**数据结构实验报告**

**实验一 一元多项式的加减运算中的应用**

学 生 姓 名 ：杨仕振

学 号 ：2015301500349

二○一六年十月

**一、简介**

编写一个程序，完成一元多项式的存储，并实现两个多项式的相加运算。要求随机从键盘上输入多项式，构建单链表存储一元多项式，其中单链表的数据域包括系数和指数两项。且最后的结果输出形式为result=3+4x^1+5x^2+...

**二、算法说明**

实验所需用到的数据结构：

多项式的一项term

term{

指数

系数

next指针

}

实验中的算法：

main

{

提示”请按升幂顺序输入第一个多项式,输入-1停止”;

创建第一个多项式对应的链表;

显示第一个多项式;

提示”请按升幂顺序输入第二个多项式,输入-1停止”;

创建第二个多项式对应的链表;

显示第二个多项式;

将两个多项式合并;

显示合并后的多项式;

销毁两个多项式;

返回 0;

}

创建多项式

CreatePoly

{ 初始化链表；

while（1）｛

输入项的幂和系数；

if 项的幂为-1 break；

创建新的项；

将项加入到链表中；

｝

末尾的next指针置空；

}

显示多项式

DispPoly

{

If（链表不空）

｛ While（1）

｛ 指针指向下一个项；

If 幂不为零

输出此项；

Else 输出系数；

If 下个项为空 break；

输出”+”；

｝

｝

Else 提示重输多项式；

}

合并多项式

AddPoly

{

链表一二指向头节点;

While(链表一不空)

{ While（链表二不空）

{If（链表一此项幂与链表二此项相同）

{

将链表二的系数加到一上；

并删去链表二中的这个节点；

链表二指向下个节点；break；

}

链表二指向下个节点；

}

链表二指向头节点；

链表一指向下个节点；

}

}

销毁多项式

DestroyPoly

{ 指向头节点的下一个节点p；

While（p节点不空）

{释放头节点；

头结点指向下一个节点；

记录节点p后移；

}

释放头节点；

}

**三、测试结果**

表**1 测试结果表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试输入 | 测试目的 | 正确输出 | 实际输出 | 错误原因 | 当前状态 |
| 0,1;-1;1,2;-1 | 测试幂为0能否正常输出 | 1.000000+2.000000x^1 | 1.000000+2.000000x^1 | 无 | 通过 |
| 0,1/1,2;-1;1,2/2,3 | 测试有多个项时是否正常输出 | 1.000000+  4.000000x^1+  3.000000x^2 | 1.000000+  4.000000x^1+  3.000000x^2 | 无 | 通过 |
| 0,1/1,2;-1;1,-1/2,3 | 测试带有负系数的项是否能够正常输出 | 1.000000+  1.000000x^1+  3.000000x^2 | 1.000000+  1.000000x^1+  3.000000x^2 | 无 | 通过 |

**四、分析与探讨**

1.测试结果分析

通过测试，可以只要按照程序的要求进行输入，就可以保证结果数值的正确性。

2.时空复杂度分析

时间复杂度：

CreatePoly:O(n)

DispPoly:O(n)

AddPoly:最坏为O(n²)，最好为O(n)

DestroyPoly:0(n)

空间复杂度：

CreatePoly:O(1)

DispPoly:O(1)

AddPoly:O(1)

DestroyPoly:0(1)

3.存在的问题

(1)当幂为-1的时候，无法进行记录。（因为输入终止的条件就是幂=-1）

(2)用户输入相对不友好，没有算式解析的算法，输入相对麻烦一些。

(3)在显示时，若某项为负系数项，则会产生多余的符号+，虽然并不影响值的正确性，但是有碍美观。

**附录：源代码**

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int pow;//定义多项式的幂

float coe;//定义多项式的系数

struct node \*next;

} term;//定义多项式的一项

void CreatePoly(term \*&L);//创建多项式

void DispPoly(term \*&L);//显示多项式

void AddPoly(term \*&p, term \*&q);//将两个多项式合并

void DestroyPoly(term \*&L);//将多项式摧毁

int main()

{

term \*i, \*ii;

printf("请按升幂顺序输入第一个多项式,输入-1停止\n");

CreatePoly(i);

printf("第一个多项式为：");

DispPoly(i);

printf("\n请按升幂顺序输入第二个多项式,输入-1停止\n");

CreatePoly(ii);

printf("第二个多项式为：");

DispPoly(ii);

AddPoly(i, ii);

printf("\n相加之后的结果是：");

DispPoly(i);

DestroyPoly(i);

free(ii);

system("PAUSE");

return 0;

}

void CreatePoly(term \*&L)//创建多项式

{

int a;//记录幂

float b;//记录系数

term \*n, \*m;

L = (term \*)malloc(sizeof(term));

L->next = NULL;//初始化多相似

m = L;

printf("请输入幂,系数");

while (1) {

scanf\_s("%d,%f", &a, &b);

if (a == -1) break;//设定输入停止的条件

n = (term \*)malloc(sizeof(term));

n->pow = a;

n->coe = b;//创建并将新的项添加到多项式中

m->next = n;

m = n;

}

m->next = NULL;

}

void DispPoly(term \*&L)//显示多项式

{

term \*p;

p = L;

if (p != NULL&&p->next != NULL)//判断多项式是否到达尾部

{

while (1)

{

p = p->next;//将指针指向下一个

if (p->pow != 0)

printf("%fx^%d", p->coe, p->pow);

else

printf("%f", p->coe);//如果幂是零，那么只显示系数

if (p->next == NULL) break;/\*如果p是最后一个项，

那么后面不放置"+"\*/

printf("+");//通过循环来放置“+”

}

}

else printf("请先输入多项式！");

}

void AddPoly(term \*&p, term \*&q)//将多项式相加

{

term \*x = p->next;//设置用来扫描第一个多项式的指针

term \*y = q->next;//设置用来扫描第二个多项式的指针

term \*temp = p;//记录x的上一个节点

term \*pre = q;//记录y的上一个节点

while (x != NULL)

{

while (y != NULL)

{

if (x->pow == y->pow)

{

x->coe = x->coe + y->coe;//将两个多项式里幂相同项的系数相加

pre->next=y->next;

free(y);//将第二个多项式里幂相同的项删除

y = q->next;

break;

}

pre = y;

y = y->next;

}

pre=q;

y = pre->next;//从第二个多项式的第一项开始扫描

x = x->next;//开始匹配第一个多项式的下一项

}

//至此，两个多项式的每一项的幂都各不相同

x = p->next;

while (x != NULL)

{

while (y != NULL)

{

if (y->pow < x->pow)

{

pre->next = y->next;

y->next = x;

temp->next = y;

pre = pre->next;/\*将第二个多项式的每一项按照幂的大小顺序插入

第一个多项式\*/

if(pre!=NULL)

y = pre->next;//如果pre不为空，那么就把y继续向后指

else

y = pre;//如果为空，那么就把y也指向NULL

}

else break;//如果y所指项的幂比第一个多项式的幂都大，那么结束循环

}

temp = x;

x = x->next;//继续指向下一个x

}

if (y != NULL) temp->next = y;/\*如果ii多项式的某一项开始幂都大于i多项式，

那么就把ii接到i链的尾部\*/

//至此，幂不同的项便从第二个多项式“移”到了第一个多项式上

}

void DestroyPoly(term \*&L)//销毁多项式

{

term \*q = L->next;

while (q != NULL) {

free(L);

L = q;

q = q->next;

}

free(L);

}