湖南大學

数据结构

课程实验报告

题　　目：线性表的应用之多项式加法

学生姓名：魏子铖

学生学号：201726010308

专业班级：软件 1703

完成时间：2018.11.2

一、需求分析

0) 问题分析

一个多项式可以表达为 x 的各次幂与系数乘积的和，比如：

现在，你的程序要读入两个多项式，然后输出这两个多项式的和，也就是把对应的幂上的系数相加然后输出。

需要实现的功能如下：

i. 将输入数据按照幂次从高到低存储
ii. 实现幂次相等的系数相加
iii. 格式化输出求和后的多项式

1) 输入数据

【输入格式】

总共要输入两个多项式，每个多项式的输入格式如下：

- 每行输入两个数字，第一个表示幂次，第二个表示该幂次的系数
- 以 0 0 结束一个多项式的输入
- 注意第一行和最后一行之间不一定按照幂次降低顺序排列
- 如果某个幂次的系数为 0，就不出现在输入数据中了
- 0 次幂的系数为 0 时还是会出现在输入数据中

【输入样例】

6 2
5 3
3 12
1 6
0 20
0 0
6 2
5 3
2 12
1 6
0 20
0 0

2) 输出数据

【输出格式】

从最高幂开始依次降到 0 幂，如：

2x^6+3x^5+12x^3-6x+20

注意其中的 x 是小写字母 x，而且所有的符号之间都没有空格，如果某个幂的系数为 0 则不需要有那项。

【输出样例】：

4x^6+6x^5+12x^3+12x^2+12x+40

3) 测试样例设计

	样例一	样例二	样例三	样例四	样例五
	7 3 4 1 2 1 1 2 0 5 0 0 7 2 4 3 2 6 1 6 0 1 0 0	9 2 1 4 0 3 5 4 6 6 0 0 9 7 1 2 6 2 5 2 0 13 0 0	8 2 6 -5 5 2 2 10 0 9 0 0 8 -3 6 4 5 1 2 -10 0 -20 0 0	6 3 5 -7 2 2 1 6 0 11 0 0 7 3 3 -6 2 1 1 5 0 4 0 0	9 5 6 13 5 -5 3 2 1 3 0 6 0 0 6 -11 3 3 1 -8 0 0
设计理由	一般情况	多项式次数乱序输入	多项式部分系数为负值	两个多项式次数不相同	两个多项式长度不同
样例输出	5x^7+4x^4 +7x^2+8x +6	9x^9+8x^6 +6x^5+6x +16	-x^8-x^6 +3x^5-11	3x^7+3x^6 -7x^5-6x^3 +3x^2+11x +15	5x^9+2x^6 -5x^5+5x^3 -5x+6

二、概要设计

1. 抽象数据类型

为实现上述程序的功能，不排除系数为小数的情况，可以用双精度浮点数存储用户的输入，并将用户输入的值存储于链表中。

抽象数据类型设计：

- 数据对象：一组两两相关的实数
- 数据关系：每一行的两个数为一个整体，一起表示多项式的某一项，满足线性特征
- 基本操作：将当前元素插入到链表并排序；清空链表中的元素；格式化输出链表中的内容
- **ADT：**

  Polynomial_List {

  数据对象：**D = {<key, value>$_i$ | key∈Q, value∈N, i = 1, 2, 3, ……, n, 1≤n≤1000}**

  数据关系：**R = {<key, value> | key∈Q, value∈N }**

  基本操作：

  ```
  void insert(k, v);
  //将元素插入到链表中，时间复杂度 O（1）
  void clear();
  //将链表内的元素清空，时间复杂度 O（n），其中 n 为 list 内元素的个数
  void print();
  //按照幂次从高到低的顺序格式化输出多项式，时间复杂度 O（n），其中 n 为 list 内元素的个数
  }
  ```

2. 算法的基本思想

对于各种功能：

1) 输入并建立多项式：

为输入输出格式的统一，本人规定因式格式为 ax^b 型，规定指数呈递增形式输出。

2) 输出多项式：

输出各系数的符号时，要判断系数的正负。

3) 多项式 A 和 B 相加：

从高次幂开始相加，若指数相同则系数相加减，若不相同则指数较大的一方即为运算结果。

3. 程序的流程

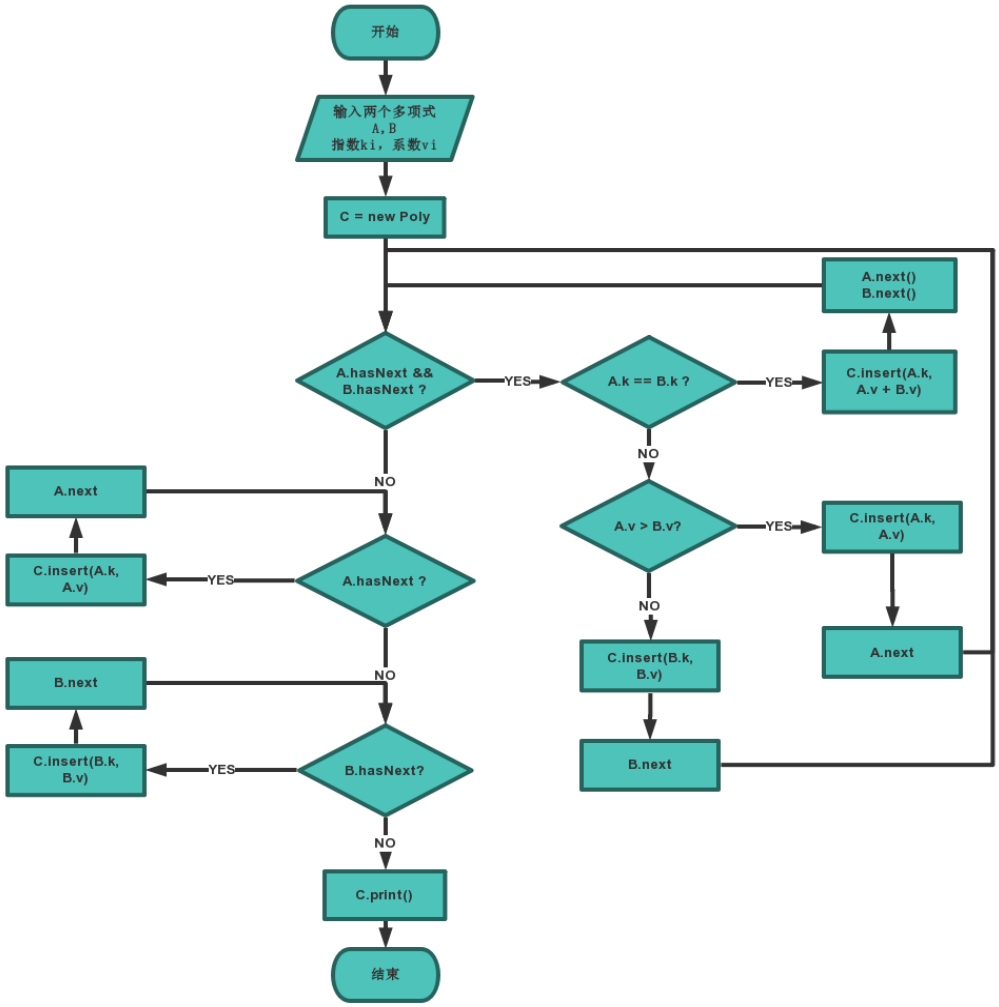
程序由三个模块组成：

1) 输入模块：提示输入格式以及结束标志，总共要输入两个多项式

2) 处理模块：新生成一个链表，按指数从大到小的顺序储存多项式，即为两个多项式相加后的结果

3) 输出模块：格式化顺序输出链表

程序流程图如下：



三、详细设计

1. 物理数据类型

输入的数据为两个一元多项式，系数用 double 类型，指数用 int 类型，由于多项式

是由 n 个独立的因式构成的，满足线性特性，所以物理数据类型为线性结构，对于插入操作来说，链表的时间复杂度明显低于顺序表，所以逻辑实现上采用链表的形式。

2. 输入和输出的格式

输入时有提示语句，说明输入的方法与结束条件，将每个多项式存储在不同的链表 A，B 中，多项式相加的结果存储在链表 C 中，最后格式化输出 C 中存储的内容。

3. 算法的具体步骤

1) 输入并建立多项式：

创建两个单链表，表示两个一元多项式，单链表的一个结点表示多项式的一个因式，结点有一元参数的系数域、指数域和连接多个因式的指针域。为方便对多项式进行删除或修改，单链表应该带空头结点。

有序链表的实现：

```
void Poly::insert(const double & coef, const int & index) {
    Node* tmp = head;
    if (tmp -> next == tail)
    //如果链表为空，则直接连接在尾部
        head -> next = new Node(coef, index, tail);
    else {
        while (tmp -> next -> index > index) {
        // 从头开始遍历，找到合适的插入位置
            tmp = tmp -> next;
        }
        tmp -> next = new Node(coef, index, tmp -> next);
        // 将结点插入链表
    }
}
```

2) 多项式 A 和 B 相加：

申请两个节点指针分别指向 A 和 B 的首个因式，比较它们的指数：

- 若相同则系数相加减，申请新的节点来储存系数和指数，两个指针同时右移；
- 若不相同，则将指数小的因式赋值给新申请的节点，插在新的单链表上，将指向指数较小因式的指针右移。

```
if (A.hasNext() && B.hasNext()) {
  if (indexA == indexB) {
    C.insert(A + B); A.next(); B.next();
  }
  else if (indexA > indexB) {
    C.insert(A); A.next();
  }
  else {
    C.insert(B); B.next();
  }
}
```

- 重复上述比较，当有指针为空时：
  - 若有一个指针为空，则将另一个指针指向的剩余的多项式因式连接到新的多项式后面；

■ 若两个指针同时为空，return。

```
else if (A.hasNext()) {
   C.insert(A); A.next();
}
else {
   C.insert(B); B.next();
}
```

由此看出，规定按指数从大到小的顺序储存多项式，决定了实现多项式相加减的这种算法的出现。

3) **删除 A 和 B：**

申请一个节点类型的指针，通过其移动，释放掉链表的全部节点，返回空的头结点。

```
while (head != nullptr) {
            curr = head;
            head = head -> next;
            delete curr;
}
```

4) **输出计算结果：**

输出各系数的符号时，要判断系数是正数则在其前输出加号，是负数则只需输出它本身（自带负号）。

4. 算法的时空分析

1) **输入并建立多项式：**时间复杂度 O（n^2），空间复杂度 O（n）
2) **多项式相加：**时间复杂度 O（n），空间复杂度 O（n）
3) **删除多项式：**时间复杂度 O（n），空间复杂度 O（1）
4) **输出多项式：**时间复杂度 O（n），空间复杂度 O（1）

四、调试分析

1.调试方案设计

调试目的：发现思维逻辑与代码实现上的区别，改进代码结构，排除语法逻辑上的错误

样例：

```
6 2
5 3
3 12
1 6
0 20
0 0
6 2
5 3
2 12
1 6
0 20
0 0
```

调试计划：设置好断点，注意观察每一步时各个变量的变化情况，找出错误的地方，然后改正；单步调试，更能准确定位出现错误的代码区域

2.调试过程和结果，及分析

调试过程中由于指针指向了未知的区域，导致代码多次崩溃，发现是 insert()方法出现

了问题，排除错误后调试成功，输出了正确的结果：4x^6+6x^5+12x^3+12x^2+12x+40

五、测试结果

1. 样例一

```
每行输入两个数字，第一个表示幂次，第二个表示该幂次的系数，以空格分离
第一行一定是最高幂，以0 0结束一个多项式的输入
===输入多项式A(x)===
7 3
4 1
2 1
1 2
0 5
0 0
===输入多项式B(x)===
7 2
4 3
2 6
1 6
0 1
0 0
A(x) + B(x) = 5x^7+4x^4+7x^2+8x+6
请按任意键继续. . .
```

一般情况，输出结果正确

2. 样例二

```
每行输入两个数字，第一个表示幂次，第二个表示该幂次的系数，以空格分离
第一行一定是最高幂，以0 0结束一个多项式的输入
===输入多项式A(x)===
9 2
1 4
0 3
5 4
6 6
0 0
===输入多项式B(x)===
9 7
1 2
6 2
5 2
0 13
0 0
A(x) + B(x) = 9x^9+8x^6+6x^5+6x+16
请按任意键继续. . .
```

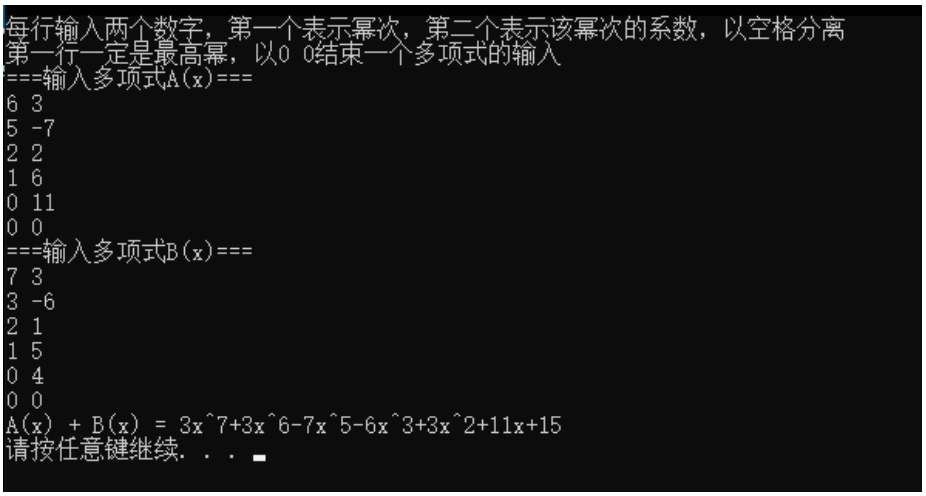
多项式次数乱序输入，输出结果正确

3. 样例三

```
每行输入两个数字，第一个表示幂次，第二个表示该幂次的系数，以空格分离
第一行一定是最高幂，以0 0结束一个多项式的输入
===输入多项式A(x)===
8 2
6 -5
5 2
2 10
0 9
0 0
===输入多项式B(x)===
8 -3
6 4
5 1
2 -10
0 -20
0 0
A(x) + B(x) = -x^8-x^6+3x^5-11
请按任意键继续. . .
```

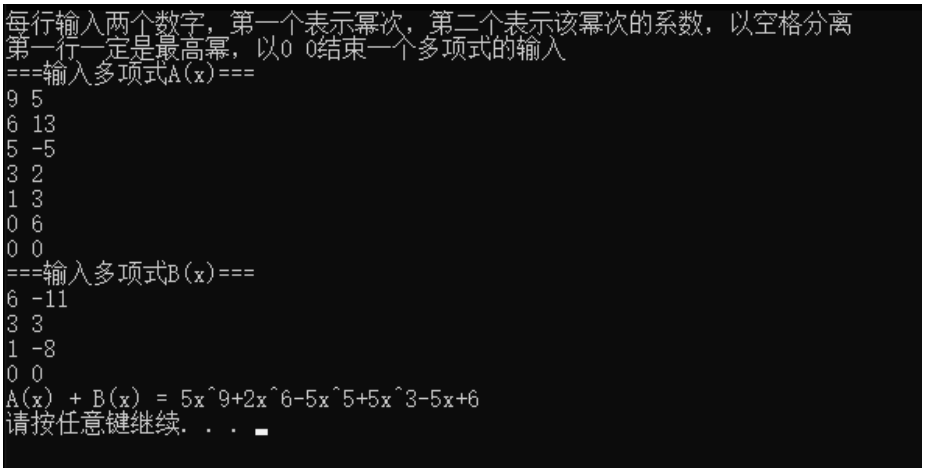
多项式部分系数为负值，输出结果正确

4. **样例四**



两个多项式次数不相同，输出结果正确

5. **样例五**



两个多项式长度不同，输出结果正确

六、实验日志

本次实验以单链表为基础，通过对其各种 ADT 的实现，完成了多项式的相加这一功能，由于链表的插入和删除操作的时间复杂度是常数级别的，所以虽然在实现难度上比顺序表要困难一些，但是在运行时间消耗上的差距在多项式项数很大时的差距时非常明显的。

在格式化输出的时候，对于系数为-1，0，1 或指数为 0，1 的情况做了特殊处理，使输出结果更加人性化，不显得刻意呆板，涉及到了字符串的细心处理。

实现各个功能时有具体的框架，主次分明，一块一块去搭建，通过实现多项式的加法，对链表有了更深入的了解。