

# 資料結構作業 2

---

姓名: 侯旭昇

August 05, 2024

# CONTENTS

1. 解題說明
- 2 . 演算法設計與實作
- 3 . 效能分析
- 4 . 測試與驗證
5. 效能量測
6. 心得

## CHAPTER 1

### 解題說明

用 Polynomial 類別來表示多項式，並提供加法、乘法和評估操作。處理多項式的輸入/出，並使用運算符號來方便這些操作。

### 舉例

假設我們有兩個多項式  $p_1(x) = 3x^2 + 2x + 1$  和  $p_2(x) = x^2 + 4$ ，我們希望實現以下功能：

1. **加法**:  $p_1 + p_2 = (3+1)x^2 + 2x + (1+4) = 4x^2 + 2x + 5$
2. **乘法**:  $p_1 * p_2 = (3x^2 + 2x + 1)(x^2 + 4) = 3x^4 + 12x^2 + 2x^3 + 8x + x^2 + 4 = 3x^4 + 2x^3 + 13x^2 + 8x + 4$
3. **評估**: 在  $x = 2$  時評估  $p_1(2) = 3*2^2 + 2*2 + 1 = 17$

## CHAPTER 2

### 演算法設計與實作

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <cmath>
5  class Term {
6  public:
7      Term(float c = 0, int e = 0) : coef(c), exp(e) {}
8      float coef;
9      int exp;
10 };
```

```

12  class Polynomial {
13  public:
14      Polynomial() = default;
15      Polynomial(const Polynomial&) = default;
16      Polynomial& operator=(const Polynomial&) = default;
17      ~Polynomial() = default;
18
19      void AddTerm(float coef, int exp);
20      Polynomial Add(const Polynomial& poly) const;
21      Polynomial Mult(const Polynomial& poly) const;
22      float Eval(float f) const;
23
24      friend std::ostream& operator<<(std::ostream& os, const Polynomial& poly);
25      friend std::istream& operator>>(std::istream& is, Polynomial& poly);
26
27  private:
28      std::vector<Term> terms;
29
30      static bool CompareTerms(const Term& a, const Term& b) {
31          return a.exp > b.exp;
32      }
33  };
34
35  void Polynomial::AddTerm(float coef, int exp) {
36      if (coef == 0) return;
37
38      for (auto& term : terms) {
39          if (term.exp == exp) {
40              term.coef += coef;
41              if (term.coef == 0) {
42                  terms.erase(std::remove_if(terms.begin(), terms.end(), [exp](const Term& t) { return t.exp == exp; }), terms.end());
43              }
44              std::sort(terms.begin(), terms.end(), CompareTerms);
45              return;
46          }
47      }
48
49      terms.emplace_back(coef, exp);
50      std::sort(terms.begin(), terms.end(), CompareTerms);
51  }

```

```

53  Polynomial Polynomial::Add(const Polynomial& poly) const {
54      Polynomial result = *this;
55      for (const auto& term : poly.terms) {
56          result.AddTerm(term.coef, term.exp);
57      }
58      return result;
59  }
60
61  Polynomial Polynomial::Mult(const Polynomial& poly) const {
62      Polynomial result;
63      for (const auto& term1 : terms) {
64          for (const auto& term2 : poly.terms) {
65              result.AddTerm(term1.coef * term2.coef, term1.exp + term2.exp);
66          }
67      }
68      return result;
69  }

71  float Polynomial::Eval(float f) const {
72      float result = 0;
73      for (const auto& term : terms) {
74          result += term.coef * std::pow(f, term.exp);
75      }
76      return result;
77  }

79  std::ostream& operator<<(std::ostream& os, const Polynomial& poly) {
80      if (poly.terms.empty()) {
81          os << "0";
82          return os;
83      }
84      bool first = true;
85      for (const auto& term : poly.terms) {
86          if (!first && term.coef > 0) os << " + ";
87          if (term.coef < 0) os << " - ";
88          if (!first && term.coef < 0) os << -term.coef;
89          else os << term.coef;
90
91          if (term.exp > 0) os << "x^" << term.exp;
92          first = false;
93      }
94      return os;
95  }

```

```

97     std::istream& operator>>(std::istream& is, Polynomial& poly) {
98         int numTerms;
99         std::cout << "輸入項數：";
100         if (!(is >> numTerms) || numTerms < 0) {
101             std::cerr << "無效的項數" << std::endl;
102             is.setstate(std::ios::failbit);
103             return is;
104         }
105
106         for (int i = 0; i < numTerms; ++i) {
107             float coef;
108             int exp;
109             std::cout << "輸入係數和指數：";
110             if (!(is >> coef >> exp)) {
111                 std::cerr << "無效的係數或指數" << std::endl;
112                 is.setstate(std::ios::failbit);
113                 return is;
114             }
115             poly.AddTerm(coef, exp);
116         }
117
118         return is;
119     }

```

```

121 int main() {
122     Polynomial p1, p2;
123     std::cout << "輸入多項式 1：" << std::endl;
124     std::cin >> p1;
125     std::cout << "輸入多項式 2：" << std::endl;
126     std::cin >> p2;
127
128     Polynomial sum = p1.Add(p2);
129     Polynomial product = p1.Mult(p2);
130
131     std::cout << "多項式 1：" << p1 << std::endl;
132     std::cout << "多項式 2：" << p2 << std::endl;
133     std::cout << "和：" << sum << std::endl;
134     std::cout << "積：" << product << std::endl;
135 }

```

```

136     float x;
137     std::cout << "輸入一個值來評估多項式：";
138     if (std::cin >> x) {
139         std::cout << "p1(" << x << ") = " << p1.Eval(x) << std::endl;
140         std::cout << "p2(" << x << ") = " << p2.Eval(x) << std::endl;
141     }
142     else {
143         std::cerr << "無效的輸入" << std::endl;
144     }
145
146     return 0;
147 }

```

## CHAPTER 3

### 效能分析

#### *Time Complexity*

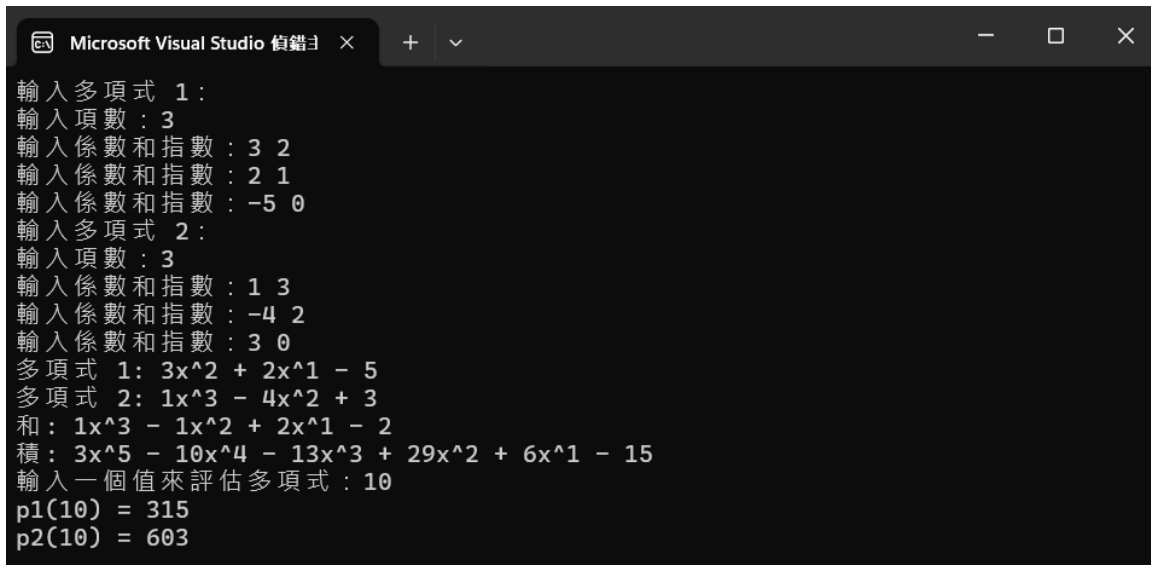
1. **AddTerm**:  $O(n)$ ，因為可能需要遍歷所有項。
2. **Add**:  $O(n+m)$ ， $n$  和  $m$  分別是兩個多項式的項數。
3. **Mult**:  $O(n*m)$ ， $n$  和  $m$  分別是兩個多項式的項數。
4. **Eval**:  $O(n)$ ， $n$  是多項式的項數。

#### *Space Complexity*

1. **AddTerm**:  $O(1)$ ，不需要額外的空間。
2. **Add**:  $O(n+m)$ ，需要儲存結果多項式。
3. **Mult**:  $O(n*m)$ ，需要儲存結果多項式。
4. **Eval**:  $O(1)$ ，僅使用常數空間

## CHAPTER 4

### 測試與驗證



```
Microsoft Visual Studio 偵錯 1  X + -
輸入多項式 1 :
輸入項數 : 3
輸入係數和指數 : 3 2
輸入係數和指數 : 2 1
輸入係數和指數 : -5 0
輸入多項式 2 :
輸入項數 : 3
輸入係數和指數 : 1 3
輸入係數和指數 : -4 2
輸入係數和指數 : 3 0
多項式 1:  $3x^2 + 2x^1 - 5$ 
多項式 2:  $1x^3 - 4x^2 + 3$ 
和:  $1x^3 - 1x^2 + 2x^1 - 2$ 
積:  $3x^5 - 10x^4 - 13x^3 + 29x^2 + 6x^1 - 15$ 
輸入一個值來評估多項式 : 10
p1(10) = 315
p2(10) = 603
```

## CHAPTER 5

### 效能量測

## CHAPTER 6

### 心得

上課上到這一章節，還是有很多地方需要再去研究復習的地方，照著課本的範例實做出來的程式碼真的有太多不完善的地方，所以我就以我原本的程式碼放上 GPT 請他幫我完善，所以我作業上面放的也是改善過後的，程式運作起來輸入輸出更清晰了

**\*(演算法驗證及過程皆參考 GPT)\***