

資料結構報告

姓名:童呈偉

日期:2024/08/04

目錄

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

解題說明

題目翻譯如下：

實作 Polynomial 類別，其抽象資料型態 (ADT) 和私有資料成員分別如圖 1 和圖 2 所示。

撰寫 C++ 函數來輸入和輸出如圖 2 所示的多項式。你的函數應該重載 << 和 >> 運算符。

根據題意和範例圖，利用ChatGPT創建類別跟設計函式如下：

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  class Polynomial; //定義類別 Polynomial
6
7  class Term //定義類別 Term
8  {
9
10     friend class Polynomial; //Polynomial可存取Term的私有成員
11     friend ostream& operator<<(ostream&, const Polynomial&); // << 可存取 Term的私有成員
12     friend istream& operator>>(istream&, Polynomial&); // >> 可存取Term的私有成員
13 private://的私有成員
14     float coef; //多項式係數
15     int exp; //多項式指數
16 };
17
```

fig1.1類別創建:Polynomial.cpp

```

/* 重載運算符，將物件Polynomial輸出到ostream
os為輸出流，如cout 返回值為 ostream&
*/
ostream& operator<<(ostream& os, const Polynomial& poly)
{
    for (int i = 0; i < poly.terms; i++)//遊歷每一項
    {
        if (i > 0 && poly.termArray[i].coef > 0) os << " + ";//當目前項不是第一項且係數為正時增添 + 號
        os << poly.termArray[i].coef << "x^" << poly.termArray[i].exp;//當前係數和指數輸出到os
    }

    return os;//回傳os
}

```

fig1.2<<運算符:Polynomial.cpp

```

230
231 /* 重載運算符，將物件Polynomial輸入到 istream
232 is為輸入流，如cin 返回值為 ostream&
233 */
234 istream& operator>>(istream& is, Polynomial& poly)
235 {
236     cout << "輸入項數: ";
237     is >> poly.terms;//讀入係數並儲存在 poly.terms
238     if (poly.terms > poly.capacity)
239     {
240         poly.resize(poly.terms);//如果超出容量就調整並使其容納
241     }
242
243     for (int i = 0; i < poly.terms; i++) //遊歷每一項
244     {
245         cout << "輸入係數跟指數: ";
246         is >> poly.termArray[i].coef >> poly.termArray[i].exp;//讀入係數跟指數並儲存
247     }
248
249     return is;//回傳
250 }

```

fig1.3>>運算符:Polynomial.cpp

設計與實作

```
251
252  ✓ int main()
253      {
254          Polynomial p1, p2, p3;
255          cout << "輸入第一個多項式:\n";
256          cin >> p1;
257
258          cout << "輸入第二個多項式:\n";
259          cin >> p2;
260
261          p3 = p1.add(p2);
262          cout << "和: " << p3 << endl;
263
264          p3 = p1.mult(p2);
265          cout << "積: " << p3 << endl;
266
267          return 0;
268      }
269
```

fig2.1主函式:Polynomial.cpp

```
--
80 // 多項式加法，將當前多項式與多項式poly相加，並返回新物件Polynomial 表示和
81  ✓ Polynomial Polynomial::add(const Polynomial& poly)
82      {
83
84          Polynomial result; //初始化多項式 result表示和
85          int i = 0, j = 0; //i為當前多項式，j為poly多項式
86
87          while (i < terms && j < poly.terms) //當還有剩餘的項，進行迴圈合併
88          {
89              if (termArray[i].exp == poly.termArray[j].exp) //情況1:指數相等，則係數相加
90              {
91                  float coef = termArray[i].coef + poly.termArray[j].coef; //係數(coef)相加
92
93                  if (coef != 0) //結果係數不為0則將新項添加到result
94                  {
95                      if (result.terms == result.capacity) //檢查result是否有足夠容量
96                      {
97                          result.resize(result.capacity * 2); //動態調整 result的容量，此為擴充為原本的兩倍
98                      }
99                      result.termArray[result.terms].coef = coef; //將新係數值儲存在result
100                      result.termArray[result.terms++].exp = termArray[i].exp; //儲存新指數，並更新計數器，將結果多項式中的項數增加一，指向下一個可用位置。
101                  }
102                  i++; //將兩個多項式移動到下一項
103                  j++;
104              }
105          }
106      }
107
108
```

fig2.2加法函式1指數相同情況:Polynomial.cpp

```

109         else if (termArray[i].exp > poly.termArray[j].exp)//當前項數較大
110         {
111
112             if (result.terms == result.capacity)
113             {
114                 result.resize(result.capacity * 2);
115             }
116             //將當前多項式(termArray) 的項，直接添加到result中，並更新計數器
117             result.termArray[result.terms++] = termArray[i++];
118         }
119         else//poly項數較大
120         {
121
122             if (result.terms == result.capacity)
123             {
124                 result.resize(result.capacity * 2);
125             }
126             //將poly多項式的項，直接添加到result中，並更新計數器
127             result.termArray[result.terms++] = poly.termArray[j++];
128
129         }
130
131     }

```

fig2.3加法函式2指數不相同情況:Polynomial.cpp

```

132 //處理剩餘項
133 while (i < terms) //將當前多項式(termArray)還有剩餘
134
135     {
136         if (result.terms == result.capacity)
137             {
138                 result.resize(result.capacity * 2);
139             }
140         //直接添加到result中，並更新計數器及移動到下一項
141         result.termArray[result.terms++] = termArray[i++];
142     }
143
144 while (j < poly.terms) // poly多項式還有剩餘
145     {
146         if (result.terms == result.capacity)
147             {
148                 result.resize(result.capacity * 2);
149             }
150         //直接添加到result中，並更新計數器及移動到下一項
151         result.termArray[result.terms++] = poly.termArray[j++];
152     }
153
154 return result;//回傳結果
155 }
---
```

fig2.4加法函式3還有剩餘項情況:Polynomial.cpp

```

157 // 多項式乘法，將當前多項式與多項式poly相乘，並返回新物件Polynomial 表示積
158 ✓ Polynomial Polynomial::mult(const Polynomial& poly)
159 {
160
161     Polynomial result;//初始化result
162
163     for (int i = 0; i < terms; i++) //遊歷當前多項式的每一項
164     {
165
166         for (int j = 0; j < poly.terms; j++)//遊歷poly多項式的每一項
167         {
168
169             float coef = termArray[i].coef * poly.termArray[j].coef;//係數成積
170             int exp = termArray[i].exp + poly.termArray[j].exp;//指數和
171             bool found = false;//紀錄是否找到相同項，初始化為無
172
173             for (int k = 0; k < result.terms; k++) //遊歷result多項式的每一項
174             {
175
176                 if (result.termArray[k].exp == exp) //比較，如果找到了
177                 {
178
179                     result.termArray[k].coef += coef;//係數相加
180                     found = true;//更新標誌函數，即找到了
181                     break;//跳出迴圈
182
183                 }
184             }
185

```

fig2.5乘法函式1新項有相同指數情況:Polynomial.cpp


```

185
186         if (!found) //沒找到
187             {
188
189                 if (result.terms == result.capacity)
190                     {
191                         result.resize(result.capacity * 2);
192                     }
193                 //更新項數跟指數，將乘積完的項數跟指數給予新項
194                 result.termArray[result.terms].coef = coef;
195                 result.termArray[result.terms++].exp = exp; //自增項數，確保之後不會被新項覆蓋
196
197             }
198     }
199 }
200 return result; //回傳結果
201 }
202

```

fig2.6乘法函式2新項為全新指數情況:Polynomial.cpp

效能分析

n 是當前 Polynomial 的項數，
m 是 poly 的項數。

1. Polynomial::add()

時間複雜度：

while (i < terms && j < poly.terms) 這段主要迴圈的複雜度是 $O(n+m)$ ，其中兩個 while 迴圈的複雜度也是 $O(n+m)$ 。

空間複雜度：

主要分配記憶體的地方是 result，其內部 termArray 會根據需要動態擴展。由於新建了一個 result 多項式，空間複雜度是 $O(n+m)$ 。

2. Polynomial::mult()

時間複雜度：

外層的 for 迴圈執行

n 次，內層的 for 迴圈執行

m 次，這使得總的時間複雜度為

$O(n \cdot m)$ 。

空間複雜度：

由於新建了一個 result 多項式，空間複雜度是 $O(n \cdot m)$ 。

3. Polynomial::eval()

時間複雜度：

單個 for 迴圈，迭代 terms 次數，因此時間複雜度是 $O(n)$

該函數只使用了一些臨時變數，空間複雜度是 $O(1)$ 。

4. Polynomial::resize()

時間複雜度：

for 迴圈拷貝原來的 termArray，時間複雜度是 $O(n)$ ，

空間複雜度：

新的 Term 陣列的大小是 newCapacity，所以空間複雜度是

$O(\text{newCapacity})$ 。

主程式 (main function)

時間複雜度：

輸入兩個多項式，時間複雜度為 $O(n+m)$ 。

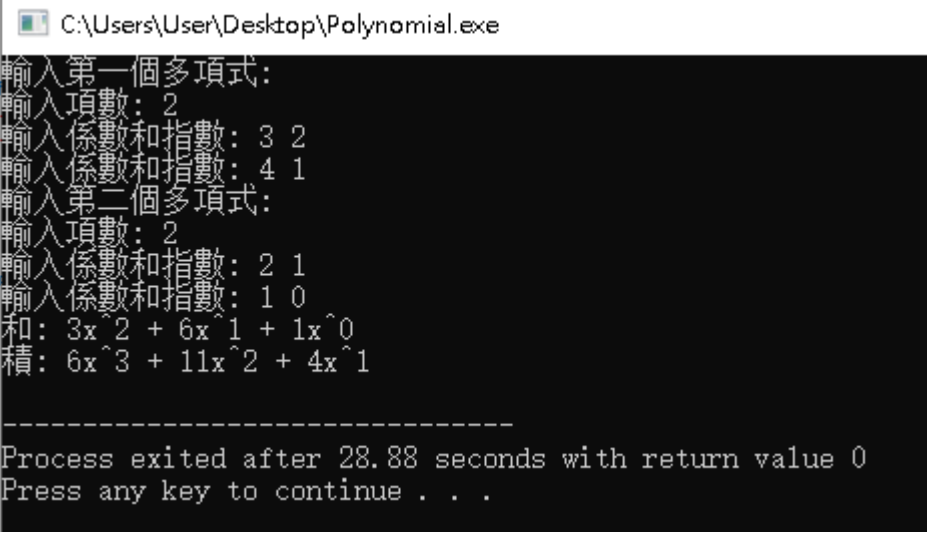
執行 add 和 mult 操作，總時間複雜度為 $O(n \cdot m)$ 。

空間複雜度：

主要來自於三個 Polynomial 物件，空間複雜度為

$O(n+m+n \cdot m)$ 。

測試與驗證



```
C:\Users\User\Desktop\Polynomial.exe
輸入第一個多項式:
輸入項數: 2
輸入係數和指數: 3 2
輸入係數和指數: 4 1
輸入第二個多項式:
輸入項數: 2
輸入係數和指數: 2 1
輸入係數和指數: 1 0
和: 3x^2 + 6x^1 + 1x^0
積: 6x^3 + 11x^2 + 4x^1
-----
Process exited after 28.88 seconds with return value 0
Press any key to continue . . .
```

fig4.1多項式計算結果:Polynomial.exe

第一個多項式為: $3x^2 + 4x$

第二個多項式為: $2x + 1$

加法和為: $3x^2 + 6x + 1$

按照加法函式, 指數相同的情況下, 直接相加係數, 並儲存在結果中

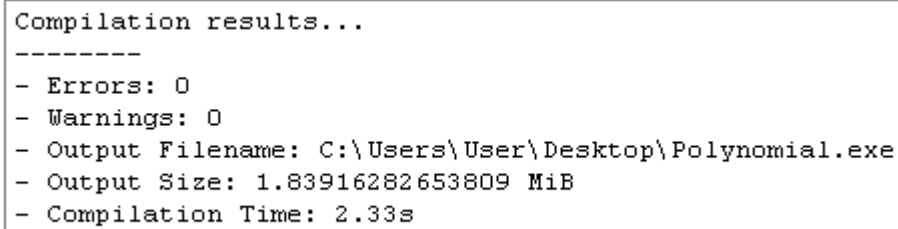
乘法積為: $(3x^2 + 4x) * (2x + 1)$

$= (6x^3 + 3x^2) + (8x^2 + 4x)$

$= 6x^3 + 11x^2 + 4x$

按照乘法函式, 當 $3x^2$ 和 $2x$ 相乘時, 由於指數是新, 則直接動態配置一個新位置儲存新項的係數跟指數, 而 $3x^2$ 和 $8x^2$ 由於指數相同直接相加一起

效能量測

A screenshot of a terminal window showing the compilation results for a program named Polynomial.exe. The text is as follows:

```
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\User\Desktop\Polynomial.exe
- Output Size: 1.83916282653809 MiB
- Compilation Time: 2.33s
```

fig5.1編譯時間:Polynomial.exe

心得

這一次縱然有AI輔助卻還是錯誤居多，在不斷，問題多是最後的輸出表達式，沒辦法顯示出正常的多項式，經過AI排錯跟自己測試之後總算能順利匯出，同時也註解程式碼增加記憶，防止忘記函式用處。