 实 验 报 告

课程名称： 数据结构（C++版）

题 目： 一元多项式相加

专业班级： 18空间

姓 名： 邓珊珊

学 号： 1812001107

指导老师： 袁莹

实验时间： 2019.11.29

第一次实验报告

班级：18空间 姓名：邓珊珊 学号：1812001107

1. 实验目的及原理

（选择公式的原理，及文字说明，所选的公式需要输入什么，计算出什么，如何计算，目的：通过实际问题学会用函数编程实现）

1. 问题描述：

已知A（x） = a0 +a1x + a2x^2 +… + anx^n和B(x) = b0 + b1x + b2x^2 + … +bmx^m,并且在A(x) 和 B（x）中指数相差很多，求A（x）= A(x)+B(x)。

1. 基本要求：
2. 设计存储结构表示一元多项式。
3. 设计算法实现一元多项式相加。
4. 分析算法的时间复杂度和空间复杂度。
5. 实验内容

（代码）

#include <iostream>

using namespace std;

struct Node {

double coe; //系数

int exp; //指数

Node \*next;

};

void CreatPoly(Node \*&head, int n) // 生成带表头结点的单链表，除头结点外另生成n个结点

{

head = (Node \*)new Node;

head->coe = 0;

head->exp = 0;

head->next = NULL; // 初始化头结点

cout << "分别每行输入各项系数及指数：" << endl;

Node \*p = head;

for(int i = 0; i < n; i++) {

p->next = (Node \*)new Node; // 生成新结点，尾插入生成链表

p = p->next;

cin >> p->coe >> p->exp;

p->next = NULL;

}

}

void ShowPoly(Node \*&head)

{

if(head->next == NULL) // 结果是0时直接输出0

putchar('0');

else {

for(Node \*p = head->next; p != NULL; p = p->next) {

if(p != head->next && p->coe >0) // 当p非首项且指向的系数为正时才输出'+'

putchar('+'); // 之前只判定了p->coe >0

if(p->coe == 1) { // 系数为1或-1时特殊处理

if(p->exp == 0)

putchar('1'); // 判断条件不能写在一起：

} // if(p->coe == 1 && p->exp == 0) putchar('1');

else if(p->coe == -1)

putchar('-');

else

cout << p->coe;

// 指数为0或1时特殊处理

switch(p->exp) {

case 0:

break;

case 1:

putchar('x');

break;

default:

p->exp < 0 ? printf("x^(%d)", p->exp) : printf("x^%d", p->exp); // 指数小于0时打括号

break;

}

}

}

cout << endl;

}

char comp(int a, int b)

{

if(a > b)

return '>';

if(a < b)

return '<';

return '=';

}

void Free(Node \*&head)

{

Node \*q = NULL;

for(Node \*p = head; p != NULL; p = q) {

q = p->next;

free(p);

}

}

void AddPolynomial(Node \*&pA, Node \*&pB) // 传进两个链表的头指针

{

Node \*ha = pA;

Node \*hb = pB;

Node \*qa = ha->next; // ha, hb分别跟在qa, qb的后一位置

Node \*qb = hb->next; // qa, qb分别指向Pa, Pb中当前比较元素

while(qa && qb)

{

double sum = 0;

int a = qa->exp;

int b = qb->exp;

switch( comp(a, b) ) {

case '<':

ha = qa;

qa = qa->next; // 非ha = ha->next;

break;

case '=':

sum = qa->coe + qb->coe;

if(sum != 0.0) {

qa->coe = sum;

ha = qa;

}

else {

if(ha->next != qa)

cout << "Error: ha->next != qa" << endl;

ha->next = ha->next->next; // 删除和为0的结点，ha不变，还在qa后一位置

free(qa);

}

if(hb->next != qb)

cout << "Error: hb->next != qb" << endl;

hb->next = hb->next->next;

free(qb);

qb = hb->next;

qa = ha->next;

break;

case '>':

hb->next = hb->next->next; // 删除qb指向的结点

qb->next = ha->next; // 将qb插入ha后qa前

ha->next = qb;

qb = hb->next; // not qb = ha->next

ha = ha->next;

break;

default:

cout << "Error!" << endl;

break;

}

}

if(qb)

ha->next = qb;

free(hb);

}

int main(void)

{

Node \*A = NULL;

Node \*B = NULL;

int countA;

int countB;

cout << "请输入A的项数：" << endl; cin >> countA;

CreatPoly(A, countA); // 生成A链表

cout << "请输入B的项数：" << endl; // 生成B链表

cin >> countB;

CreatPoly(B, countB);

cout << " A = ";

ShowPoly(A);

cout << " B = ";

ShowPoly(B);

AddPolynomial(A, B); // A = A + B

cout << "A+B= ";

ShowPoly(A); // 输出相加的和

cout << endl;

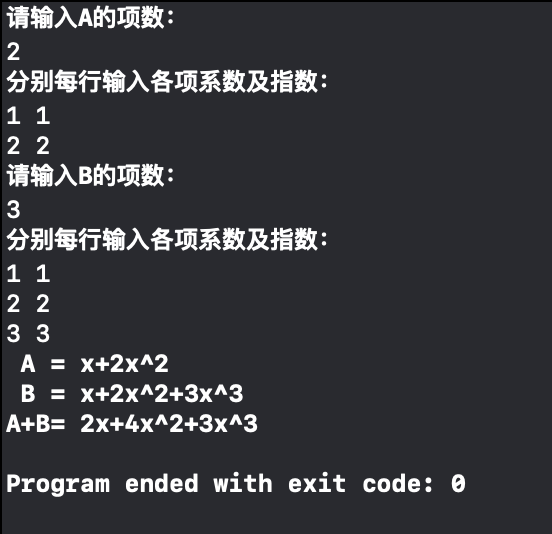
// 释放结点

delete A;

}

1. 实验成果

（运行结果截图）



1. 实验心得

在定义结构体指针的时候，最好给其赋值为空，谨防出现野指针的情况。