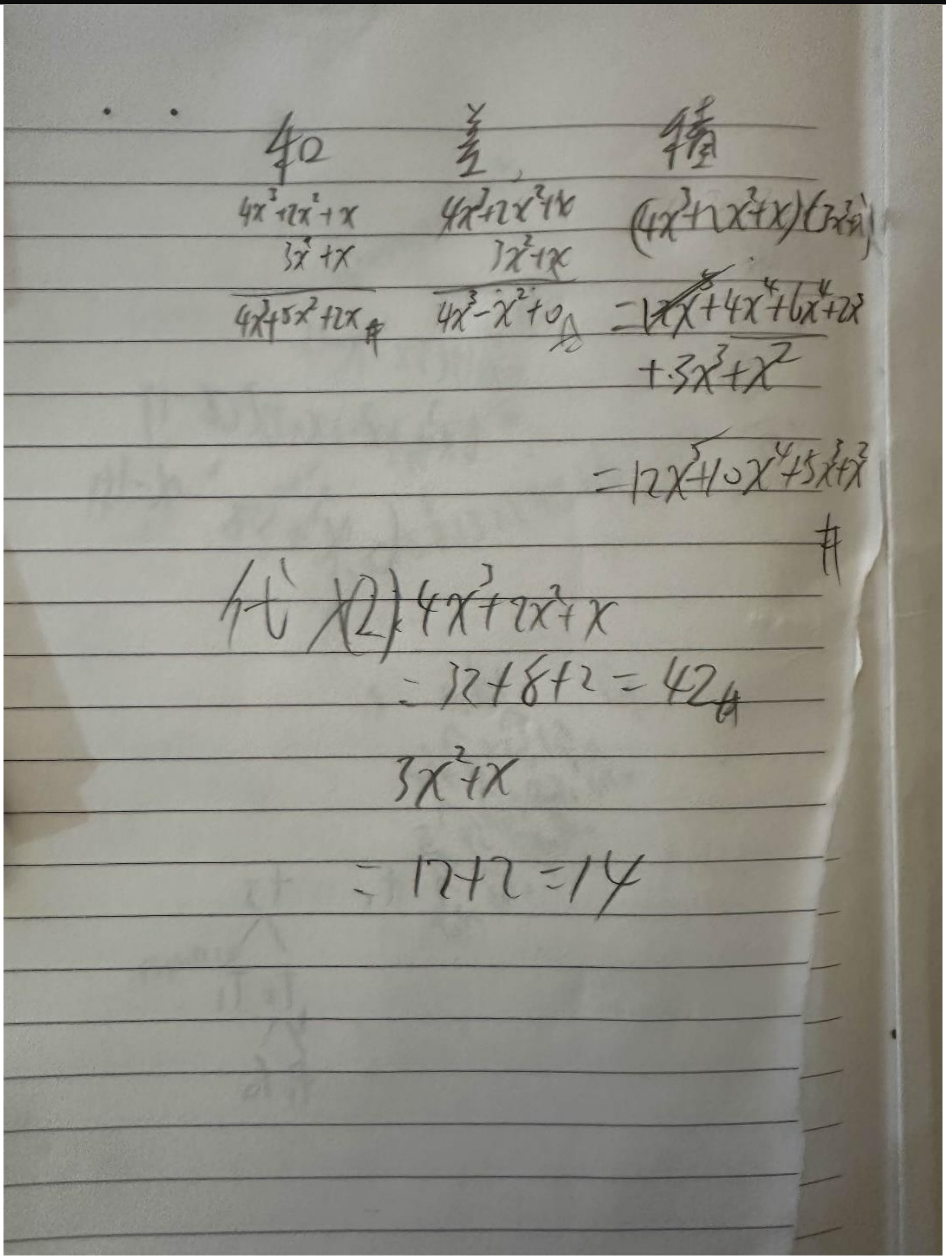
Operation	Time Complexity	Space Complexity
Constructor	$O(1)$	$O(1)$
Destructor	$O(n)$	$O(1)$
Input	$O(n)$	$O(1)$
Output	$O(n)$	$O(1)$
Addition/Subtraction	$O(n+m)$	$O(n+m)$
Multiplication	$O(n \cdot m)$	$O(n \cdot m)$
Evaluation	$O(n)$	$O(1)$

測試

1.



```
第一個多項式(format: n coef1 exp1 coef2 exp2 ...): 3 4 3 2 2 1 1
第二個多項式(format: n coef1 exp1 coef2 exp2 ...): 2 3 2 1 1
p1: 4x^3 + 2x^2 + 1x^1
p2: 3x^2 + 1x^1
和: 4x^3 + 5x^2 + 2x^1
差: 4x^3 + -1x^2
積: 12x^5 + 10x^4 + 5x^3 + 1x^2
Enter a value for x to evaluate p1: 2
p1(2) = 42
p2(2) = 14
```



和

$4x^3+2x^2+x$

$3x^2+x$

$4x^3+5x^2+2x$ #

差

$4x^3+2x^2+x$

$3x^2+x$

$4x^3-x^2+0$ △

積

$(4x^3+2x^2+x)(3x^2+x)$

$=12x^5+4x^4+6x^4+2x^3$

$+3x^3+x^2$

$=12x^5+10x^4+5x^3+x^2$

#

代 x(2) $4x^3+2x^2+x$

$=32+8+2=42$ #

$3x^2+x$

$=12+2=14$

經過我精美的運算，答案應該是非常正確的。

心得:

在實作這個程式的時候，比我想像的還要複雜一點，也具有挑戰性，讓我對鏈結串列更加熟悉。