

# 資料結構實習

10/06 作業報告

多項式 ADT 實作

班級：資訊二甲

學號：D1109023

姓名：楊孟憲

## **Contents**

<b>1</b>	<b>引言</b>	<b>3</b>
<b>2</b>	<b>題目敘述</b>	<b>4</b>
<b>3</b>	<b>作法</b>	<b>4</b>
<b>4</b>	<b>執行結果</b>	<b>10</b>
<b>5</b>	<b>心得與討論</b>	<b>12</b>

# 1 引言

今天要使用 **struct** (結構) 來實作多項式的操作運算。

```
1 #define MAX 100
2
3 struct polynomial{
4     int coef ; //多項式的係數
5     int expon ; //多項式的指數
6 };
7
8 struct MyPoly{
9     polynomial terms[MAX] ; //多項式陣列
10    int size ; //多項式大小
11    MyPoly() ; //建構子 初始化 size = 0
12    MyPoly(char*) ; //建構子 讀入檔案
13    void ShowPoly( ) ; //印出多項式內容
14    MyPoly Add(MyPoly) ; //多項式相加
15    void SingleMult(int) ; //將多項式乘上一係數
16    int Lead_Exp() ; //回傳多項式中最大指數的次方
17    void Attach(float , int) ; //新增 1 個項式到多項式中
18    void Remove(int) ; //刪除多項式中的某一指數
19    MyPoly Mult(MyPoly) ; //多項式相乘
20 };
```

## 2 題目敘述

題意說明: 撰寫一個程式，讀取 B1.txt 以及 B2.txt 多項式內容，並包含以下功能。請提供選單的方式，讓使用者能夠對此多項式 ADT 做操作：

1. 讀入多項式
2. 印出多項式內容
3. 多項式相加
4. 多項式乘上一數值
5. 印出多項式中最大指數的係數
6. 新增項式
7. 刪除多項式中的項
8. 多項式相乘

## 3 作法

因為這個題目需要不斷輸入你所需要的功能 (1 ~ 8)，所以使用 while loop 來做最外圍的流程控制，依照不同的輸入來呼叫相對應功能的函式。以下介紹代碼中的函式 ( 程式碼講解在註解裡)

1. void MyPoly::refresh(const char\* filename)

```

1 void MyPoly::refresh(const char* filename) {
2     idx = 0; // 初始化 idx
3     ifstream ifs(filename, ifstream::in);
4     polynomial input;
5     while(ifs >> input.coef >> input.expon) {
6         terms[idx] = input; // 將每一筆資料存入 terms[idx]
7         idx++; // 每次做完 idx + 1
8     }
9     ifs.close(); // 關閉檔案
10 }

```

## 2. void MyPoly::ShowPoly()

```

1 void MyPoly::ShowPoly() {
2     // 遍歷迴圈內所有的項次
3     for(int i = 0; i < idx; i++) {
4         cout << terms[i].coef << "X^" << terms[i].expon << "+";
5         // 輸出係數以及指數
6     }
7     cout << "\b \n"; // 因為會多輸出一個 + 號，所以使用 "\b" + " "
8     // 把它蓋掉。
9     return;
10 }

```

## 3. mypoly mypoly::add(mypoly poly2)

```

1 MyPoly MyPoly::Add(MyPoly poly2) {
2     MyPoly res; // res 來儲存相加結果
3     int idxR = 0, idxL = 0; // idxR 遍歷 idx, idxL 遍歷 ploy.idx
4     while(idxR < idx && idxL < poly2.idx) { // 當 idxR < idx 且
5         // idxL < poly.idx 時執行迴圈
6         if(terms[idxR].expon > poly2.terms[idxL].expon) { // 判斷指數大小
7             res.terms[res.idx] = terms[idxR];
8         }
9     }
10 }

```

```

7     idxR++; // 向右移
8 }
9 else if (terms[idxR].expon < poly2.terms[idxL].expon) {
10     res.terms[res.idx] = poly2.terms[idxL];
11     idxL++; // 向右移
12 }
13 else {
14     res.terms[res.idx] = {terms[idxR].coef + poly2.terms[idxL].coef, terms[idxR].expon};
15     idxR++; // 向右移
16     idxL++; // 向右移
17 }
18 res.idx++; // 每次做完 res.terms 都會多一位，所以 idx++
19 }
20
21 while (idxR < idx) {
22     res.terms[res.idx] = terms[idxR];
23     idxR++;
24     res.idx++;
25 }
26
27 while (idxL < poly2.idx) {
28     res.terms[res.idx] = poly2.terms[idxL];
29     idxL++;
30     res.idx++;
31 }
32
33 return res;
34 }

```

#### 4. void MyPoly::SingleMult(int n)

```

1 void MyPoly::SingleMult(int n) {

```

```

2   for(int i = 0; i < idx; i++) { // 遍歷每一個項次
3       terms[i].coef *= n; // 將係數乘上 n
4   }
5   return;
6 }

```

## 5. int MyPoly::Lead \_\_ Exp()

```

1 int MyPoly::Lead_Exp() {
2     return terms[0].expon; // 因為保證多項式是降序的排列方式，所以
        取第一個
3 }

```

## 6. void MyPoly::Attach(int coef, int expon)

```

1 void MyPoly::Attach(int coef, int expon) {
2     // 尋找同一個指數，如果有，將係數相加
3     for(int i = 0; i < idx; i++) {
4         if(terms[i].expon == expon) {
5             terms[i].coef += coef;
6             return; // 找到就直接結束
7         }
8     }
9     // 如果沒有找到，新增在尾巴
10    terms[idx++] = {coef, expon};
11    sort(terms, terms + idx, cmp); // 自定義排序 bool cmp(
        polynomial, polynomial);
12    return;
13 }

```

## 7. void MyPoly::Remove(int expon)

```

1 void MyPoly::Remove(int expon) {

```

```

2  int re_idx = -1; // re_idx 儲存目標項次的 idx ，如果沒有找到就
   會是 -1
3  MyPoly temp; // 儲存新的 terms
4
5  for(int i = 0; i < idx; i++) {
6      if(terms[i].expon == expon) {
7          re_idx = i; // 更新 re_idx
8          break; // 如果找到就可以退出了
9      }
10 }
11
12 if(re_idx == -1) {
13     // 如果沒有找到，就提示使用者
14     cout << "The expon isn't in the function\n";
15     return;
16 }
17
18
19 for(int i = 0; i < idx; i++) { // 遍歷 terms
20     if(i == re_idx) { // 如果 idx == re_idx ，代表不需儲存
21         continue;
22     }
23
24     temp.terms[temp.idx++] = terms[i];
25 }
26
27 idx = 0; // 初始化 terms
28
29 // 將 terms 更新為 temp.terms
30 for(int i = 0; i < temp.idx; i++) {
31     terms[idx++] = temp.terms[i];
32 }
33

```



```

34     cout << "Success\n";
35     return;
36 }

```

## 8. MyPoly MyPoly::Mult(MyPoly poly2)

```

1 MyPoly MyPoly::Mult(MyPoly poly2) {
2     MyPoly result, result2; // result 儲存相乘結果， result2 儲存
    合併後的結果
3
4     // 遍歷每一個組合
5     for(int i = 0; i < idx; i++) {
6         for(int j = 0; j < poly2.idx; j++) {
7             polynomial current = {terms[i].coef * poly2.terms[j].coef,
                terms[i].expon + poly2.terms[j].expon};
8             result.terms[result.idx] = current; // 相乘後，存到 result
                .terms
9             result.idx++;
10        }
11    }
12
13    sort(result.terms, result.terms + result.idx, cmp); // 排序
14
15    // 合併同一幂次
16    polynomial cur = result.terms[0];
17    for(int i = 1; i < result.idx; i++) {
18        if(result.terms[i].expon == cur.expon) {
19            cur.coef += result.terms[i].coef;
20        } else {
21            result2.terms[result2.idx++] = cur;
22            cur = result.terms[i];
23        }
24    }

```

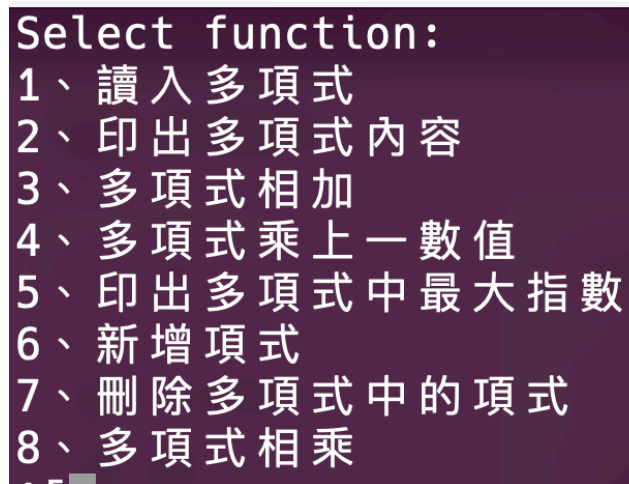
```

25
26     result2.terms[result2.idx++] = cur;
27
28     return result2; // 回傳 result2, terms = result2
29 }

```

## 4 執行結果

輸入輸出結果：(每次輸入後都會輸出結果)



```

Select function:
1、讀入多項式
2、印出多項式內容
3、多項式相加
4、多項式乘上一數值
5、印出多項式中最大指數
6、新增項式
7、刪除多項式中的項式
8、多項式相乘
^5

```

Figure 1: 選單



```

3X^20+2X^5+4X^0

```

Figure 2: 輸入 2

```
3X^20+2X^5+1X^4+10X^3+3X^2+5X^0
```

Figure 3: 輸入 3

```
input a number: 3  
9X^20+6X^5+12X^0
```

Figure 4: 輸入 4

```
20
```

Figure 5: 輸入 5

```
Please input the coef: 10  
Please input the expon: 100  
10X^100+3X^20+2X^5+4X^0
```

Figure 6: 輸入 6

```
Please input the expon you want to remove: 20  
Success  
2X^5+4X^0
```

Figure 7: 輸入 7

```
3X^24+30X^23+9X^22+3X^20+2X^9+20X^8+6X^7+2X^5+4X^4+40X^3+12X^2+4X^0
```

Figure 8: 輸入 8

## 5 心得與討論

這次的功課實作的部分非常多，在操作不同成員的陣列也要非常小心，陣列索引值需注意現在的變數屬於哪一個而去指定。整體來說我覺得可以優化的部分有：陣列可以更改成可以擴充的 `vector` 等，如此一來，也比較不會操作到不同成員的索引值，而且在做擴充或是刪除的動作也比較簡單。以及在 `struct` 內可以自行定義 `operator` 排序的方式，這樣就不需每次操作完都排序過。我覺得這次作業能讓我更加了清楚構變數以及函示的使用。