

**课 程 实 验 报 告**

**课程名称： 数据结构实验**

实验名称： 基于链表的树、图实现

**专业班级：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**学 号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**姓 名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**报告日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**网络空间安全学院**

**目 录**

**[1 链式循环队列的设计 1](#_Toc118025690)**

[1.1 需求分析 1](#_Toc118025691)

[1.2 总体设计 1](#_Toc118025692)

[1.3 数据结构 1](#_Toc118025693)

[1.4 算法设计 2](#_Toc118025694)

[1.5 系统实现 2](#_Toc118025695)

[1.6 系统测试 3](#_Toc118025696)

[1.7 复杂度分析 3](#_Toc118025697)

[1.8 结果分析 3](#_Toc118025698)

[1.9 实验小结 4](#_Toc118025699)

**[2 链式栈建栈与操作 5](#_Toc118025700)**

[2.1 需求分析 5](#_Toc118025701)

[2.2 总体设计 5](#_Toc118025703)

[2.3 数据结构 5](#_Toc118025704)

[2.4 算法设计 6](#_Toc118025705)

[2.5 系统实现 8](#_Toc118025706)

[2.6 系统测试 8](#_Toc118025707)

[2.7 复杂度分析 8](#_Toc118025708)

[2.8 结果分析 9](#_Toc118025709)

[2.9实验小结 9](#_Toc118025710)

**[3 一元稀疏多项式加减运算 10](#_Toc118025711)**

[3.1 需求分析 10](#_Toc118025712)

[3.2 总体设计 10](#_Toc118025713)

[3.3 数据结构 10](#_Toc118025714)

[3.4 算法设计 11](#_Toc118025715)

[3.5 系统实现 14](#_Toc118025716)

[3.6 系统测试 14](#_Toc118025717)

[3.7 复杂度分析 15](#_Toc118025718)

[3.8 结果分析 15](#_Toc118025719)

[3.9 实验小结 15](#_Toc118025720)

**[4 一元稀疏多项式运算的升级功能 16](#_Toc118025721)**

[4.1 需求分析 16](#_Toc118025722)

[4.2 总体设计 16](#_Toc118025723)

[4.3 数据结构 17](#_Toc118025724)

[4.4 算法设计 17](#_Toc118025725)

[4.5 系统实现 18](#_Toc118025726)

[4.6 系统测试 19](#_Toc118025727)

[4.7 复杂度分析 19](#_Toc118025728)

[4.8 结果分析 20](#_Toc118025729)

[4.9 实验小结 20](#_Toc118025730)

**[参考文献 21](#_Toc118025731)**

**[附录 基于链式存储结构线性表实现的源程序 22](#_Toc118025733)**

# 1 链式循环队列的设计

## 1.1 需求分析

### 1.1.1 功能需求

以链表作为物理结构，使用C语言编程实现循环队列的基本功能，可参考约瑟夫环的设计。依据循环队列的功能，实现对输入/输出数据进行入队/出队操作，能够处理满队/空队等边界情况。

### 1.1.2 输入输出需求

使用链表建立容量为8的循环空队列，下标依次记为0,1,2...,7,指定起始位置（队首）下标i，读入数据后从起始位置i顺序存储整数序列，队列满队(读入8个元素)之后停止读入；输出时，如果输出个数大于存储最大容量8则输出"Error"，否则如果输出个数大于已存储元素个数(即会发生空队列输出)则输出"Fault"；如果队列已满则先输出"Full"，然后从起始位置开始顺序输出k个数据，否则直接顺序输出k个数据。

输入形式： 非负整数序列n,i,k,e1,e2,...,en。（n为读入元素个数,i为队首下标,k为输出个数）

输出形式：非负整数序列"Full",e1',e2',...,ek'或"Error"或"Fault"。（"Full"只有在队满时才会输出，后面是循环队列中存储的k元素）

## 1.2 总体设计

整个程序分为建立循环队列并初始化和入队/出队处理两个部分。

## 1.3 数据结构

struct node {

    int index;

    int data;

    struct node \*pre;

    struct node \*next;

} Node;

使用结点内的指针连接结点，形成双向循环链表，构成了链式循环队列。

## 1.4 算法设计

链式队列入队/出队系统如图1-1所示。

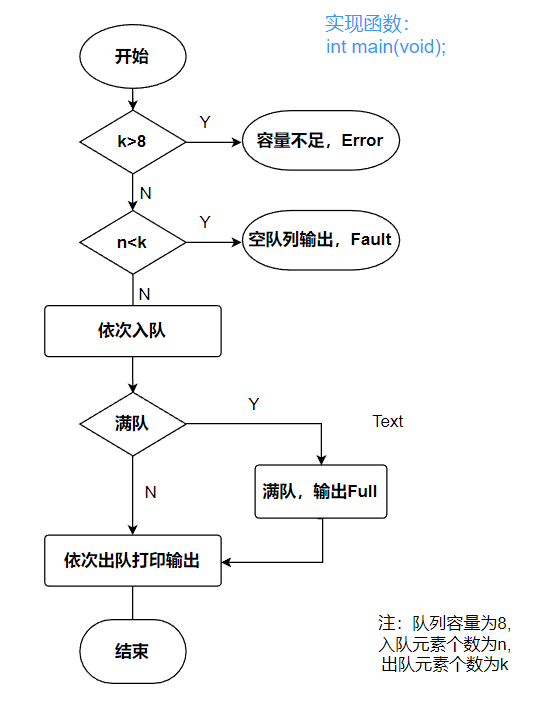


图1-1 链式队列入队/出队系统

## 1.5 系统实现

本程序全程在Microsoft Visual Studio 2022上编写、编译、调试、运行，并最终在Educoder平台上运行通过。

主要函数以及功能如表1-1所示。

表1-1主要函数及功能

|  |  |
| --- | --- |
| 函数名 | 主要功能 |
| int main() | 实现队列入队和出队功能 |
| void init() | 完成链式队列的创建与初始化 |

## 1.6 系统测试

支持Educoder平台的所有可见测试用例与隐藏测试用例，均通过，如图1-2所示。



图1-2 通过所有测试

## 1.7 复杂度分析

创建固定大小的链式队列只需要的时间；入队和出队时最好的情况是判断一次n和k后，输出“Error”或“Fault”，时间复杂度为，最坏的情况是需要入队出队，时间复杂度为（n为入队个数，k为出队个数，0<k<n<9）。故时间复杂度为，n，k分别为入队和出队的个数。

数据存储空间为固定8个大小的链式循环队列。故空间复杂度为。

## 1.8 结果分析

成功通过所有的给定测试用例，表明该链式循环队列操作程序设计成功实现使用需求。（若存在测试用例没有通过，试分析原因，下同）

## 1.9 实验小结

略。（在实验中遇到的问题，以及解决方案等，下同）

# 2 链式栈建栈与操作

## 2.1 需求分析

### 2.1.1 功能需求

以链表作为物理结构，使用C语言编程实现栈的基本功能。依据栈的功能，实现对输入/输出数据进行入栈/出栈操作，能够处理溢出等边界情况。

### 2.1.2 输入输出需求

建立链式栈(长度=5)来存储多项式序列并合并，之后逆序输出。

输入形式：

按幂次升序输入多项式序列

c1 e1 c2 e2...cn en ; c1' e1' c2' e2'...cn' en'

输出形式：

将两多项式合并，按幂次降序输出多项式序列。

注：

两多项式序列中没有相等的幂次（即不存在1<=i<=n，1<=j<=n'，使得ei=ej'）。其中ci,ci'为整数，代表系数；ei,ei'为整数，代表指数；';'为分隔符。

## 2.2 总体设计

整个程序分为字符串读取系统，链表合并系统2个部分。

字符串读取系统将读入的字符串数据转化为数值存入链表的结点中，同时实现对于中断标志‘;’的判断。

链表合并系统将两个多项式链表进行按顺序比较合并，实现幂次降序排列。

## 2.3 数据结构

struct node

{

    int coef;

    int exp;

    int layer;

    struct node\* next;

}

使用结点内的指针连接结点，形成链表，构建链式栈。

## 2.4 算法设计

字符串读取系统，链表合并系统分别如图2-1,2-2所示。

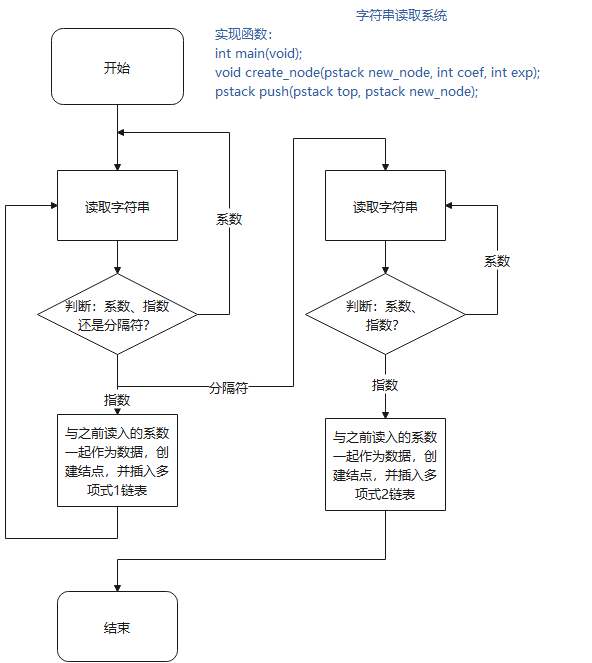


图2-1 字符串读取系统

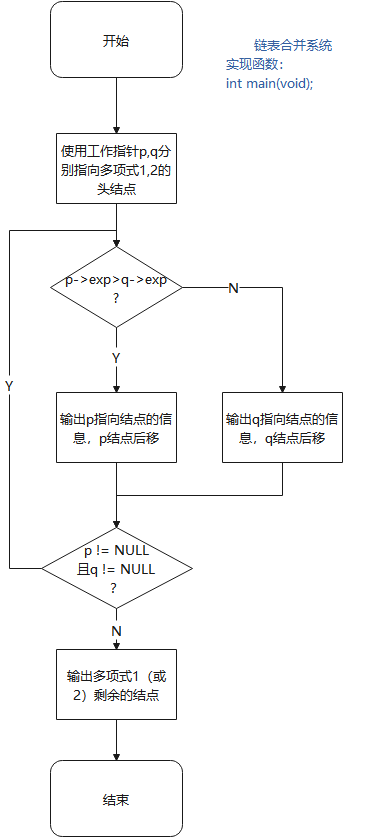


图2-2 链表合并系统

## 2.5 系统实现

本程序全程在Microsoft Visual Studio 2022上编写、编译、调试、运行，并最终在Educoder平台上运行通过。

主要函数以及功能如表2-1所示。

表2-1 主要函数及功能

|  |  |
| --- | --- |
| 函数名 | 主要功能 |
| int main() | 实现读取多项式、合并多项式、输出多项式功能 |
| void create\_node(pstack new\_node, int coef, int exp) | 为多项式的一项创建结点 |
| pstack push(pstack top, pstack new\_node) | 完成入栈操作 |

## 2.6 系统测试

支持Educoder平台的所有可见测试用例与隐藏测试用例，均通过，如图2-3所示。

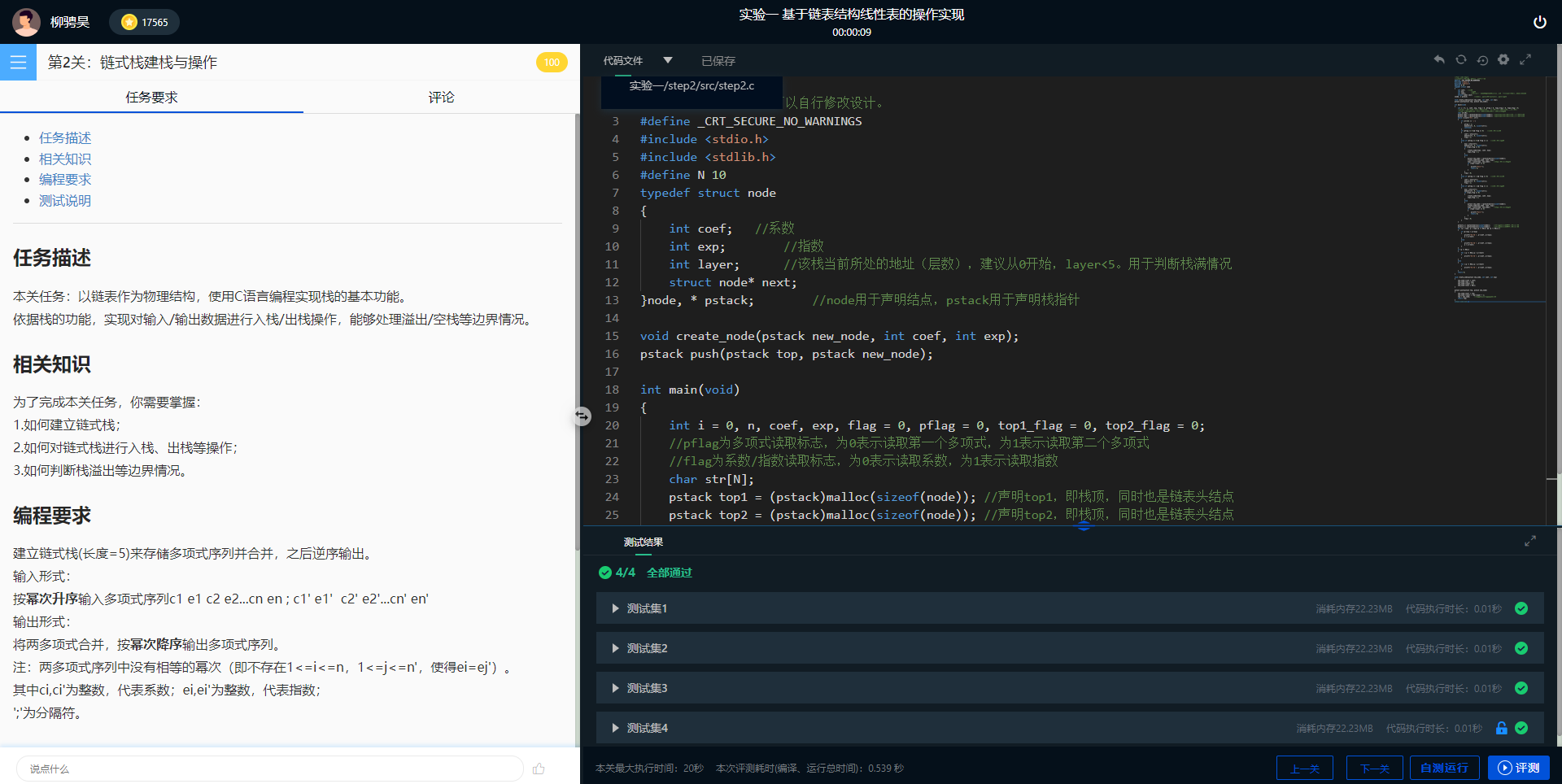


图2-3 测试通过

## 2.7 复杂度分析

读取多项式和合并链表时都只需将数据遍历一遍。故时间复杂度为，n，n'分别为两个多项式的项数。

数据存储空间为多项式项数的线性级别。故空间复杂度为，n，n'分别为两个多项式的项数。

## 2.8 结果分析

成功通过所有的给定测试用例，表明该链式栈操作程序设计成功实现使用需求。

## 2.9实验小结

略。

# 3 一元稀疏多项式加减运算

## 3.1 需求分析

### 3.1.1 功能需求

对两个一元稀疏多项式进行加减运算，运算结果按照幂次从高到低依次输出多项式的幂次和系数。

### 3.1.2 输入输出需求

对两个一元稀疏多项式进行加减运算。

输入形式：

c1 e1 c2 e2...... cn1 en1 ± c1' e1' c2' e2'........cn2' en2'；ci，ci'为浮点数，分别为两个多项式第i项的系数；ei，ei'为整数，分别为两个多项式第i项的指数，指数乱序排列；±为加号或减号。

输出形式：

将运算后的结果多项式按幂次降序排列，依次输出结果多项式的系数和指数。

## 3.2 总体设计

整个程序分为字符串读取系统，多项式链表排序系统，多项式链表加减系统3个部分。

字符串读取系统将读入的字符串数据转化为数值存入链表的结点中，同时实现对于加减符号‘±’的判断。

多项式链表排序系统分别将两个多项式按幂次降序排序，为之后的加减运算做准备。

多项式链表加减系统对两个多项式进行加减。

## 3.3 数据结构

struct node

{

    int coef;

    int exp;

    int layer;

    struct node\* next;

}

使用结点内的指针连接结点，形成链表。

## 3.4 算法设计

字符串读取系统，多项式链表排序系统，多项式链表加减系统分别如图3-1,3-2,3-3所示。

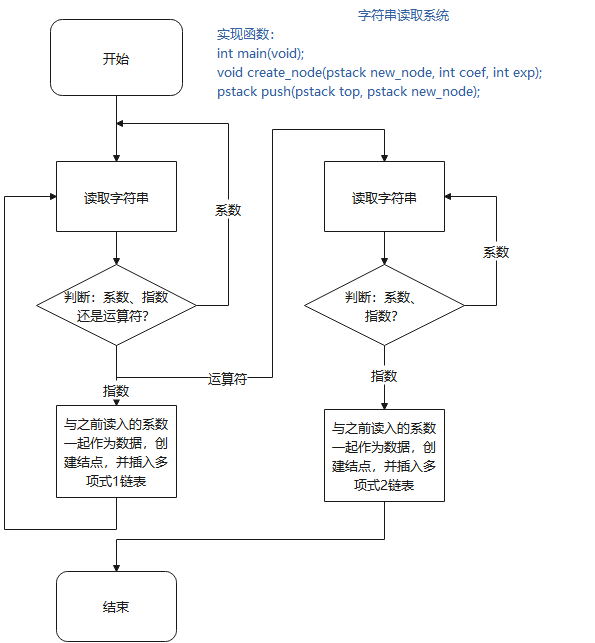


图3-1 字符串读取系统

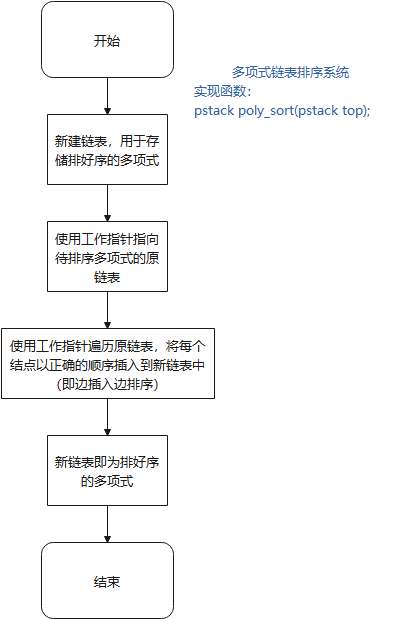


图3-2 多项式链表排序系统

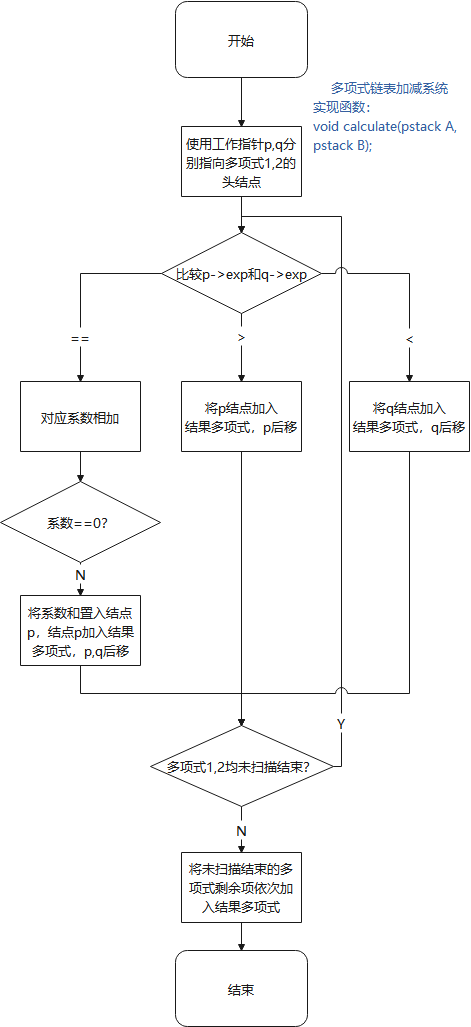


图3-3 多项式链表加减系统

## 3.5 系统实现

本程序全程在Microsoft Visual Studio 2022上编写、编译、调试、运行，并最终在Educoder平台上运行通过。

主要函数以及功能如表3-1所示。

表3-1 主要函数及功能

|  |  |
| --- | --- |
| 函数名 | 主要功能 |
| int main() | 实现读取多项式、输出多项式功能 |
| pstack poly\_sort(pstack top) | 实现对多项式进行排序的功能 |
| void calculate(pstack A, pstack B); | 实现对两个多项式进行加法运算的功能 |
| void display(pstack A) | 实现按ci，ei格式输出多项式各项 |
| void create\_node(pstack new\_node, int coef, int exp) | 实现创建结点的功能 |
| pstack push(pstack top, pstack new\_node) | 实现结点入栈的功能 |

## 3.6 系统测试

支持Educoder平台的所有可见测试用例与隐藏测试用例，均通过，如图3-4所示。

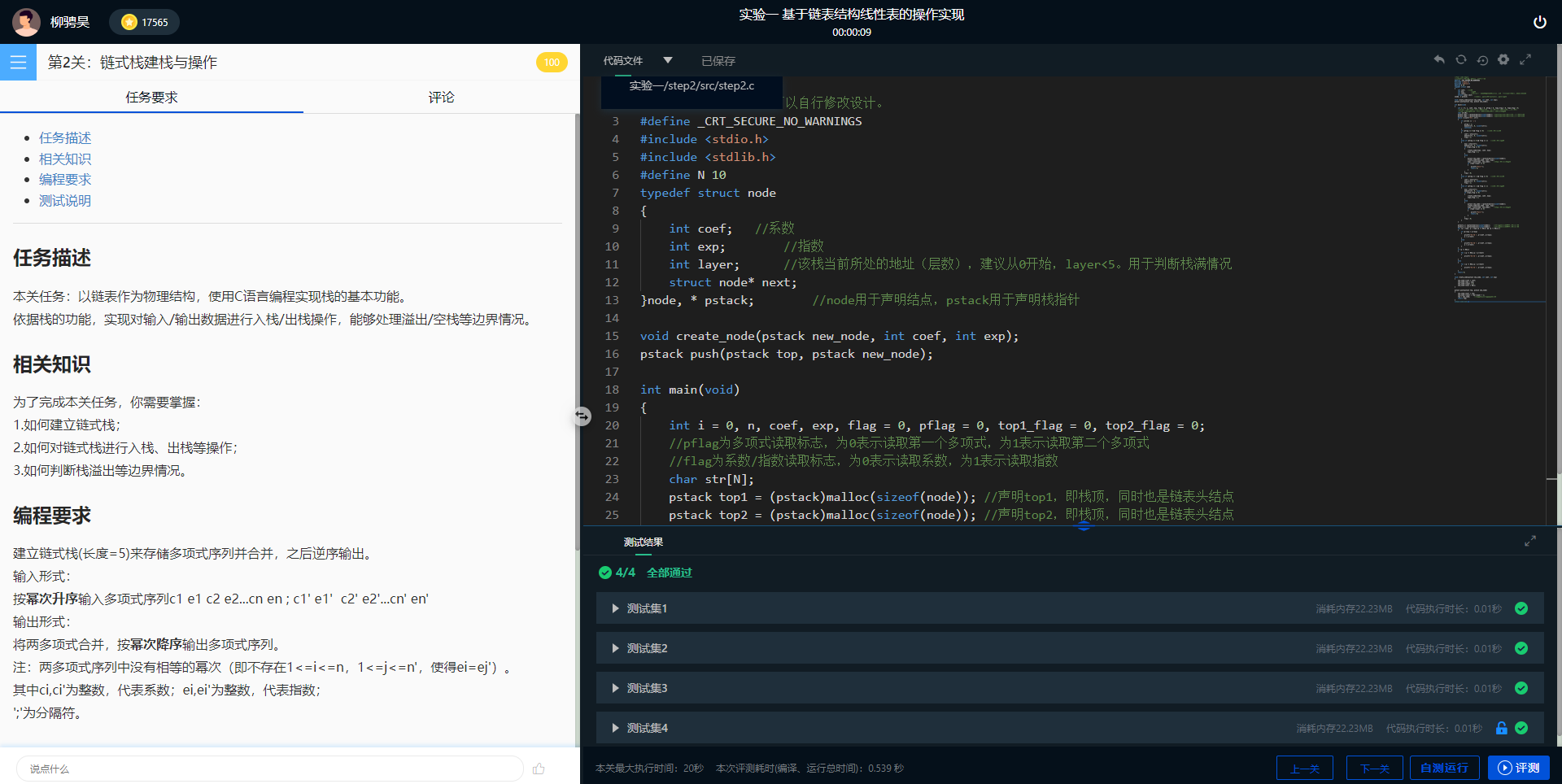


图3-4 测试通过

## 3.7 复杂度分析

读取多项式和加减多项式时都只需将数据遍历一遍，故时间复杂度为；将两个多项式排序时，时间复杂度取决于使用的排序算法，此处是插入排序，故时间复杂度为。故总的时间复杂度为。n，n'分别为两个多项式的项数。

数据存储空间为多项式项数的线性级别。故空间复杂度为，n，n'分别为两个多项式的项数。

## 3.8 结果分析

成功通过所有的给定测试用例，表明该一元稀疏多项式加减运算器成功实现使用需求。

## 3.9 实验小结

略。

# 4 一元稀疏多项式运算的升级功能

## 4.1 需求分析

### 4.1.1 功能需求

对一元稀疏多项式的加减运算提供升级功能，满足以下几点：

可以实现自然语言格式的输入输出；可以处理输入项的幂次有重复的情况；可以处理输入项的次序是按幂次乱序的情况。

### 4.1.2 输入输出需求

对两个一元稀疏多项式进行加减运算。

输入形式：(c1x^e1±c2x^e2±...±cnx^en1)±(c1'x^e1'±c2'x^e2'±...±cn'x^en2')

ci，ci'为浮点数，分别为两个多项式第i项的系数；

ei，ei'为整数，分别为两个多项式第i项的指数；

多项式按幂次乱序排列。

输出形式： c1x^e1±c2x^e2±...±cnx^en1 按幂次降序排列

注意，输出形式需满足常见的数学多项式的书写形式。 比如如下特殊情况：

当系数为±1且指数不为0时，1省略；

当指数为0时，x和指数省略；

当系数为0时，该项省略；

加一个负数，需要表示成减一个正数；

等等。

## 4.2 总体设计

整个程序分为字符串匹配转换系统、多项式链表排序系统，多项式链表加减系统、字符串输出系统4个部分。

字符串匹配转换系统将用户输入的自然表达式转换为链表，每个链表保存一项的信息，从表头至表尾呈降幂排列（如果输入不为降幂会进行自动调整，如果输入中存在两项次数相同会进行自动合并，如果输入为0则表为空）。

字符串输出系统将结果多项式的链表进行输出。

## 4.3 数据结构

struct node

{

    int coef;

    int exp;

    int layer;

    struct node\* next;

}

使用结点内的指针连接结点，形成链表。

## 4.4 算法设计

字符串匹配转换系统与字符串输出系统如图4-1,4-2所示，多项式链表排序系统和多项式链表加减系统已在第三关报告中给出，此处不再次给出。

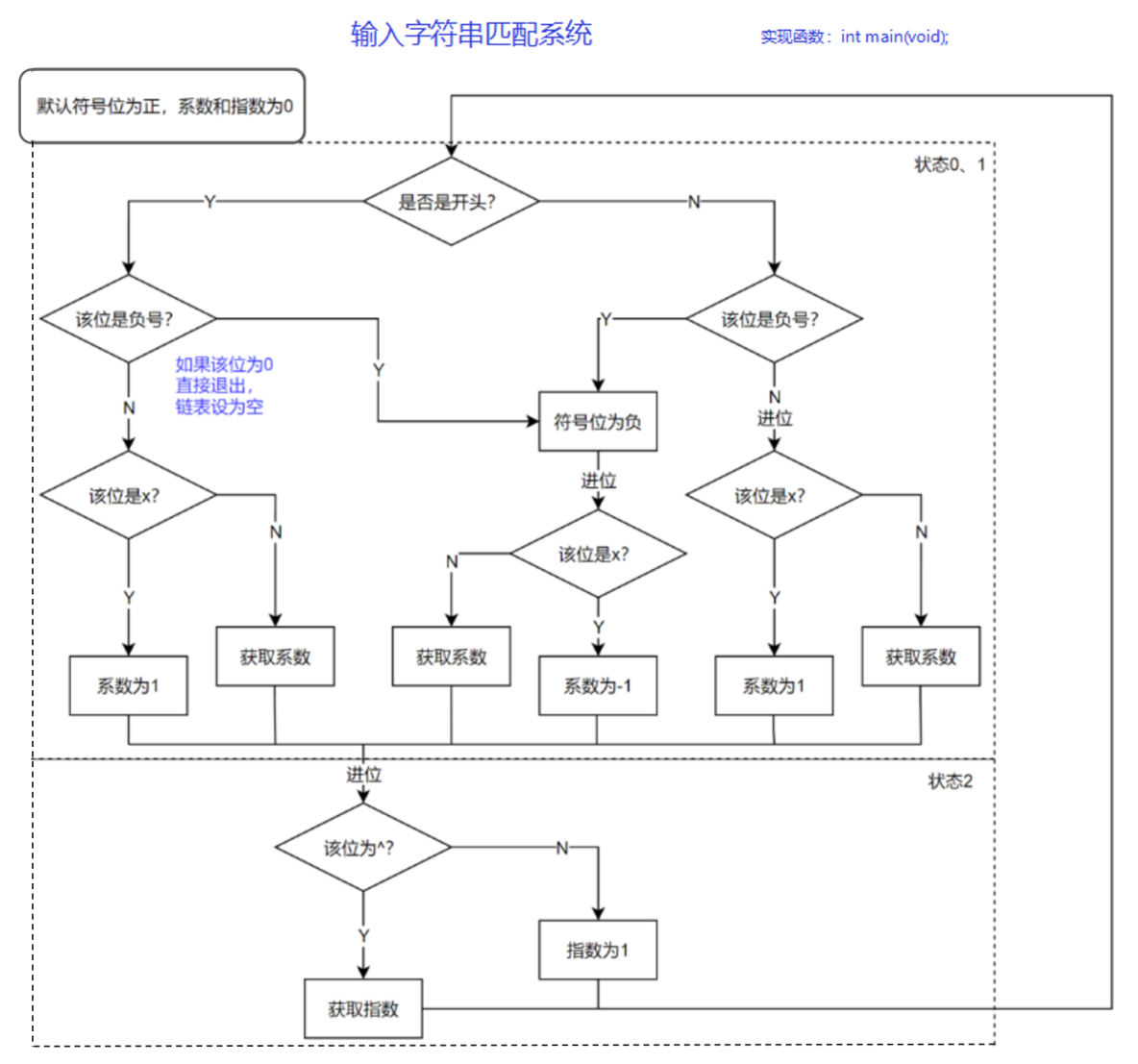


图4-1 输入字符串匹配系统

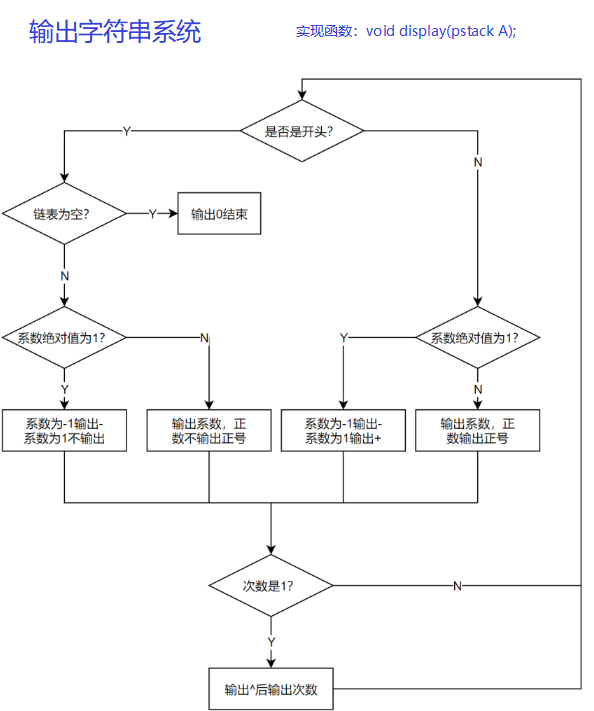


图4-2 输出字符串系统

## 4.5 系统实现

本程序全程在Microsoft Visual Studio 2022上编写、编译、调试、运行，并最终在Educoder平台上运行通过。

主要函数以及功能如表4-1所示。

表4-1 主要函数及功能

|  |  |
| --- | --- |
| 函数名 | 主要功能 |
| int main() | 实现按自然语言格式读取多项式的功能，并调用其他系统中的函数 |
| pstack poly\_sort(pstack top) | 实现对多项式进行排序的功能 |
| void calculate(pstack A, pstack B); | 实现对两个多项式进行加法运算的功能 |
| void display(pstack A) | 实现自然语言格式输出多项式各项 |
| void create\_node(pstack new\_node, int coef, int exp) | 实现创建结点的功能 |
| pstack push(pstack top, pstack new\_node) | 实现结点入栈的功能 |

## 4.6 系统测试

支持Educoder平台的所有可见测试用例与隐藏测试用例，均通过，如图4-3所示。

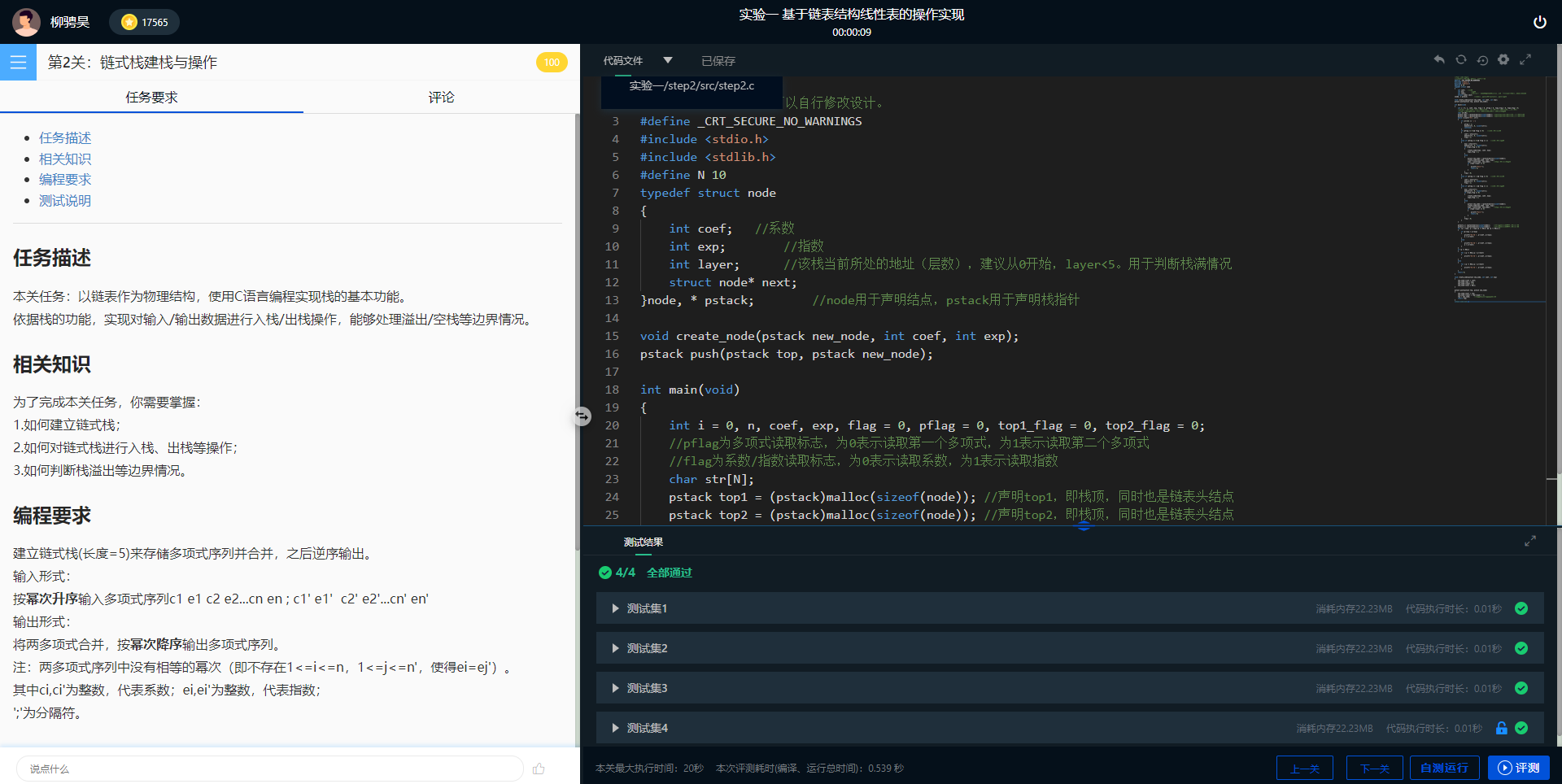


图4-3 测试通过

## 4.7 复杂度分析

读取多项式、处理自然语言和加减多项式时都只需将数据遍历一遍，故时间复杂度为；将两个多项式排序时，时间复杂度取决于使用的排序算法，此处是插入排序，故时间复杂度为。故总的时间复杂度为。n，n'分别为两个多项式的项数。

数据存储空间为多项式项数的线性级别。故空间复杂度为，n，n'分别为两个多项式的项数。

## 4.8 结果分析

成功通过所有的给定测试用例，表明该一元稀疏多项式加减运算器的升级功能需求成功实现。

## 4.9 实验小结

略。

# 参考文献

[1] 严蔚敏等. 数据结构(C语言版). 清华大学出版社

[2] [Larry Nyhoff](http://www.calvin.edu/~nyhl/index.html). [ADTs, Data Structures, and Problem Solving with C++.](http://vig.prenhall.com/catalog/academic/product/0,1144,0131409093,00.html)Second Edition, [Calvin College](http://cs.calvin.edu/), 2005

[3] 严蔚敏等.数据结构题集(C语言版). 清华大学出版社

# **附录 基于链式存储结构线性表实现的源程序**

略。