# 資料結構 HW 2

學號:41043138
姓名:紀宇駿

August 06, 2024

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

# CH 1

## 解題說明

利用`Polynomial`類別來表示多項式，並實現加法、乘法和求值操作。這個類別能處理多項式的輸入和輸出，並使用運算符重載來簡化這些操作。

## 舉例

使用本題測試數值來舉例：poly1: 4x^4 + 3x^3 + 2x^2 + 1x^1,poly2: 5x^3 + 4x^2 + 3x^1 + 2

1. **和**: poly1 + poly2 = (3+1)x^2 + 2x + (1+4) = 4x^2 + 2x + 5
2. **積**: poly1 * poly2 = (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. **計算**: 在 x=2 時poly1(2) = 74,poly2(2)=74

# CH 2

## 演算法設計與實作

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>


// 定義表示多項式的單項式類別
class SingleTerm {
public:
    SingleTerm(float coefficient = 0, int exponent = 0) : coef(coefficient), exp(exponent) {}
    float coef; // 係數
    int exp;    // 指數
};

// 定義表示多項式的類別
class Polynomial {
public:
    Polynomial() = default;
    Polynomial(const Polynomial&) = default;
    Polynomial& operator=(const Polynomial&) = default;
    ~Polynomial() = default;

    void AppendTerm(float coefficient, int exponent); // 插入單項式
    Polynomial SumWith(const Polynomial& other) const; // 多項式相加
    Polynomial MultiplyWith(const Polynomial& other) const; // 多項式相乘
    float EvaluateAt(float value) const; // 計算多項式在某點的值
    Polynomial Derivative() const; // 計算多項式的一次導數

    friend std::ostream& operator<<(std::ostream& out, const Polynomial& poly); // 重載輸出運算符
    friend std::istream& operator>>(std::istream& in, Polynomial& poly); // 重載輸入運算符

```

```
32  private:
33      std::vector<SingleTerm> terms; // 存儲多項式的單項式
34
35      // 比較單項式的指數，用於排序
36      static bool CompareTerms(const SingleTerm& a, const SingleTerm& b) {
37          return a.exp > b.exp;
38      }
39  };
40
41  // 插入單項式到多項式中
42  void Polynomial::AppendTerm(float coefficient, int exponent) {
43      if (coefficient == 0) return; // 忽略係數為零的項
44
45      // 查找是否存在相同指數的單項式
46      for (auto& term : terms) {
47          if (term.exp == exponent) {
48              term.coef += coefficient;
49              if (term.coef == 0) {
50                  // 如果係數相加後為零，則刪除該單項式
51                  terms.erase(std::remove_if(terms.begin(), terms.end(), [exponent](const SingleTerm& t) { return t.exp == exponent; }), terms.end());
52              }
53              std::sort(terms.begin(), terms.end(), CompareTerms); // 按指數排序
54              return;
55          }
56      }
56      }
57
58      // 如果沒有相同指數的單項式，則添加新單項式
59      terms.emplace_back(coefficient, exponent);
60      std::sort(terms.begin(), terms.end(), CompareTerms); // 按指數排序
61  }
62
63  // 多項式相加
64  Polynomial Polynomial::SumWith(const Polynomial& other) const {
65      Polynomial result = *this;
66      for (const auto& term : other.terms) {
67          result.AppendTerm(term.coef, term.exp);
68      }
69      return result;
70  }
71
72  // 多項式相乘
73  Polynomial Polynomial::MultiplyWith(const Polynomial& other) const {
74      Polynomial result;
75      for (const auto& term1 : terms) {
76          for (const auto& term2 : other.terms) {
77              result.AppendTerm(term1.coef * term2.coef, term1.exp + term2.exp);
78          }
79      }
80      return result;
81  }
82
```

```
82
83  // 計算多項式在某點的值
84  float Polynomial::EvaluateAt(float value) const {
85      float result = 0;
86      for (const auto& term : terms) {
87          result += term.coef * std::pow(value, term.exp);
88      }
89      return result;
90  }
91
92  // 計算多項式的一次導數
93  Polynomial Polynomial::Derivative() const {
94      Polynomial deriv;
95      for (const auto& term : terms) {
96          if (term.exp != 0) {
97              deriv.AppendTerm(term.coef * term.exp, term.exp - 1);
98          }
99      }
100     return deriv;
101 }
102
103 // 重載輸出運算符
104 std::ostream& operator<<(std::ostream& out, const Polynomial& poly) {
105     if (poly.terms.empty()) {
106         out << "0";
107         return out;
108     }
109     bool first = true;
110     for (const auto& term : poly.terms) {
111         if (!first && term.coef > 0) out << " + ";
```

```
112         if (term.coef < 0) out << " - ";
113         if (!first && term.coef < 0) out << -term.coef;
114         else out << term.coef;
115
116         if (term.exp > 0) out << "x^" << term.exp;
117         first = false;
118     }
119     return out;
120 }
121
122 // 重載輸入運算符
123 std::istream& operator>>(std::istream& in, Polynomial& poly) {
124     int numTerms;
125     std::cout << "輸入單項式數量：";
126     if (!(in >> numTerms) || numTerms < 0) {
127         std::cerr << "無效的單項式數量" << std::endl;
128         in.setstate(std::ios::failbit);
129         return in;
130     }
131
132     for (int i = 0; i < numTerms; ++i) {
133         float coefficient;
134         int exponent;
135         std::cout << "輸入係數和指數：";
136         if (!(in >> coefficient >> exponent)) {
137             std::cerr << "無效係數或指數" << std::endl;
138             in.setstate(std::ios::failbit);
139             return in;
140         }
141         poly.AppendTerm(coefficient, exponent);
```

```
142         }
143
144         return in;
145     }
146
147 int main() {
148     Polynomial poly1, poly2;
149     std::cout << "輸入多項式 1：" << std::endl;
150     std::cin >> poly1;
151     std::cout << "輸入多項式 2：" << std::endl;
152     std::cin >> poly2;
153
154     Polynomial sum = poly1.SumWith(poly2);
155     Polynomial product = poly1.MultiplyWith(poly2);
156
157     std::cout << "多項式 1: " << poly1 << std::endl;
158     std::cout << "多項式 2: " << poly2 << std::endl;
159     std::cout << "和: " << sum << std::endl;
160     std::cout << "積: " << product << std::endl;
161
162     float x;
163     std::cout << "輸入計算這兩個多項式：";
164     if (std::cin >> x) {
165         std::cout << "poly1(" << x << ") = " << poly1.EvaluateAt(x) << std::endl;
166         std::cout << "poly2(" << x << ") = " << poly2.EvaluateAt(x) << std::endl;
167     }
168     else {
169         std::cerr << "無效輸入" << std::endl;
170     }
171
172     Polynomial deriv1 = poly1.Derivative();
173     Polynomial deriv2 = poly2.Derivative();
174
175     std::cout << "多項式 1 的導數: " << deriv1 << std::endl;
176     std::cout << "多項式 2 的導數: " << deriv2 << std::endl;
177
178     return 0;
179 }
180
```

# CH 3

## 效能分析

### Time Complexity

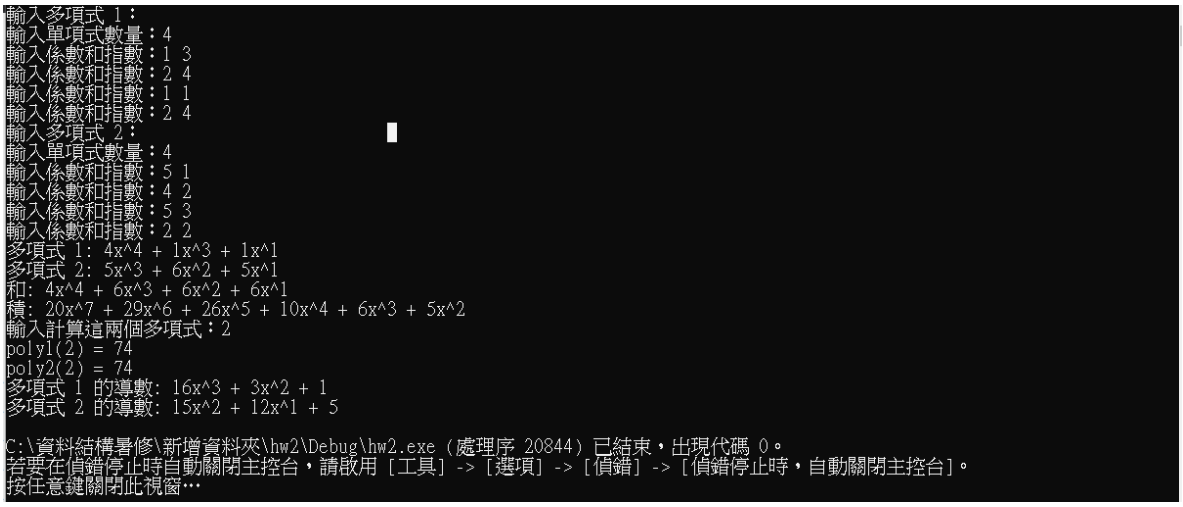
1. **AppendTerm:** O(n), n 是單項式的數量。
2. **SumWith**: O(nlog n)
3. **MultiplyWith**: O(nm)，n 和 m 是兩個多項式的單項式數量。
4. **EvaluateAt**: O(n)，n 是單項式的數量。
5. **Derivative**: O(n), n 是單項式的數量。

### Space Complexity

1. **AppendTerm:** O(n), n 是單項式的數量。
2. **SumWith**: O(n + m)
3. **MultiplyWith**: O(nm)
4. **EvaluateAt**: O(1)
5. **Derivative**: O(n)

## CH 4

### 測試與驗證




## CH 5

### 效能量測

## CH 6

### 心得

這次暑修速度比起以往上課時來的快許多，因此更能知道自己不足需要加強的地方是哪些，這次作業是實作多項式類別與其相關操作等等，都是我必須要學會的東西，也透過此次作業讓我有更進一步的認識，在自己寫程式上有很多不足的地方透過使用chatgpt才得以完成，也讓我更加體認到自己在程式方面上的實力不夠，我會更加努力的。

寫程式以及計算複雜度有透過chatgpt進行輔助