学生学号	0121710880223	实验课成绩	

武汉理工大学

学 生 实 验 报 告 书

实验课程名称	数据结构
开 课 学 院	计算机科学与技术学院
指导教师姓名	胡燕
学 生 姓 名	刘佳迎
学生专业班级	计算机类 m1702 班

2018 -- 2019 学年 第 一 学期

实验教学管理基本规范

实验是培养学生动手能力、分析解决问题能力的重要环节；实验报告是反映实验教学水平与质量的重要依据。为加强实验过程管理，改革实验成绩考核方法，改善实验教学效果，提高学生质量，特制定实验教学管理基本规范。

1、 本规范适用于理工科类专业实验课程，文、经、管、计算机类实验课程可根据具体情况参照执行或暂不执行。

2、 每门实验课程一般会包括许多实验项目，除非常简单的验证演示性实验项目可以不写实验报告外，其他实验项目均应按本格式完成实验报告。

3、 实验报告应由实验预习、实验过程、结果分析三大部分组成。每部分均在实验成绩中占一定比例。各部分成绩的观测点、考核目标、所占比例可参考附表执行。各专业也可以根据具体情况，调整考核内容和评分标准。

4、 学生必须在完成实验预习内容的前提下进行实验。教师要在实验过程中抽查学生预习情况，在学生离开实验室前，检查学生实验操作和记录情况，并在实验报告第二部分教师签字栏签名，以确保实验记录的真实性。

5、 教师应及时评阅学生的实验报告并给出各实验项目成绩，完整保存实验报告。在完成所有实验项目后，教师应按学生姓名将批改好的各实验项目实验报告装订成册，构成该实验课程总报告，按班级交课程承担单位（实验中心或实验室）保管存档。

6、 实验课程成绩按其类型采取百分制或优、良、中、及格和不及格五级评定。

附表：实验考核参考内容及标准

	观测点	考核目标	成绩组成
实验预习	1. 预习报告 2. 提问 3. 对于设计型实验，着重考查设计方案的科学性、可行性和创新性	对实验目的和基本原理的认识程度，对实验方案的设计能力	20%
实验过程	1. 是否按时参加实验 2. 对实验过程的熟悉程度 3. 对基本操作的规范程度 4. 对突发事件的应急处理能力 5. 实验原始记录的完整程度 6. 同学之间的团结协作精神	着重考查学生的实验态度、基本操作技能；严谨的治学态度、团结协作精神	30%
结果分析	1. 所分析结果是否用原始记录数据 2. 计算结果是否正确 3. 实验结果分析是否合理 4. 对于综合实验，各项内容之间是否有分析、比较与判断等	考查学生对实验数据处理和现象分析的能力；对专业知识的综合应用能力；事实求实的精神	50%

实验课程名称：数据结构

实验项目名称	实验一			实验成绩	
实 验 者	刘佳迎	专业班级	计算机类 m1702 班	组 别	
同 组 者				实验日期	2018 年 12 月 12 日

一部分：实验预习报告（包括实验目的、意义，实验基本原理与方法，主要仪器设备及耗材，实验方案与技术路线等）

实验目的

1、熟悉 VC 环境，学习使用 C 语言利用链表的存储结构解决实际的问题。
2、在编程、上机调试的过程中，加深对线性链表这种数据结构的基本概念理解。
3、锻炼较强的思维和动手能力和更加了解编程思想和编程技巧。

实验内容

1．约瑟夫环问题。
2．设计一个一元稀疏多项式简单计算器，要求基本功能：
（1）输入并建立多项式
（2）输出多项式
（3）两个多项式相加
（4）两个多项式相减
3．设计一个程序，演示运算符优先法对算术表达式求值的过程

实验基本原理及方法

问题一

1．从键盘输入整数 m，通过 create 函数生成一个具有 m 个结点的单向环表。环表中的结点编号依次为 1，2，……，m。
2．从键盘输入整数 s（1<=s<=m）和 n，从环表的第 s 个结点开始计数为 1，当计数到第 n 个结点时，输出该第 n 结点对应的编号，将该结点从环表中消除，从输出结点的下一个结点开始重新计数到 n，这样，不断进行计数，不断进行输出，直到输出了这个环表的全部结点为止。
例如，m=10，s=3，n=4。则输出序列为：6，10，4，9，5，2，1，3，8，7。

问题二

1．掌握线性结构的逻辑特性和物理特性。
2．建立一元多项式。
3．将一元多项式输入，并存储在内存中，并按照指数降序排列输出多项式。
4．能够完成两个多项式的加减运算，并输出结果。

问题三

1．建立运算数栈 SqStack1 和运算符栈 SqStack2 辅助分析算符有限关系.
2．用户输入以“+”结尾的算数表达式，本程序需要用户自行输入表以字符形式读

入，在读入的同时，完成运算符和运算数的识别处理，在识别出运算数的同时，要将其字符序列形式转换成整数形式。

3．在程序的适当位置输出运算符栈、运算数栈、输入字符和主要操作的内容，即演示运算操作。测试数据见原题。

4．程序执行的命令包括：

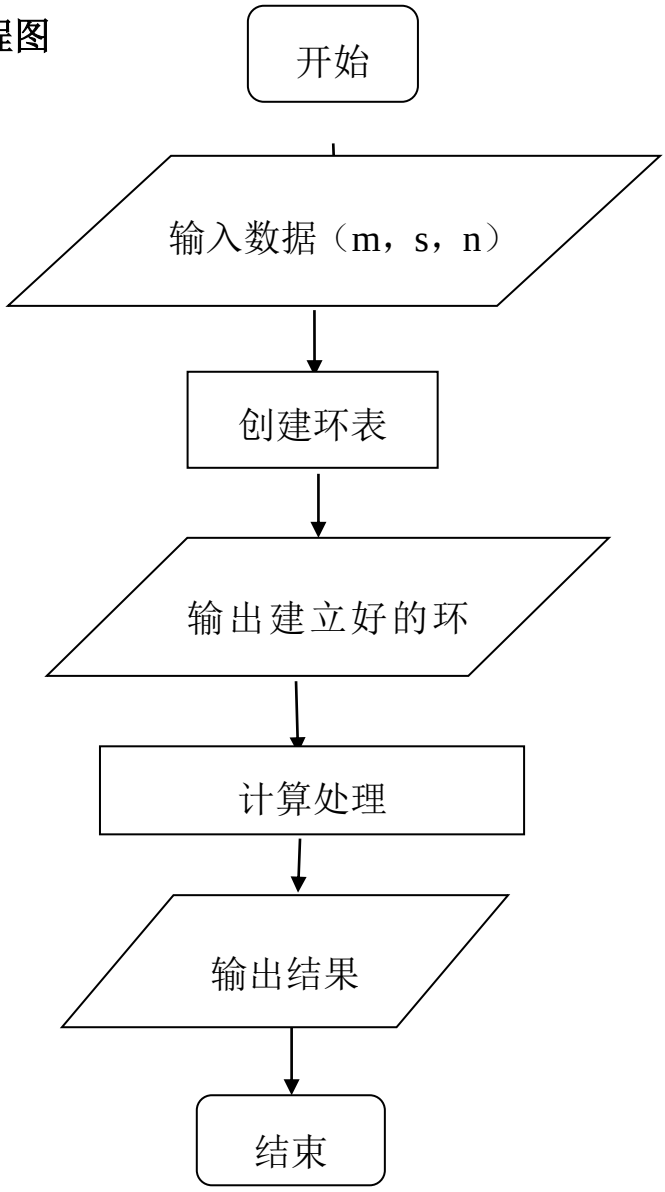
（1）建立算数表达式；

（2）得到运算表达式的值；

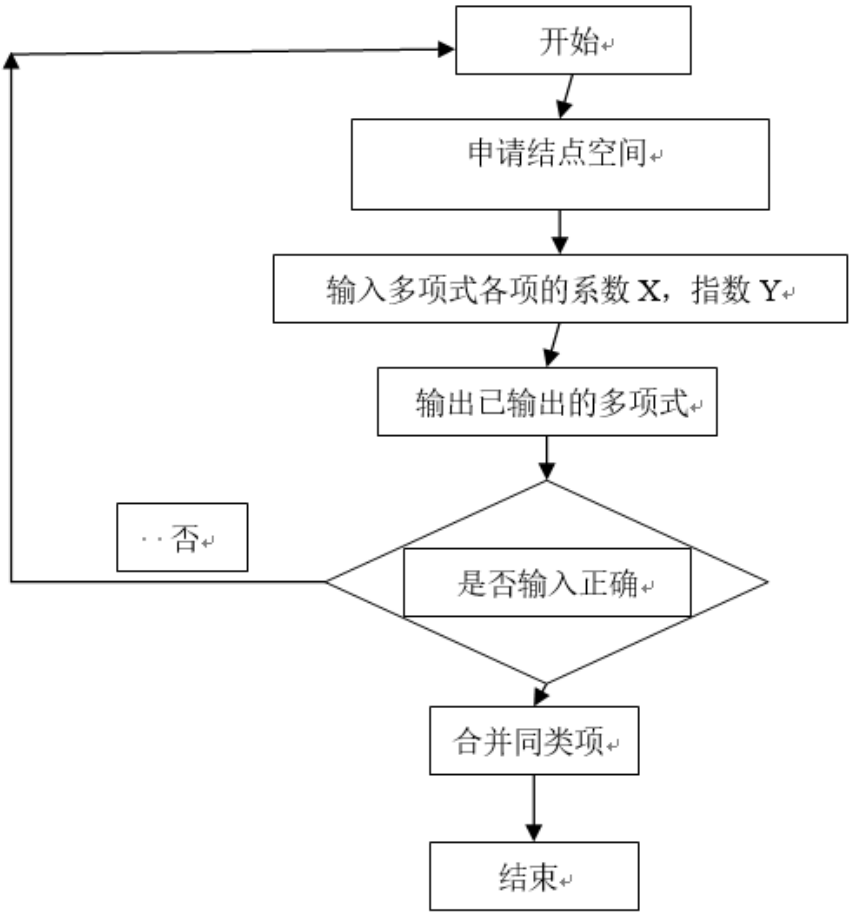
（3）演示结果

实验问题主流程图

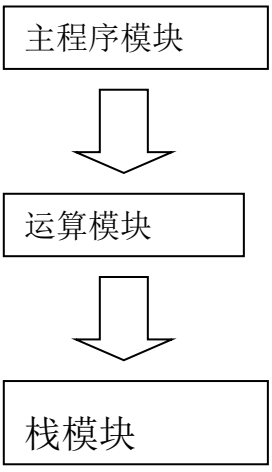
问题一



问题二



问题三



主要设备仪器

Vs2017

第二部分：实验过程记录（可加页）（包括实验原始数据记录，实验现象记录，实验过程发现的问题等）

实验代码如下

//model.h

```
#include<stdlib.h>
#include<stdio.h>
struct item {
    float coef;
    int expn;
    struct item *next;
};
struct Node {
    int flag;
    struct Node *next;
};
void calcu();
void calculator();
void runysf(int n, struct Node *head);
void Yuesefu();
struct Node* initList(int m);
void polynomial_addition();
struct item *initPoly();
void printPoly(struct item *head);
struct item *addPoly(struct item * h1, struct item *h2);
struct item *subtractPoly(struct item * h1, struct item *h2);
```

//function.cpp

```

#include<stdio.h>
#include"model.h"
#pragma warning(disable:4996);

#define n0 30
double s1[n0 + 1];
char s2[n0 + 1];
int  t1, t2;
void calcu()
{
    double x1, x2, x;
    char p;
    p = s2[t2--];
    x2 = s1[t1--];
    x1 = s1[t1--];
    switch (p)
```

```
    {
    case'+': x = x1 + x2; break;
    case'-': x = x1 - x2; break;
    case'*': x = x1 * x2; break;
    case'/': x = x1 / x2;
    }
    s1[++t1] = x;
}
void calculator() {
    char c; double v;
    t1 = t2 = 0;
    scanf("%c", &c);
    while (c != '=')
        switch (c)
        {
        case'+': case'-':
            while (t2 && (s2[t2] != '('))
                calcu();
            s2[++t2] = c;
            scanf("%c", &c);
            break;
        case'*': case'/':
            if (t2 && ((s2[t2] == '*')) || (s2[t2] == '/'))
                calcu();
            s2[++t2] = c;
            scanf("%c", &c);
            break;
        case '(':case '（':
            s2[++t2] = c;
            scanf("%c", &c);
            break;
        case ')':case '）':
            while (s2[t2] != '(')
                calcu();
            t2--;
            scanf("%c", &c);
            break;
        case '\n':case ' ':case'  ':
            scanf("%c", &c);
            break;
        default:
            v = 0;
            do
            {
```

```
            v = 10 * v + c - '0';
            scanf("%c", &c);
        } while ((c >= '0') && (c <= '9'));
        s1[++t1] = v;
    };

    while (t2)
    calcu();
    printf("%lf", s1[t1]);
}
void runysf(int n, struct Node *head) {
    int i = 1;
    struct Node *ptr = head, *ptr1 = head->next;
    while (head->next != head) {
        if (i % n == 0) {
            printf("%d ", ptr1->flag);
            ptr->next = ptr1->next;
            free(ptr1);
            ptr1 = ptr->next;
        }
        else {
            ptr1 = ptr1->next;
            ptr = ptr->next;
        }
        i++;
        if (ptr1 == head)
        {
            ptr1 = ptr1->next;
            ptr = ptr->next;
        }
    }
}
void Yuesefu()
{
    int m, n;
    struct Node *head;
    printf("输入总人数m，以及循环数n：");
    scanf_s("%d%d", &m, &n);
    head = initList(m);
    runysf(n, head);


}
struct Node* initList(int m) {
```

```
    struct Node *head, *ptr, *ptr1;
    head = (struct Node *)malloc(sizeof(struct Node));
    ptr1 = head;
    for (int i = 0; i < m; i++) {
        ptr = (struct Node *)malloc(sizeof(struct Node));
        ptr->flag = i + 1;
        ptr1->next = ptr;
        ptr1 = ptr;
    }
    ptr1->next = head;

    return head;
}
void polynomial_addition() {
    struct item *headA, *headB, *headC, *headD, *headE, *headF;
    printf("多项式相加\n");
    printf("第一项多项式项数:");
    headA = initPoly();
    printf("第二项多项式项数:");
    headB = initPoly();

    printPoly(headA);

    printf("\n");

    printPoly(headB);

    printf("\n");

    headC = addPoly(headA, headB);
    printPoly(headC);

    printf("\n");
    printf("多项式相减\n");
    printf("第一项多项式项数:");
    headE = initPoly();
    printf("第二项多项式项数:");
    headF = initPoly();

    headD = subtractPoly(headE, headF);
    printPoly(headD);
}

struct item *initPoly() {
```

```
    int num;
    struct item *head = (struct item *)malloc(sizeof(struct item)), *ptr =
head, *ptr1;

    scanf_s("%d", &num);

    for (int i = 0; i < num; i++) {
        ptr1 = (struct item *)malloc(sizeof(struct item));
        printf("请输入第%d项系数，指数：", i + 1);
        scanf_s("%f%d", &ptr1->coef, &ptr1->expn);
        ptr->next = ptr1;
        ptr = ptr->next;
    }
    ptr->next = NULL;

    return head;
}

void printPoly(struct item *head) {
    struct item *ptr = head->next;
    while (ptr != NULL) {
        if (ptr->next != NULL) {
            printf("%f*x^%d+", ptr->coef, ptr->expn);
        }
        else {
            printf("%f*x^%d", ptr->coef, ptr->expn);
        }
        ptr = ptr->next;
    }
}

struct item *addPoly(struct item * h1, struct item *h2) {
    struct item *head = h1, *pa = h1->next, *pb = h2->next;
    struct item *ppa = h1, *pc;
    while (pa != NULL && pb != NULL) {
        if (pa->expn < pb->expn) {
            ppa = ppa->next;
            pa = pa->next;
        }
        else if (pa->expn > pb->expn) {
            ppa->next = pb;
            pb = pb->next;
            ppa->next->next = pa;
            ppa = ppa->next;
```

```
        }
        else {
            if (pa->coef + pb->coef == 0)
            {
                pc = pb;
                pb = pb->next;
                free(pc);
                pc = pa;
                pa = pa->next;
                ppa->next = pa;
                free(pc);
            }
            else
            {
                pa->coef = pa->coef + pb->coef;
                pc = pb;
                pb = pb->next;
                free(pc);
            }
        }
    }
    if (pb != NULL) {
        ppa->next = pb;
    }

    return head;
}

struct item *subtractPoly(struct item * h1, struct item *h2) {
    struct item *ptr = h2->next;
    while (ptr != NULL) {
        ptr->coef = -ptr->coef;
        ptr = ptr->next;
    }

    ptr = addPoly(h1, h2);

    return ptr;
}
//main.cpp

#include<stdio.h>
#include<stdlib.h>
#include"model.h"
```

```
int main()
{
    printf("\n问题一\n");
    Yuesefu();
    printf("\n问题二\n");
    polynomial_addition();
    printf("\n问题三\n输入计算式（以=结束）\n");
    calculator();

    system("pause");
    return 0;
}
```

实验遇到的问题：

读数字时只能读取一位整数

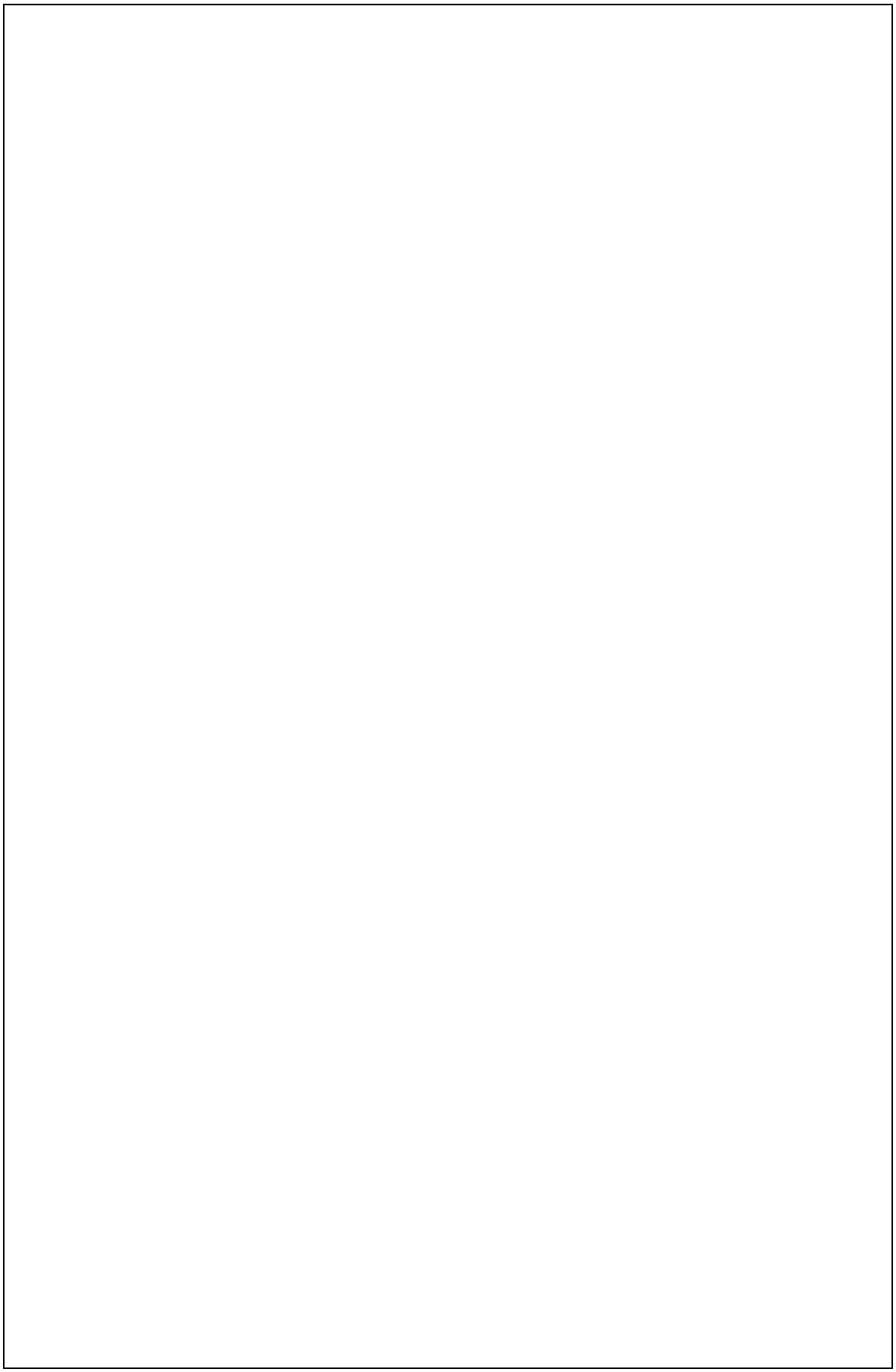
解决问题：

增加循环读入，当下一位读入的仍为整数时，进行移位处理。
另外在对小数进行操作时稍有缺陷，可以考虑读入小数点后进行计数并移位处理接下来读入的数字。

调试分析

算术表达式求值程序较为庞大，调试花费时间较多，主要是在 for 循环和 while 循环时容易出错，对于涉及的循环的操作开始和结束条件设置很关键。

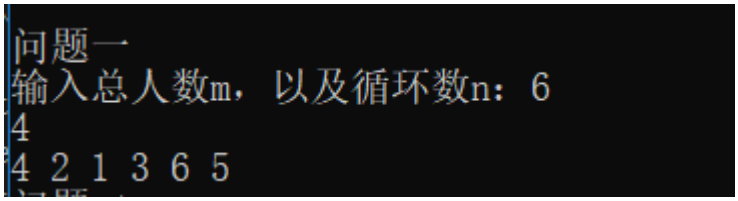
教师签字_____



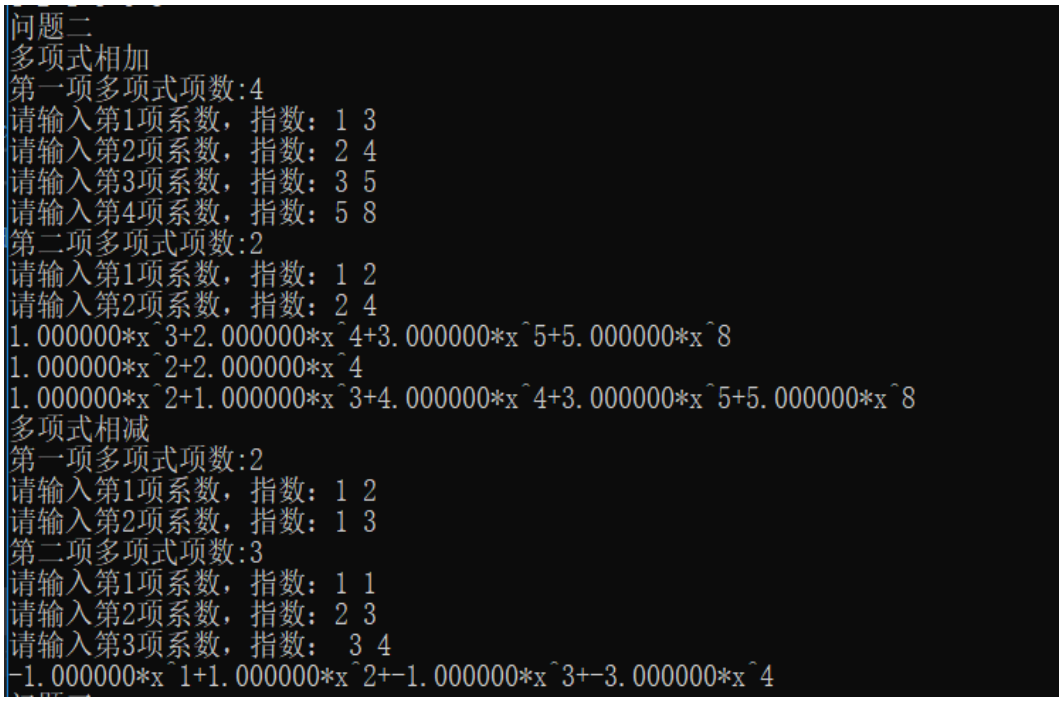
第三部分　结果与讨论（可加页）

一、实验结果分析（包括数据处理、实验现象分析、影响因素讨论、综合分析和结论等）

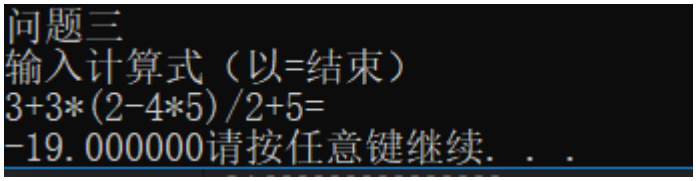
第一题：



第二题：



第三题：



二、小结、建议及体会

本次实验第一题较为简单，建一个数组，套用公式，即可得到想要的结果；

1、细节决定成败，编程最需要的是严谨，如何的严谨都不过分，往往检查了半天发现错误发生在某个括号，分号，引号，或者数据类型上。在写主要操作函数 caculate（）时，

在终止条件的书写处不是很清楚，导致我浪费了很多时间。

第二题，加法判断最后减法为调用系数为相反数的加法；

第三题，运算符优先法，创两个栈分别放数字和运算符，还算简单

三、思考题

此次实验的第二题，创建链表，我采用的是先输入多项式的项数，在创建相应长度的链表，可以优化为创建未知长度的多项式，以最后输入多项式系数和指数都为 0，作为结束符。

第三题，对于某写运算式子还是有缺陷，没又判断运算符的优先符，也许我可以加入运算符的优先判断，例遇到中括号和小括号，要先算小括号的数。