解題說明

(a) Istream

使用 is 來代替 cin ，輸入進 c 與 e 裡面然後個別放入 temp 裡面

```
    is >> c >> e;
    temp = x.GetNode();
    temp->coef = c;
    temp->exp = e;
```

如果放的係數是亂的需要找到對應位置才能放入

```
// 找到正確的插入位置
Node* current = x.head->link;
while (current != x.head && current->exp >
    prev = current;
    current = current->link;
}
```

然後插入節點回歸插入頭

```
// 插入節點
temp->link = current;
prev->link = temp;
prev = x.head; // 重置 prev 指向頭節點
```

(b) Ostream

顧名思義將節點個別輸出

```
    Node* temp = x.head->link;
    while (temp != x.head) {
        os << temp->coef << "x^" << temp->exp <<
        temp = temp->link;
    }
    return os;
```

呼叫當前節點的數值外加 ostream

(c) Polynomial(const polynomial& a)

```
// 複製建構子實作
Polynomial::Polynomial(const Polynomial& a) {
    head = new Node; // 創建新的頭節點
    head->link = head;
    Node* temp = a.head->link;

    while (temp != a.head) { // 遍歷源多項式的每個節點
        Node* newNode = GetNode(); // 從可用空間列表中獲取新的節點
        newNode->coef = temp->coef; // 複製系數,指數
        newNode->exp = temp->exp;
        newNode->link = head->link; // 將新節點插入到新多項式的鏈表中
        head->link = newNode; // 更新頭節點的鏈接
        temp = temp->link; // 移動到下一個節點
    }
}
```

就是創建一個新節點然後將該節點連接是 head node 身上，完成複製

(d) Operator=

賦值 a 給我們的 this

```
// 賦值運算子實作
const Polynomial& Polynomial::operator=(const Polynomial& a) {
    if (this != &a) {
        this->~Polynomial();
        new (this) Polynomial(a);
    }
    return *this;
}
```

先清空 this 裡面 再將數值 a 丟進去

(e) ~polynomial()

```cpp
// 解構子實作
Polynomial::~Polynomial() {
    Node* temp = head->link;
    while (temp != head) {
        Node* toDelete = temp;
        temp = temp->link;
        ReturnNode(toDelete);
    }
    delete head;
}
```

將節點一個一個做刪除 temp 從頭 while 直接刪除到結尾然後結束

(f) Operator+

```cpp
while (p1 != head && p2 != b.head) {
        Node* temp = GetNode();
       if (p1->exp == p2->exp) {
           temp->coef = p1->coef + p2->coef;
           temp->exp = p1->exp;
           p1 = p1->link;
           p2 = p2->link;
       } else if (p1->exp > p2->exp) {
           temp->coef = p1->coef;
           temp->exp = p1->exp;
           p1 = p1->link;
       } else {
           temp->coef = p2->coef;
           temp->exp = p2->exp;
           p2 = p2->link;
       }
       temp->link = result.head;
       p3->link = temp;
```

```
        p3 = temp;
    }
```

將相同的相加，大於目前的並且沒得相加的先放進去

```
while (p1 != head) {
        Node* temp = GetNode();
        temp->coef = p1->coef;
        temp->exp = p1->exp;
        temp->link = result.head;
        p3->link = temp;
        p3 = temp;
        p1 = p1->link;
    }
```

再度確認沒有沒得輸出的在 p1,p2

(g) Operator-

同理 operator+,但要加負號

```
} else {
    temp->coef = -p2->coef;
    temp->exp = p2->exp;
    p2 = p2->link;
}
```

(h) Operator*

```
Polynomial result;
Node* p1 = head->link;
Node* p2 = nullptr;

// 使用哈希表來儲存相同次方項的系數和避免重複計算
unordered_map<int, int> terms; // 次方 -> 系數
```

```
int expSum = p1->exp + p2->exp;
int coefProd = p1->coef * p2->coef;
terms[expSum] += coefProd;
p2 = p2->link;
```

項數相乘結果儲存

```
// 將哈希表中的項目轉換回多項式鏈表
for (auto& term : terms) {
    Node* newNode = GetNode();
    newNode->coef = term.second;
    newNode->exp = term.first;

    // 插入到結果多項式中按次方由大到小排序
    Node* prev = result.head;
    Node* current = result.head->link;
    while (current != result.head && current->exp > newNode->exp) {
        prev = current;
        current = current->link;
    }
    newNode->link = current;
    prev->link = newNode;
}
```

(i) Evaluate(float x)

```
// 計算多項式值實作
float Polynomial::Evaluate(float x) const {
    float result = 0;
    Node* temp = head->link;
    while (temp != head) {
        result += temp->coef * pow(x, temp->exp);
        temp = temp->link;
    }
    return result;
}
```

x 次方係相加(同解構子做法

Algorithm Design & Programming

參閱程式檔案 ” Polynomial.cpp”

效能分析

時間複雜度

O(n*m)

空間複雜度

O(n*m)

N 為多項式 p1 的總數

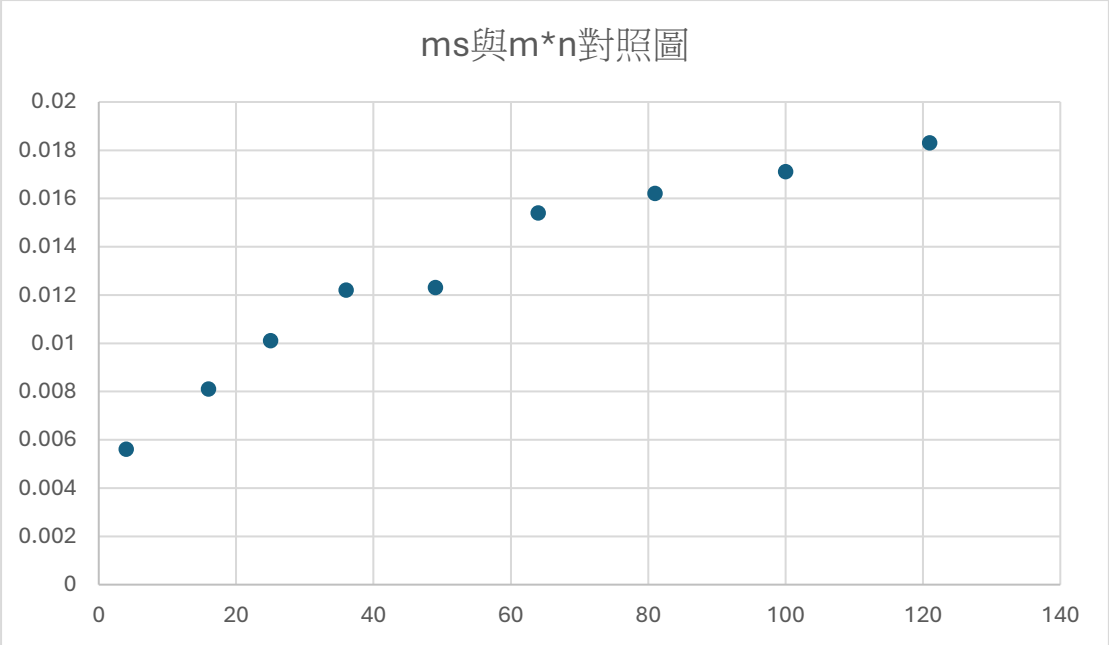
m 為多項式 p2 的總數

測試與驗證

```
輸入多項式1 : 4 5 0 2 4 3 3 1 2
多項式1: 2x^4 3x^3 1x^2 5x^0
輸入多項式2 : 6 6 0 5 1 4 2 3 3 4 2 5 1
多項式2: 3x^3 4x^2 4x^2 5x^1 5x^1 6x^0
多項式1 + 多項式2: 11x^0 5x^1 5x^1 4x^2 5x^2 6x^3 2x^4
多項式1 - 多項式2: -1x^0 -5x^1 -5x^1 -4x^2 -3x^2 0x^3 2x^4
多項式1 * 多項式2: 6x^7 25x^6 47x^5 50x^4 43x^3 46x^2 50x^1 30x^0
多項式1 在 x = 2 的值: 65

--------------------------------
```

效能量測



心得

參閱檔案

I/O stream https://cp.wiwiho.me/iostream/

鏈結串列(Linked List) https://pisces1026.wordpress.com/2017/09/21/cc-linked-list/