湖 州 学 院

学生实验报告

课程名称： 《数据结构课程设计》

实验项目： 一元多项式运算

实验类型： 综合型大作业

实验课时： 16课时

授课班级：

学 号:

姓 名:

指导老师：

实验时间：

实验地点：

湖州学院教务处制

**课程设计：一元多项式运算**

**一、目的要求**

1.了解线性表的特性,以及它在实际问题中的应用。

2.掌握线性表的链式存储实现方法以及基于该存储结构的基本操作的实现，学会运用线性表来解决问题。

**二、设计内容**

实现一元多项式的加法、减法和乘法。具体要求为：

用链式存储结构实现一元多项式的创建、加法、减法和乘法；然后在主函数中调用这些函数（以菜单的形式呈现）。

**三、测试数据**

多项式１：3x5+6x2+7x+1

多项式２：7x6-3x5+6x2+2

**四、数据结构的定义**

typedef struct PNode

{

float coef;

int expn;

struct PNode\* next;

}PNode, \* polynomial;

**五、源程序清单及运行结果（源程序中应该附有必要的注释，运行结果附截图）**

#include<iostream>

using namespace std;

typedef struct PNode

{

float coef;

int expn;

struct PNode\* next;

}PNode, \* polynomial;

void display();

void createpolyn(polynomial& p);

void add\_polynomial(polynomial& pa, polynomial& pb);

void subtruct\_polynomial(polynomial& p1, polynomial& p2);

void multiply\_polynomial(polynomial& p1, polynomial& p2);

void printout(const polynomial& p);

int main() {

int n;

polynomial p1 = new PNode; //为链表P1申请一个空间;

polynomial p2 = new PNode; //为链表P2申请一个空间;

display();

while (true) {

cout << "请输入要执行的操作：";

cin >> n;

switch (n) {

case 1:

createpolyn(p1);

cout << "第一个多项式为：";

printout(p1);

cout << endl;

createpolyn(p2);

cout << "第二个多项式为：";

printout(p2);

cout << endl;

break;

case 2:

//多项式相加

add\_polynomial(p1, p2);

cout << endl;

break;

case 3:

//多项式减法

subtruct\_polynomial(p1, p2);

cout << endl;

break;

case 4:

//多项式相乘

multiply\_polynomial(p1, p2);

cout << endl;

break;

case 0:

exit(1);

default:

cout << "输入不合法，请重新输入！" << endl;

cout << endl;

break;

}

}

return 0;

}

void display()

{

cout << "\*\*\*\*\*\*\*\*\*\*多项式的加减乘法\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "1.输入多项式" << endl;

cout << "2.多项式相加" << endl;

cout << "3.多项式相减" << endl;

cout << "4.多项式相乘" << endl;

cout << "0.退出" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void createpolyn(polynomial& p)

{

p = new PNode;

p->next = NULL;

int n;

cout << "请输入多项式的项数：";

cin >> n;

polynomial pre;

polynomial q;

for (int i = 1; i <= n; i++)

{

polynomial s = new PNode;

cout << "请输入第" << i << "项的系数和指数：";

cin >> s->coef >> s->expn;

pre = p;

q = p->next;

while (q && q->expn < s->expn)

{

pre = q;

q = q->next;

}

s->next = q;

pre->next = s;

}

}

void printout(const polynomial& p)

{

polynomial current = p->next;

bool isFirstTerm = true; //标识当前是否是第一个非零项

while (current) {

int coef = current->coef;

int expn = current->expn;

if (coef != 0) {

if (!isFirstTerm && coef > 0) { //用在打印 "+" 符号之前，如果是第一个非零项，不需要打印 "+

cout << "+";

}

if (expn == 0) {

cout << coef;

}

else {

if (coef != 1 && coef != -1) {

cout << coef;

}

else if (coef == -1) {

cout << "-";

}

if (expn == 1) {

cout << "x";

}

else {

cout << "x^" << expn;

}

}

isFirstTerm = false;

}

current = current->next;

}

if (isFirstTerm) {

cout << "0";

}

cout << endl;

}

void add\_polynomial(polynomial& pa, polynomial& pb)

{

polynomial p1 = pa->next;

polynomial p2 = pb->next;

polynomial result = new PNode; //创建新的多项式来保存结果

result->next = nullptr;

polynomial p3 = result; // p3用于指向结果多项式的最后一个节点

while (p1 && p2) {

if (p1->expn == p2->expn)

{

int sum;

sum = p1->coef + p2->coef;

if (sum != 0)

{

polynomial r = new PNode;

r->coef = sum;

r->expn = p1->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

}

p1 = p1->next;

p2 = p2->next;

}

else if (p1->expn < p2->expn) {

polynomial r = new PNode;

r->coef = p1->coef;

r->expn = p1->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

p1 = p1->next;

}

else {

polynomial r = new PNode;

r->coef = p2->coef;

r->expn = p2->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

p2 = p2->next;

}

}

while (p1) {

polynomial r = new PNode;

r->coef = p1->coef;

r->expn = p1->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

p1 = p1->next;

}

while (p2) {

polynomial r = new PNode;

r->coef = p2->coef;

r->expn = p2->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

p2 = p2->next;

}

cout << "相加的结果为：" << endl;

printout(result); //输出结果

}

void subtruct\_polynomial(polynomial& pa, polynomial& pb)

{

polynomial p1 = pa->next;

polynomial p2 = pb->next;

polynomial result = new PNode; //创建新的多项式来保存结果

result->next = nullptr;

polynomial p3 = result; // p3用于指向结果多项式的最后一个节点

while (p1 && p2) {

if (p1->expn == p2->expn) {

int diff = p1->coef - p2->coef;

if (diff != 0) {

polynomial r = new PNode;

r->coef = diff;

r->expn = p1->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

}

p1 = p1->next;

p2 = p2->next;

}

else if (p1->expn < p2->expn) {

polynomial r = new PNode;

r->coef = p1->coef;

r->expn = p1->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

p1 = p1->next;

}

else {

polynomial r = new PNode;

r->coef = -p2->coef;

r->expn = p2->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

p2 = p2->next;

}

}

while (p1) {

polynomial r = new PNode;

r->coef = p1->coef;

r->expn = p1->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

p1 = p1->next;

}

while (p2) {

polynomial r = new PNode;

r->coef = -p2->coef;

r->expn = p2->expn;

r->next = nullptr;

p3->next = r;

p3 = r;

p2 = p2->next;

}

cout << "相减的结果为：" << endl;

printout(result); //输出结果

}

void multiply\_polynomial(polynomial& pa, polynomial& pb)

{

polynomial p1 = pa->next;

polynomial p2 = pb->next;

polynomial result = new PNode; //创建新的多项式来保存结果

result->next = nullptr;

while (p1) {

polynomial tempP2 = p2;

while (tempP2) {

// 创建一个新节点，保存 p1 项和 p2 项的乘积

polynomial r = new PNode;

r->coef = p1->coef \* tempP2->coef;

r->expn = p1->expn + tempP2->expn;

r->next = nullptr;

// 将新节点插入到结果多项式中，按照指数的顺序插入

polynomial pre = result;

polynomial cur = result->next;

while (cur && cur->expn < r->expn) {

pre = cur;

cur = cur->next;

}

if (cur && cur->expn == r->expn) {

// 如果结果多项式中已经有了相同指数的项，则将它们的系数相加

cur->coef += r->coef;

delete r;

}

else {

// 否则，将新节点插入到正确的位置

r->next = cur;

pre->next = r;

}

tempP2 = tempP2->next;

}

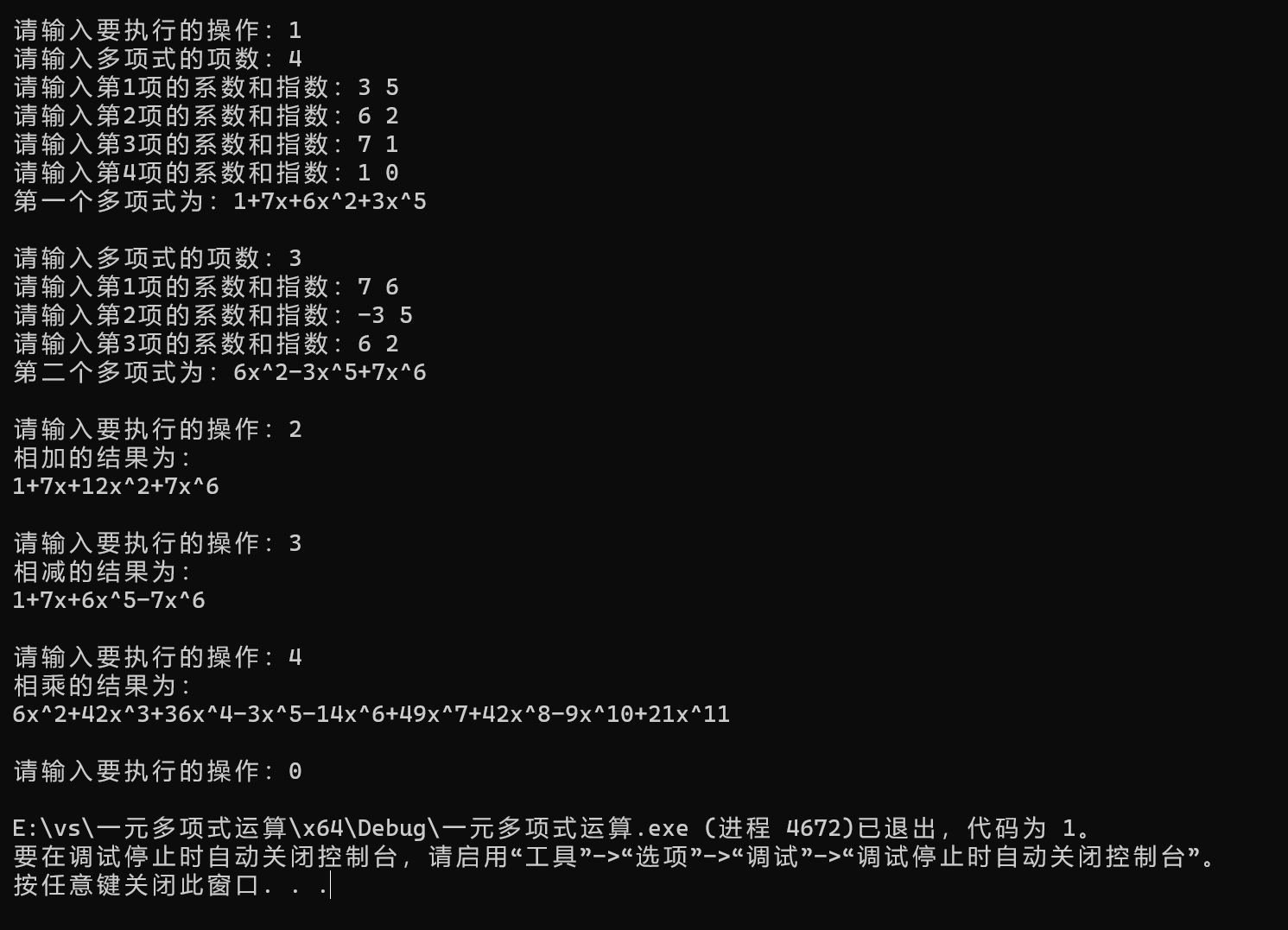
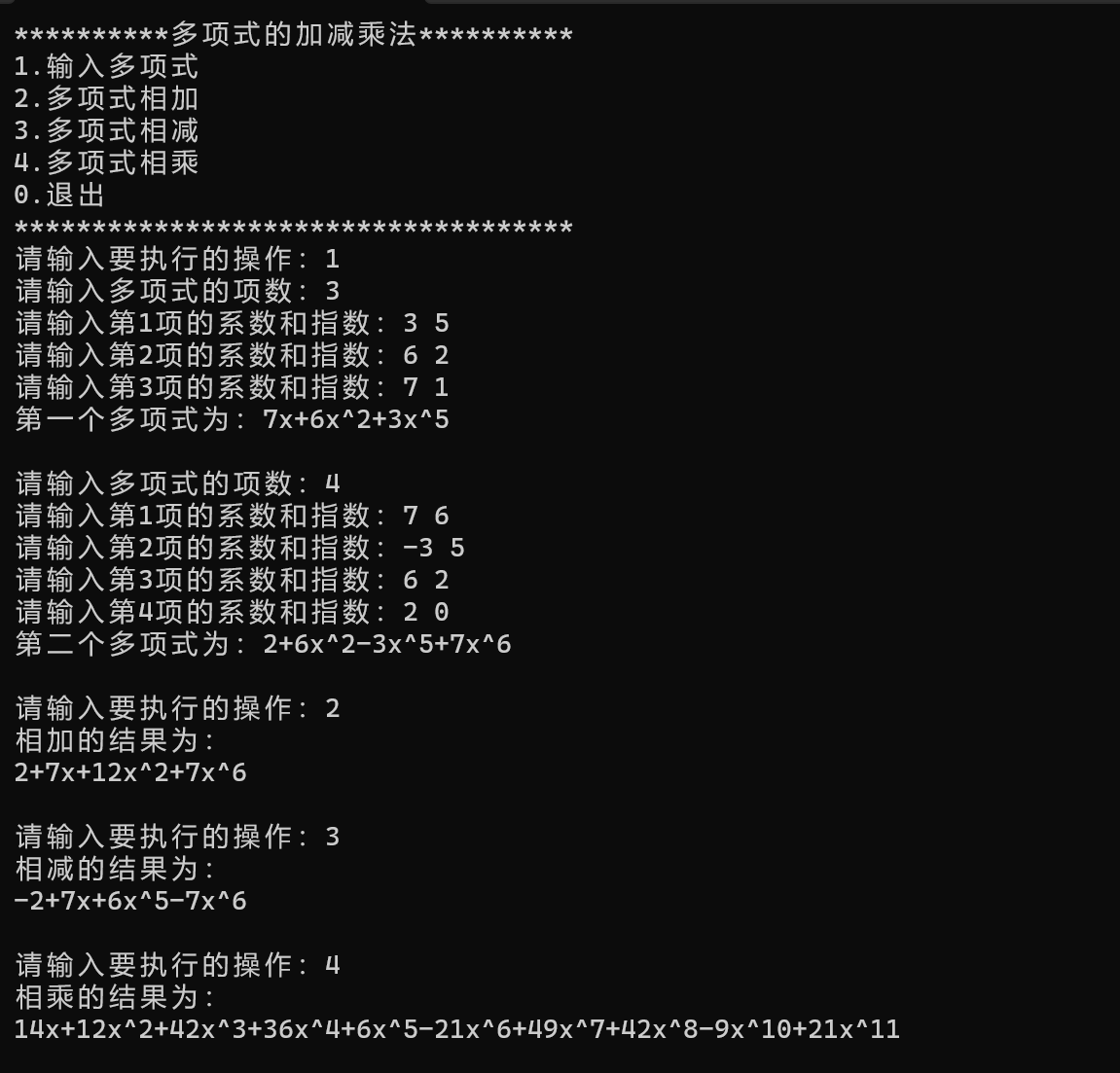
p1 = p1->next;

}

cout << "相乘的结果为：" << endl;

printout(result); //输出结果

}

****

**六、小结**

1.详细说明遇到的问题及解决方法；

文本

描述已自动生成

错误原因：

文本

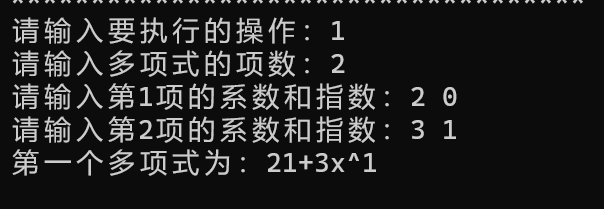
描述已自动生成



应该为

文本

描述已自动生成



在输出多项式时，遇到指数为0和1时格式错误

解决办法：重新修改函数增加面对1/-1/0时的操作

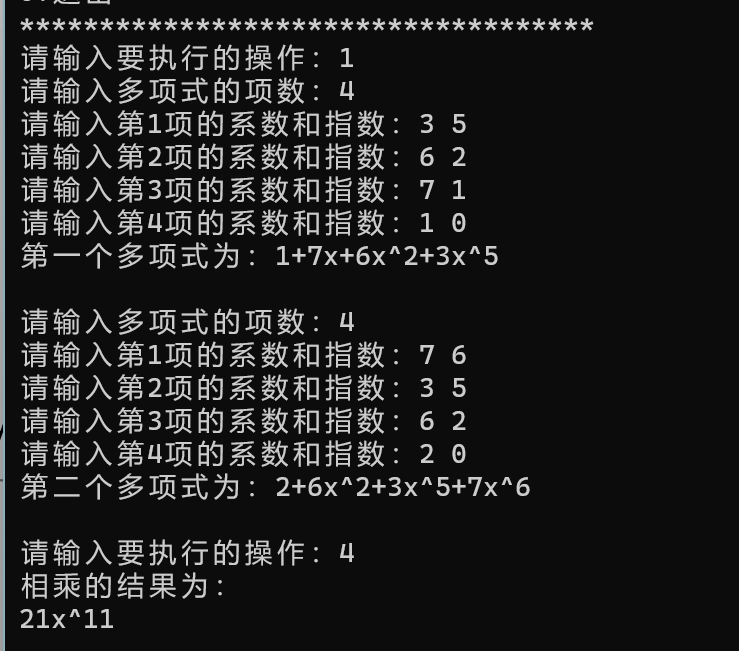
在输入2后会导致程序卡住

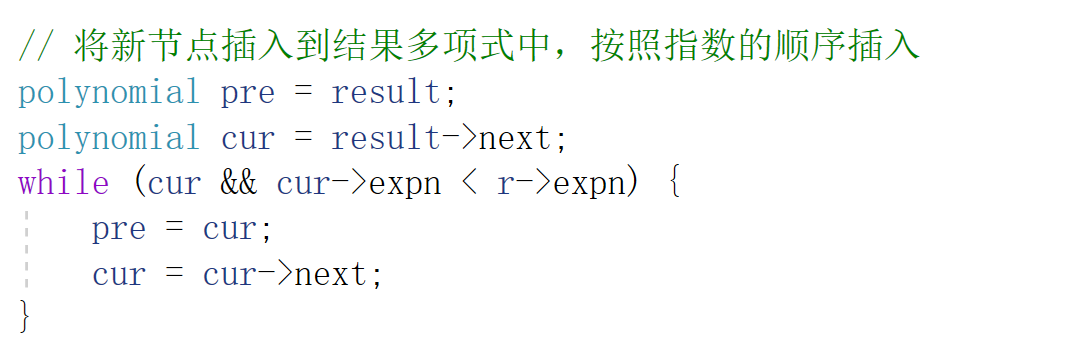
图形用户界面, 应用程序

描述已自动生成

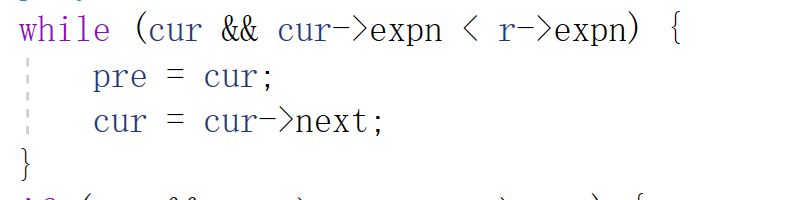
错误原因：程序在已有的 p1 和 p2 上进行修改，但是在合并之后，p3变量并没有正确地指向结果多项式的头部。

解决办法：创建一个新的多项式来保存结果，然后正确地处理 p1 和 p2 的剩余项。（在处理减法时也遇到该错误，解决办法同理）





错误原因：在判断时只对尾节点进行了后移，未对头节点进行后移，实际上在插入新节点时需要插入两节点之间

解决办法：将头节点与尾节点一起后移

错误原因2：



Tempp2只进行了一次乘法

解决办法：增加一个while循环





2.关于程序的特色和改进设想；

引入临时变量r与result来储存中间变量与结果，使过程更易理解与操作

3.经过此实验所得到的收获,其他需要说明的情况。

1. 编程技能的提高：实现这样的程序需要用到很多基本的编程技能，如数据结构（链表）的使用，循环和条件语句的使用，函数的定义和调用等。通过实践，我更好地掌握了这些技能。
2. 算法理解和实现：程序中包含了多项式相加、相减、相乘等算法的实现，通过编程实现这些算法，可以帮助我更深入地理解这些算法。
3. 问题解决能力：在编程过程中，遇到了各种问题，如逻辑错误、语法错误等。通过解决这些问题，提高了问题解决能力。
4. 代码调试和优化：在编程过程中，需要不断进行代码调试和优化。这不仅可以帮助我找出和修复程序中的错误，还可以提高程序的效率。
5. 抽象思维能力：在编程中，需要将现实世界中的问题抽象化，然后用链表实现。这提高了我抽象思维能力。