

实验报告

课程名称：数据结构与算法

学 院： 计算机与信息安全学院

专业班级： 计算机类

学 号： 1800301632

姓 名： 史帅

报告日期：2019年 10 月 2 日

目录

实验目的**1**

**第一题：一元多项式相加1**

题目描述1

存储一元多项式1

多项式相加运算规则2

输入输出一元多项式2

实验目标2

函数定义，算法设计与分析1、3

复杂性分析3

流程图4

测试程序9

**第二题：使用单链表完成基本集合运算10**

题目描述10

存储集合10

实验目标10

函数定义，算法设计与分析11

复杂性分析12

流程图13

测试程序17

**实验小结17**

**线性表应用**

1. **实验目的**

掌握顺序线性表额链式线性表基本操作，理解线性表不同存储结构的使用场合，并能根据实际应用背景，选用或设计适当的结构并设计算法解决问题。

#### 第一题：一元多项式相加

题目描述：设计一种单链表存储结构，每个结点存储一项的系数和指数，类型都是整型，编写完成产生多项式的函数、多项式相加及输出多项式的函数。

##### 存储一元多项式

在数学上，一元多项式的形式：

p\_n(x)=p\_0+p\_1x^1+p\_2x^2+…+p\_nx^n*pn*​(*x*)=*p*0​+*p*1​*x*1+*p*2​*x*2+…+*pn*​*xn*

可由线性表（p\_0,p\_1,…p\_n*p*0​,*p*1​,…*pn*​）表示。一般情况下，一元多项式只表示非0系数项，采用链式存储，对应链表结点数据结构可采取：（设多项式的系数和指数都是整型）

若采用顺序存储，例如存储X + X^时浪费大量空间，为节省空间与方便处理，采用链式存储。

1. struct node
2. {
3. int exp; //表示指数
4. int coef; //表示系数
5. struct node \*next; //指向下一个结点的指针
6. }；

##### 多项式相加运算规则

两个一元多项式中所有指数相同的项，对于系数相加，若和不为0，则构成结果多项式中的一项，对于两个多项式中所有指数不同的项，则分别复制到结果多项式中。

举例：

p(x)=5+2x+3x^5-2x^7*p*(*x*)=5+2*x*+3*x*5−2*x*7

Q(x)=12x+2x^7+13x^{15}*Q*(*x*)=12*x*+2*x*7+13*x*15

则结果多项式为：

R(x)=5+14x+3x^5+13x^{15}*R*(*x*)=5+14*x*+3*x*5+13*x*15

##### 输入输出一元多项式

输入时逐项、按顺序输入一元多项式的系数、指数，输入系数为0时表述输入结束。

例如：

p(x)=5+2x+3x^5-2x^7*p*(*x*)=5+2*x*+3*x*5−2*x*7

输入： 5 0 2 1 3 5 -2 7 0 0

输出时采取以下格式，若预期输出的多项式为：

p(x)=5+2x+3x^5-2x^7*p*(*x*)=5+2*x*+3*x*5−2*x*7

则输出：5x^0 + 2x^1 + 3x^5 -2x^7

##### 实验目标

将输入的两个多项式进行合并并输出。

函数定义，算法设计与分析

实验需要用到多项式的存储，两个多项式的加和，输出多项式这三个操作

为实现这些操作，需要对功能函数进行如下的定义。

1. PNODE createPoly(void)

模块功能：将用链表存储一个多项式。

1. PNODE addPoly(PNODE polyAddLeft , PNODE polyAddRight)

模块功能：对两个多项式进行合并，并返回一个结果多项式。

1. void printPoly(PNODE poly)

模块功能：将结果多项式显示在屏幕上。

复杂性分析

在存储每一个多项式的项的时候，每个项只需要申请一个新的链表空间，再将指数与系数分别插入即可，所以 createPoly的复杂度为O（n）。

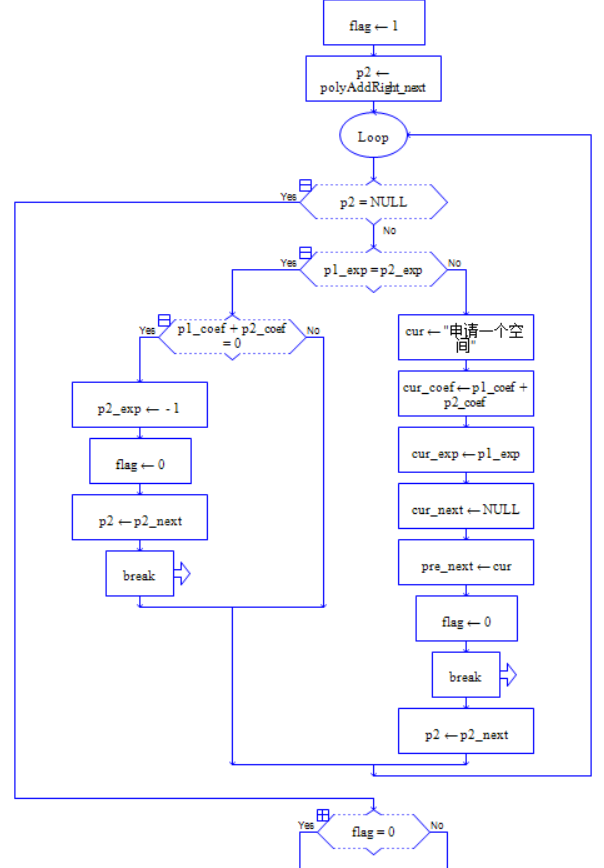
对两个多项式进行合并的时候，由于要遍历两个链表中每一个数据，所以addPoly的复杂度为O（m\*n）输出结果多项式，只需要遍历结果链表中的每一个值，再将其输出即可，所以printPoly的复杂度为O（n）。

|  |  |
| --- | --- |
| 函数名 | 复杂度 |
| createPoly | O（n） |
| addPoly | O（m\*n） |
| printPoly | O（n） |

流程图

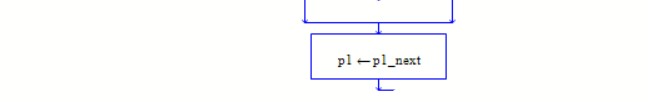


main creatPoly

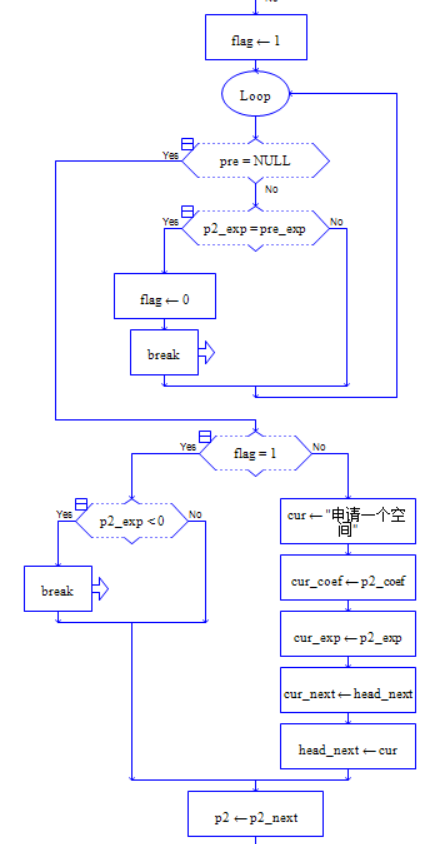
 addPoly 

一号内函数

二号内函数



一号内函数

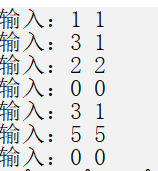


二号内函数

printPoly

测试程序

第一次：（系数均为正数）输入两个多项式分别为 1 + 3X + 2X＾2 和 3X + 5X＾5（程序输入时，当系数为0时停止输入）



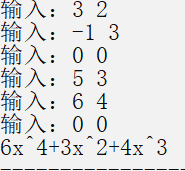
经过程序进行合并，应得到多项式 1 + 6X + 2X＾2 + 5X＾5

图片包含 屏幕截图

描述已自动生成

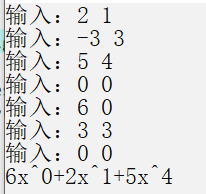
第二次：（系数为正数和负数）输入两个多项式分别为 3X＾2 - X＾3 和 5X＾3 + 6X＾3（程序输入时，当系数为0时停止输入）

经过程序进行合并，应得到多项式 3X＾2 + 4X＾3 + 6X＾4



第三次：（两个多项式中有相互抵消的项）输入两个多项式分别为 2X -3X＾3 + 5X＾4 和 6 + 3X＾3（程序输入时，当系数为0时停止输入）

经过程序进行合并，应得到多项式 6+2X^1+5X^4



#### 第二题：利用单链表实现集合基本运算

题目描述：完成单链表基本操作，并使用基本操作实现集合的并、交和差功能。

##### 存储集合

集合的实现方法有多种，其中一种方法是用单链表表示。存储结构可定义如下：

1. struct Node
2. {
3. DataType element;
4. struct Node \*next;
5. }；

集合也可以用顺序存储，但对其进行并，、交和差处理时非常不方便，没有链表灵活。

假设单链表的头结点不存放集合元素，但头结点的element存放链表中数据元素的个数。

实验目标

测试输入：

5 91 51 2 32 7

4 21 51 2 16

预期输出：

并集：21 16 91 51 2 32 7

交集：51 2

差集：91 32 7

函数定义，算法设计与分析

此程序目的是为了实现两个集合的并、交和差，需要实现的操作为新建一个新的集，三个函数分别实现集合的并、交和差的操作，再有一个函数可以将生成的结果集合显示在屏幕上。

为实现这些操作，需要对功能函数进行如下的定义。

1. SET InitSet(int num)

模块功能：将一个集合保存在一个单链表中，返回这个新的集合

1. int find(DataType datax, SET set)

模块功能：判断集合中是否有dataX这个值，存在返回1，否则返回0

1. void insert(DataType datax, SET set)

模块功能：在集合中插入值为datax的成员 ，插入位置在表头

1. void copySet(SET setA, SET setB)

模块功能：将集合setA复制生成集合setB

1. void printSet(SET set)

模块功能：将集合中的值显示在屏幕上

1. SET setUnion(SET setA ,SET setB)

模块功能：求两个集合setA 和 setB的并集，并返回新的并集集合

1. SET setIntersect (SET setA ,SET setB)

模块功能：求两个集合setA 和 setB的交集，并返回新的交集集合

1. SET setExcept (SET setA ,SET setB)

模块功能：求两个集合setA 和 setB的差集，并返回新的差集集合

复杂性分析

InitSet函数存储一个集合，每次申请空间，插入就可以复杂度为O（n），

Find函数只需将整个集合遍历一遍即可，复杂度为O（n），insert函数为initSet函数的内部调用函数，为O（1）复杂度，copySet函数复制一个集合，每次申请一个空间，复制一个集合中的成员，复杂度也为O（n），printSet函数将集合中的元素一个一个读取并输出，复杂度为O（n），setUnion函数调用find函数与initSet函数，进行两次遍历，复杂度为O（n），setIntersect函数内部有两个嵌套循环，复杂度为O（m\*n），setExcept函数内部进行一次遍历，复杂度为O（n）。

|  |  |
| --- | --- |
| 函数名 | 复杂度 |
| InitSet | O（n） |
| Find | O（n） |
| insert | O（1） |
| copySet | O（n） |
| printSet | O（n） |
| setUnion | O（n） |
| setIntersect | O（m\*n） |
| setExcept | O（n） |

流程图



Main InitSet



printSet setUnion(setA,setB)

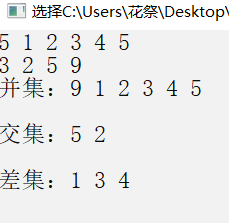


setExcept(setA,setB)

测试程序

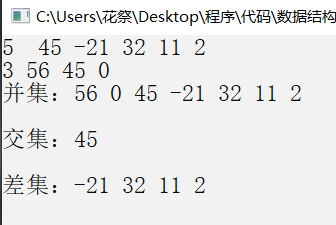
第一次：输入两个集合 1 2 3 4 5和2 5 9

得到结果



第二次：输入两个集合 45 -21 32 11 2和56 45 0

得到结果



实验小结

经过本次实验，实验者对线性表的基本操作有了较好的掌握，但是为了完成本次实验还是废了不少时间的，C++ 中对指针的使用，使得其处理数据速度非常快，这是它的优点，但是想要灵活使用指针还需要大量的练习，不然经常出莫名其妙的错误，比如没有申请空间，只有一个类型指针就去调用别的函数进行处理，这样根本返回不了结果，别的函数没问题，编译器也不报错，不小心的话这种错误真的是太令人头疼了！！！

而且尤其是表头和表尾处理的时候，表头要有头节点，不能改变头指针，在实验中就不小心给指向头指针的指针赋了别的值，导致整个链表的迷失，表尾处理也要注意指针有没有越界！！！