

一元稀疏多项式运算器

一、实习题目与要求

1.1 一元稀疏多项式运算器

- 输入并建立两个多项式；
- 多项式a与b相加，建立和多项式c；
- 多项式a与b相减，建立差多项式d；
- 输入多项式a, b, c, d。

输出格式：比如多项式a为： $A(x)=c_1x^{e_1}+c_2x^{e_2}+\dots+c_mx^{e_m}$ (c_i 和 e_i 分别为第 i 项的系数和指数，且按各项指数的升幂排列，即 $0\leq e_1<e_2<\dots<e_m$)。

二、需求分析

2.1 问题描述

设计一个一元稀疏多项式简单运算器。

在输入栏中依次输入多项式A和B，将其转换为功能要求中的格式，选择要进行的运算（加法或减法）。先比较指数，指数相同则进行系数运算，如果两系数运算结果为0，则结果多项式中不存储该指数项。

2.2 系统环境

Windows、Linux、macOS等能运行C++程序的系统

2.3 运行要求

Visual Studio 2017

三、概要设计

3.1 数据结构的设计

本程序中利用带头结点的单链表来存储多项式，是线性结构。

3.2 存储结构的设计

利用带头结点的单链表来存储多项式。

3.3 算法设计

依次输入多项式A每一项的系数和指数，以输入0、-1结束，继续输入多项式B每一项的系数和指数，以输入0、-1结束，对输入的多项式A、B进行相加运算得到多项式C或进行相减运算得到多项式D，输出多项式C或D。

3.4 模块设计



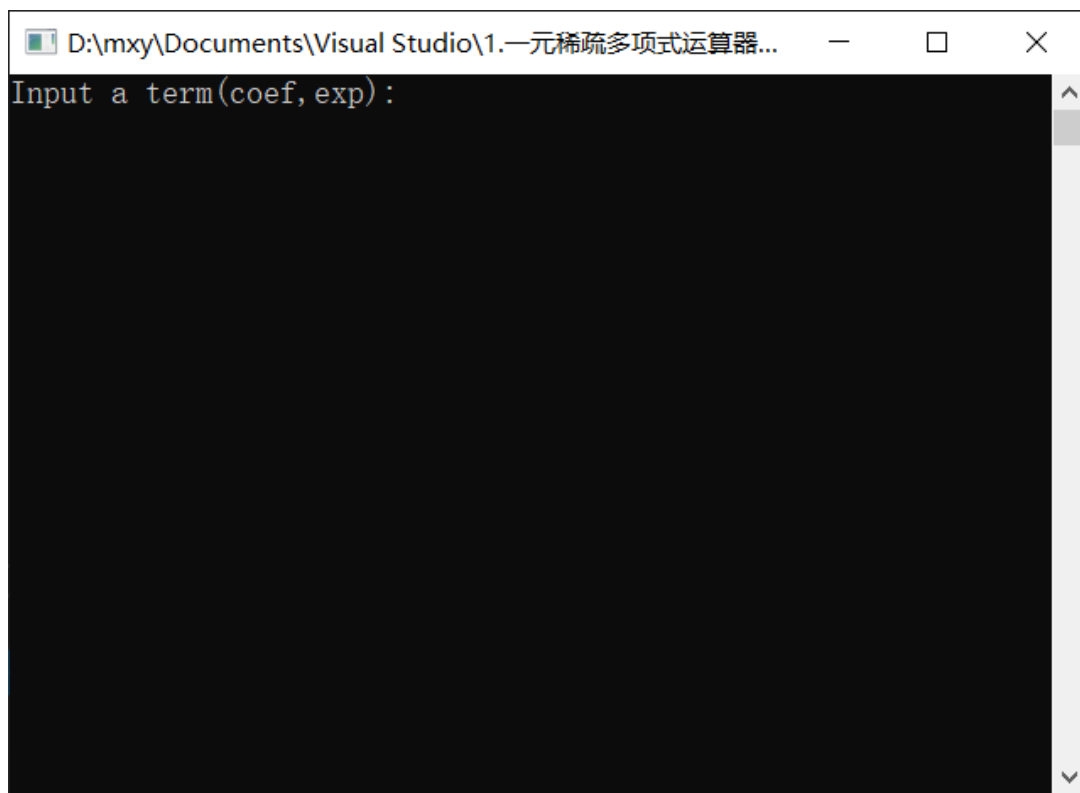
四、详细设计

4.1 类的函数成员和成员函数的设计

- 结构Term
 - 成员函数：InsertAfter (float c, int e) 链表尾插入
 - 函数成员：
 - float coef; 系数
 - int exp; 指数
- 类Polynomial成员函数：
 - Polynomial () 构造函数，建立空链
 - Polynomial (Polynomial&R) 复制构造函数
 - int maxOrder () 计算最大阶数
 - friend ostream&operator<<(ostream&,const Polynomial&) 输出多项式链表
 - friend istream&operator>>(istream&,Polynomial&) 导入输入的系数和指数
 - friend Polynomial operator + (Polynomial&,Polynomial&)加法的重载
 - friend Polynomial operator - (Polynomial&,Polynomial&)减法的重载

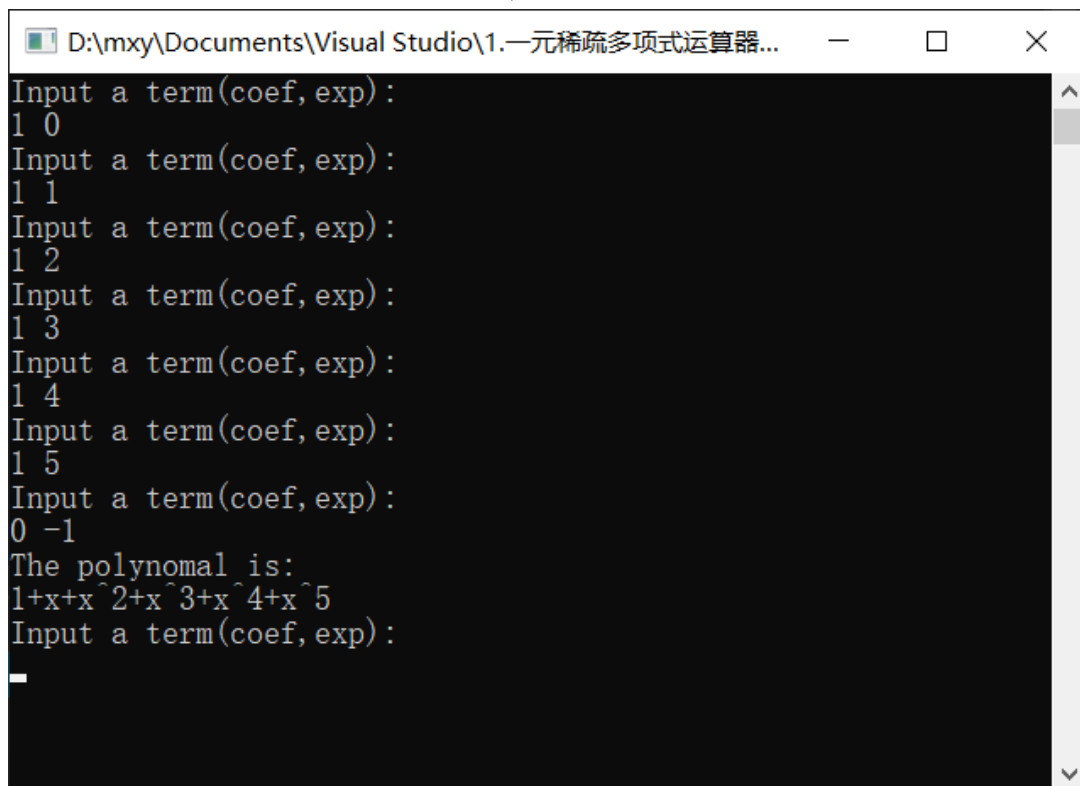
4.2 界面设计

4.2.1 初始：



4.2.2 输入多项式A

$1+x+x^2+x^3+x^4+x^5$ (当输入0 -1时, 结束输入)



4.2.3 输入多项式B

$-x^3-x^4$

```
D:\mxy\Documents\Visual Studio\1.一元稀疏多项式运算器...
Input a term(coef, exp):
0 -1
The polynomial is:
1+x+x^2+x^3+x^4+x^5
Input a term(coef, exp):
-1 3
Input a term(coef, exp):
-1 4
Input a term(coef, exp):
0 -1
The polynomial is:
-x^3-x^4
选择进行的运算方式:
    1. 加法
    2. 减法
Input:
```

4.2.4 选择运算方式

选择加法运算，输出运算结果

```
Microsoft Visual Studio 调试控制台
1+x+x^2+x^3+x^4+x^5
Input a term(coef, exp):
-1 3
Input a term(coef, exp):
-1 4
Input a term(coef, exp):
0 -1
The polynomial is:
-x^3-x^4
选择进行的运算方式:
    1. 加法
    2. 减法
Input:1
The polynomial is:
1+x+x^2+x^5

D:\mxy\Documents\Visual Studio\1.一元稀疏多项式运算器\Debug\1
.一元稀疏多项式运算器.exe (进程 3444) 已退出，返回代码为：0。
按任意键关闭此窗口...
```

4.3 其他模块设计与实现

4.3.1 系数为0的情况

系数为0的项，不做存储

例如：系数为0，指数任意，该项值即为0，不做存储。

4.3.2 系数为±1的情况

只显示正负不显示字符“1”

例如：系数为-1，指数为2，输出时应输出“-x^2”，而不是“-1x^2”。

4.3.3 减法运算

减法运算中多余的项系数取反后存储

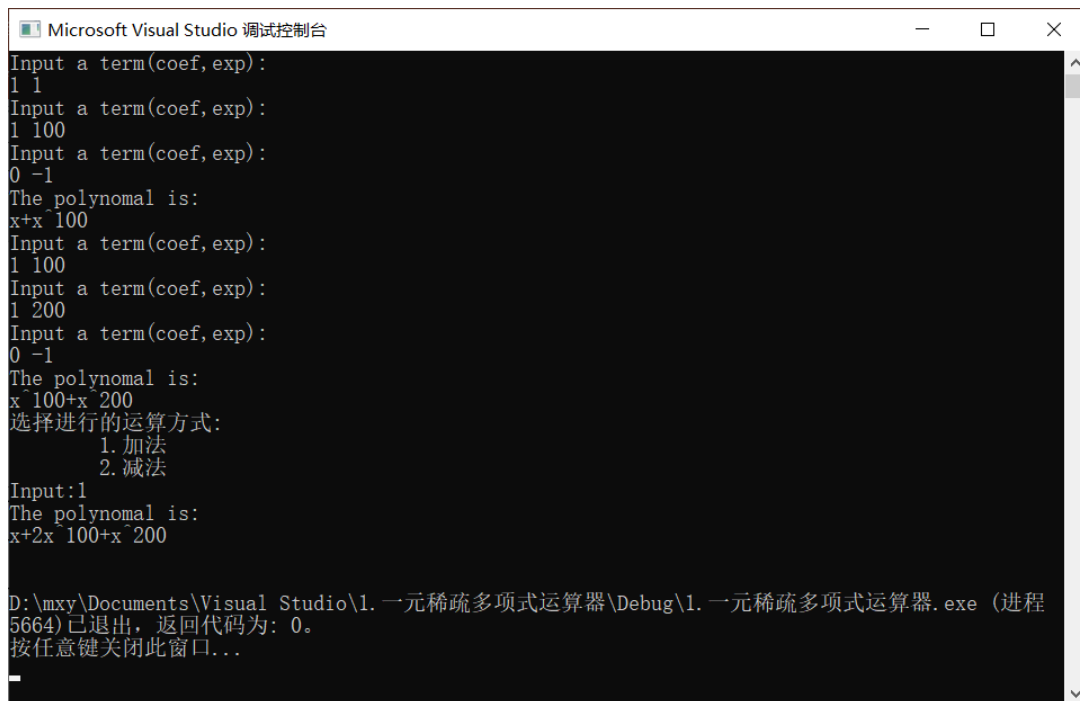
例如：多余项系数为1，指数为2，存到链表尾时应将系数取反存为-1，指数不变为

2

五、测试

5.1 测试一

- $(x+x^{100})+(x^{100}+x^{200})=(x+2x^{100}+x^{200})$



```
Microsoft Visual Studio 调试控制台
Input a term(coef, exp):
1 1
Input a term(coef, exp):
1 100
Input a term(coef, exp):
0 -1
The polynomial is:
x+x^100
Input a term(coef, exp):
1 100
Input a term(coef, exp):
1 200
Input a term(coef, exp):
0 -1
The polynomial is:
x^100+x^200
选择进行的运算方式:
1. 加法
2. 减法
Input:1
The polynomial is:
x+2x^100+x^200

D:\mxy\Documents\Visual Studio\1. 一元稀疏多项式运算器\Debug\1. 一元稀疏多项式运算器.exe (进程
5664) 已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

5.2 测试二

- $(2x+5x^8-3x^{11})+(7-5x^8+11x^9)=(7+2x+11x^9-3^{11})$

```
Microsoft Visual Studio 调试控制台
Input a term(coef,exp):
2 1
Input a term(coef,exp):
5 8
Input a term(coef,exp):
-3 11
Input a term(coef,exp):
0 -1
The polynomial is:
2x+5x^8-3x^11
Input a term(coef,exp):
7 0
Input a term(coef,exp):
-5 8
Input a term(coef,exp):
11 9
Input a term(coef,exp):
0 -1
The polynomial is:
7-5x^8+11x^9
选择进行的运算方式:
1. 加法
2. 减法
Input:1
The polynomial is:
7+2x+11x^9-3x^11

D:\mxy\Documents\Visual Studio\1. 一元稀疏多项式运算器\Debug\1. 一元稀疏多项式运算器.exe (进程
12332) 已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

5.3 测试三

- $(6x^{-3}-x+4.4x^2-1.2x^9)-(-6x^{-3}+4.4x^2+7.8x^{15})=(12x^{-3}-x-1.2x^9-7.8x^{15})$

```
Microsoft Visual Studio 调试控制台
Input a term(coef,exp):
6 -3
Input a term(coef,exp):
-1 1
Input a term(coef,exp):
4.4 2
Input a term(coef,exp):
-1.2 9
Input a term(coef,exp):
0 -1
The polynomial is:
6x^-3-x+4.4x^2-1.2x^9
Input a term(coef,exp):
-6 -3
Input a term(coef,exp):
4.4 2
Input a term(coef,exp):
7.8 15
Input a term(coef,exp):
0 -1
The polynomial is:
-6x^-3+4.4x^2+7.8x^15
选择进行的运算方式:
1. 加法
2. 减法
Input:2
The polynomial is:
12x^-3-x-1.2x^9-7.8x^15

D:\mxy\Documents\Visual Studio\1. 一元稀疏多项式运算器\Debug\1. 一元稀疏多项式运算器.exe (进程
15776) 已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

六、附录

1.1 程序源代码:

```

1  #include "pch.h"
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  struct Term {
7      float coef;                //系数
8      int exp;                   //指数
9      Term *link;
10     Term(float c, int e, Term *next = NULL)
11     {
12         coef = c; exp = e; link = next;
13     }
14     Term *InsertAfter(float c, int e);
15     friend ostream& operator<<(ostream&, const Term&);
16 };
17
18 class Polynomial {
19 public:
20     Polynomial() { first = new Term(0, -1); }    //建立空链表
21     Polynomial(Polynomial&R);
22     int maxOrder();
23     Term *getHead() const { return first; }    //取得多项式单链表的表头指针
24 private:
25     Term *first;
26     friend ostream& operator<<(ostream&, const Polynomial&);
27     friend istream& operator>>(istream&, Polynomial&);
28     friend Polynomial operator+(Polynomial&, Polynomial&);
29     friend Polynomial operator-(Polynomial&, Polynomial&);
30 };
31
32 Term*Term::InsertAfter(float c, int e) {        //在当前由this指针指示的项后面
    面插入一个新项
33     link = new Term(c, e, link);
34     return link;
35 };
36
37 ostream&operator<<(ostream&out, const Term&x) { //输出一个项x的内容到输出流
    out中
38     if (x.coef == 0.0) return out;    //系数为0返回输出流
39     if (x.coef != 1 && x.coef != -1) out << x.coef;    //系数不为1或-1, 输出
    系数
40     switch (x.exp) {
41     case 0: {
42         if (fabs(x.coef) == 1) out << x.coef;    //如果指数为0且系数为1
    或-1, 直接输出系数
43     } break;
44     case 1: {
45         if (x.coef == -1) out << "-x";    //指数为1的情况下, 如果系数为-1则输出-x

```

```

46         else out << "x";          //否则输出x
47     }break;
48     default: {
49         if (x.coef == -1) out << "-x^" << x.exp;
50         else out << "x^" << x.exp;
51     }break;
52     }
53     return out;
54 };
55
56 Polynomial::Polynomial(Polynomial&R) {          //用已有多项式对象R初始化当前
多项式对象
57     first = new Term(0, -1);
58     Term *destptr = first, *srcptr = R.getHead()->link;
59     while (srcptr != NULL) {
60         destptr->InsertAfter(srcptr->coef, srcptr->exp);
61         srcptr = srcptr->link;
62         destptr = destptr->link;
63     }
64 };
65
66 int Polynomial::maxOrder() {                  //计算最大阶数
67     Term *current = first;
68     while (current->link != NULL) current = current->link;
69     return current->exp;
70 };
71
72 istream&operator>>(istream&in, Polynomial&x) { //输入多项式
73     Term *rear = x.getHead();
74     float c;
75     int e;
76     while (1) {
77         cout << "Input a term(coef,exp):" << endl;
78         in >> c >> e;
79         if (c == 0 && e == -1) break;          //系数为0, 指数为1时输
入结束
80         rear = rear->InsertAfter(c, e);
81     }
82     return in;
83 };
84
85 ostream&operator<<(ostream&out, Polynomial&x) { //输出得到的和/差多项式链表
86     Term *current = x.getHead()->link;
87     cout << "The polynomial is:" << endl;
88     bool isEnd = true;
89     while (current != NULL) {
90         if (isEnd == false && current->coef > 0.0) out << "+";    //输入未结
束则输出一个“+”
91         isEnd = false;
92         out << *current;      //输出当前项
93         current = current->link;

```



```

94     }
95     out << endl;
96     return out;
97 };
98
99 Polynomial operator + (Polynomial&A, Polynomial&B) { //加法的重载
100     Term*pa, *pb, *pc, *p;
101     float temp;
102     Polynomial C; pc = C.first;
103     pa = A.getHead()->link; pb = B.getHead()->link;
104     while (pa != NULL && pb != NULL) {
105         if (pa->exp == pb->exp) {
106             temp = pa->coef + pb->coef;
107             if (fabs(temp) > 0.001) //系数相加不为0
108                 pc = pc->InsertAfter(temp, pa->exp);
109             pa = pa->link; pb = pb->link;
110         }
111         else if (pa->exp < pb->exp) {
112             pc = pc->InsertAfter(pa->coef, pa->exp);
113             pa = pa->link;
114         }
115         else {
116             pc = pc->InsertAfter(pb->coef, pb->exp);
117             pb = pb->link;
118         }
119     }
120     if (pa != NULL) p = pa; //处理链剩余部分
121     else p = pb;
122     while (p != NULL) {
123         pc = pc->InsertAfter(p->coef, p->exp);
124         p = p->link;
125     }
126     return C;
127 };
128
129 Polynomial operator - (Polynomial&A, Polynomial&B) { //减法的重载
130     Term*pa, *pb, *pd, *p;
131     float temp;
132     Polynomial D; pd = D.first;
133     pa = A.getHead()->link; pb = B.getHead()->link;
134     while (pa != NULL && pb != NULL) {
135         if (pa->exp == pb->exp) {
136             temp = pa->coef - pb->coef;
137             if (fabs(temp) > 0.001)
138                 pd = pd->InsertAfter(temp, pa->exp);
139             pa = pa->link; pb = pb->link;
140         }
141         else if (pa->exp < pb->exp) {
142             pd = pd->InsertAfter(pa->coef, pa->exp);
143             pa = pa->link;
144         }

```

```
145         else {
146             pd = pd->InsertAfter(pb->coef, pb->exp);
147             pb = pb->link;
148         }
149     }
150     if (pa != NULL) p = pa;
151     else p = pb;
152     while (p != NULL) {
153         pd = pd->InsertAfter(-(p->coef), p->exp);
154         p = p->link;
155     }
156     return D;
157 };
158
159 int main() {
160     int a;
161     Polynomial A, B, C, D;
162     cin >> A;
163     cout << A;
164     cin >> B;
165     cout << B;
166     cout << "选择进行的运算方式:" << endl;
167     cout << '\t' << "1.加法" << endl;
168     cout << '\t' << "2.减法" << endl;
169     cout << "Input:";
170     cin >> a;
171     if (a == 1) {
172         C = A + B;
173         cout << C << endl;
174     }
175     else {
176         D = A - B;
177         cout << D << endl;
178     }
179     return 0;
180 }
```