**实验一 一元多项式计算器**

实验学时：2 实验类型：综合类型

**一、目的与任务**

1．目的：掌握顺序表和单链表的存储特点及插入、删除等算法。

2．任务：灵活运用顺序表和单链表的相关算法实现一元多项式的计算。

## 二、内容、要求与安排方式

1．实验内容：设有一元多项式 Am(x)和 Bn(X)，编程实现多项式 Am(x)和 Bn(x)的加法、减法和乘法运算。其中多项式描述为：

Am(x)=A0+A1x1+A2x2+A3x3+….+Amxm；

Bn(x)=B0+B1x1+B2x2+B3x3+….+Bnxn。

2．输入和输出：

（1）输入：

* 从键盘输入运算指令（相加、相减、相乘），根据运算指令进行相应运算；
* 从键盘输入两个多项式的系数和指数；
* 系数和指数采用 int 类型，运算结果不超出 int 取值范围。

（2）输出：

* 每种运算结果以多项式形式输出，要输出升幂和降幂两种情况。
* 结果多项式中无重复阶项、无零系数项，输出多项式时请采用如下易读形式（一元多项式，总变元为 x）： x^4 - 3 x^2 + 5

3．实验要求：

* 实现一个简单的交互式界面，包括系统菜单、输入提示等。
* 多项式运算前首先判定多项式特点，根据多项式是否稀疏来选用合适的存储结构；
* 根据多项式不同的运算要求选择合适的存储结构；
* 上机编辑、调试出完整正确的程序，包括相加、相减、相乘运算。

4．实验安排方式：

* 在实验课前编写出完整程序，在实验课时进行调试；
* 每组 1 人，独立完成上机实验。

## 三、注意事项：

1．本实验内容涉及到链式存储结构，请理解链式存储的特点及指针的用法。

2．注意比较带头结点、无头结点链表实现插入、删除算法时的区别。

3．请在实验报告中说明系统使用的主要数据结构。

## 四、源码及结果

## 多项式加法：

#include <stdio.h>  
#include <stdlib.h>  
  
typedef struct PolynNote\* Polynomial;  
typedef struct PolynNote{  
 int coef; *//系数* int expn; *//指数* Polynomial next;  
}PolynNote;  
  
void AddPolyn(Polynomial Pa,Polynomial Pb);  
int cmp(Polynomial a,Polynomial b);  
void CreatPolyn(Polynomial L,int n);  
void print(Polynomial L);  
int main() {  
 Polynomial L1=(Polynomial)malloc(sizeof(PolynNote));  
 Polynomial L2=(Polynomial)malloc(sizeof(PolynNote));  
 int n1,n2;  
 printf("请输入第一个多项式的项数：\n");  
 scanf("%d", &n1);  
 CreatPolyn(L1,n1);  
 printf("请输入第二个多项式的项数：\n");  
 scanf("%d", &n2);  
 CreatPolyn(L2,n2);  
 AddPolyn(L1,L2);  
 print(L1);  
}  
int cmp(Polynomial a,Polynomial b){  
 if (a->expn > b->expn) return -1;  
 else if (a->expn < b->expn) return 1;  
 else return 0;  
}  
void AddPolyn(Polynomial Pa,Polynomial Pb){  
 int sum;  
 Polynomial qa;  
 Polynomial qb;  
 Polynomial p=Pa;  
 qa=Pa->next; qb=Pb->next;  
 while(qa!=**NULL** && qb!=**NULL**){  
 switch(cmp(qa,qb)){  
 case 0:  
 sum = qa->coef+qb->coef;  
 if(sum!=0) p->next->coef=sum;  
 qa=qa->next;  
 qb=qb->next;  
 break;  
 case 1:  
 p->next=qa;  
 qa=qa->next;  
 break;  
 case -1:  
 p->next = qb;  
 qb=qb->next;  
 break;  
 }  
 p=p->next;  
 }  
 if(qa!=**NULL**) p->next=qa;  
 else p->next=qb;  
}  
*//创建多项式*void CreatPolyn(Polynomial L,int n){  
 Polynomial q; *//创建一个新的节点* for(int i=1;i<=n;i++){  
 q=(Polynomial)malloc(sizeof(PolynNote));  
 q->next = **NULL**;  
 printf("输入第%d项的系数和指数:\n",i);  
 scanf("%d%d",&q->coef,&q->expn);  
 L->next=q;  
 L=L->next; *//把节点链到多项式中* }  
}  
void print(Polynomial L){*//输出单链表* Polynomial p;  
 p = (Polynomial)malloc(sizeof(PolynNote));  
 p = L->next;  
 while(p->next != **NULL**){*//如果p的后继节点不为空时执行下列语句* if(p->coef != 0 && p->expn == 0){*//系数不为0指数为0的情况（常数）* printf("%d + ",p->coef);*//所以只输入系数* }  
 else if(p->coef == 1 && p->expn == 1){*//系数为1指数为1的情况* printf("x + ");  
 }  
 else if(p->coef != 0 && p->expn == 1){*//系数不为0指数为1的情况（x，2x，5x……）* printf("%dx + ",p->coef);*//注意前面%d后的x* }  
 else if(p->coef == 1 && p->expn != 0){*//系数为1指数不为0的情况（x^2，x^5……）* printf("x^%d + ",p->expn);  
 }  
 else if(p->coef == 0){*//两个多项式相加的和为0时的情况，不输出* printf("");  
 }  
 else if(p->coef != 1 && p->expn != 0){*//系数不为1指数不为0的情况（2x^3,3x^2……）* printf("%dx^%d + ",p->coef,p->expn);  
 }  
 p = p->next;*//让p等于它的后继结点，继续循环* }  
 *//判断最后一个多项式的类型 （ps：因为上述循环当中我们定义p->next!=NULL时执  
 // 行，那么当p为最后一个结点时，p->next已经是空了，那么这个时候就会跳出循环，  
 // 最后一个结点的数据就会不运行出来，所以我们需要另写，如果不想写这么多代码的  
 // 话让它在循环内输出也很简单，大家可以动脑筋想一想），下列语句同上* if(p->expn == 0 && p->coef != 0){  
 printf("%d",p->coef);  
 }  
 else if(p->coef == 1 && p->expn == 1){*//系数为1指数为1的情况* printf("x +");  
 }  
 else if(p->expn == 1 && p->coef != 0){  
 printf("%d x",p->coef);  
 }  
 else if(p->coef == 1 && p->expn != 0){  
 printf(" x^%d",p->expn);  
 }  
 else if(p->coef == 0){  
 printf("0"); *//这条语句和上面不一样，因为这是最后一个多项式的输出，如果最后一个多项式的计算结果为0，那么不这样写就会输出（……+）有一个加号，不美观* }  
 else if(p->coef != 1 && p->expn != 0){  
 printf("%dx^%d",p->coef,p->expn);  
 }  
}

## 多项式减法

#include <stdio.h>  
#include <stdlib.h>  
  
typedef struct PolynNote\* Polynomial;  
typedef struct PolynNote{  
 int coef; *//系数* int expn; *//指数* Polynomial next;  
}PolynNote;  
  
void AddPolyn(Polynomial Pa,Polynomial Pb);  
int cmp(Polynomial a,Polynomial b);  
void CreatPolyn(Polynomial L,int n);  
void print(Polynomial L);  
int main() {  
 Polynomial L1=(Polynomial)malloc(sizeof(PolynNote));  
 Polynomial L2=(Polynomial)malloc(sizeof(PolynNote));  
 int n1,n2;  
 printf("请输入第一个多项式的项数：\n");  
 scanf("%d", &n1);  
 CreatPolyn(L1,n1);  
 printf("请输入第二个多项式的项数：\n");  
 scanf("%d", &n2);  
 CreatPolyn(L2,n2);  
 AddPolyn(L1,L2);  
 print(L1);  
}  
int cmp(Polynomial a,Polynomial b){  
 if (a->expn > b->expn) return -1;  
 else if (a->expn < b->expn) return 1;  
 else return 0;  
}  
void AddPolyn(Polynomial Pa,Polynomial Pb){  
 int sum;  
 Polynomial qa;  
 Polynomial qb;  
 Polynomial p=Pa;  
 qa=Pa->next; qb=Pb->next;  
 while(qa!=**NULL** && qb!=**NULL**){  
 switch(cmp(qa,qb)){  
 case 0:  
 sum = qa->coef-qb->coef;  
 if(sum!=0) p->next->coef=sum;  
 qa=qa->next;  
 qb=qb->next;  
 break;  
 case 1:  
 p->next=qa;  
 qa=qa->next;  
 break;  
 case -1:  
 p->next = qb;  
 qb=qb->next;  
 break;  
 }  
 p=p->next;  
 }  
 if(qa!=**NULL**) p->next=qa;  
 else p->next=qb;  
}  
*//创建多项式*void CreatPolyn(Polynomial L,int n){  
 Polynomial q; *//创建一个新的节点* for(int i=1;i<=n;i++){  
 q=(Polynomial)malloc(sizeof(PolynNote));  
 q->next = **NULL**;  
 printf("输入第%d项的系数和指数:\n",i);  
 scanf("%d%d",&q->coef,&q->expn);  
 L->next=q;  
 L=L->next; *//把节点链到多项式中* }  
}  
void print(Polynomial L){*//输出单链表* Polynomial p;  
 p = (Polynomial)malloc(sizeof(PolynNote));  
 p = L->next;  
 while(p->next != **NULL**){*//如果p的后继节点不为空时执行下列语句* if(p->coef > 0 && p->expn == 0){*//系数大于0指数为0的情况（常数）* printf("%d + ",p->coef);*//所以只输入系数* }

else if(p->coef < 0 && p->expn == 0){*//系数<0指数为0的情况（常数）* printf("**-**%d + ",p->coef);*//所以只输入系数* }  
 else if(p->coef == 1 && p->expn == 1){*//系数为1指数为1的情况* printf("x + ");  
 }

else if(p->coef == -1 && p->expn == 1){*//系数为-1指数为1的情况* printf("-x + ");  
 }  
 else if(p->coef != 0 && p->expn == 1){*//系数>0指数为1的情况（x，2x，5x……）* printf("%dx + ",p->coef);*//注意前面%d后的x* }

else if(p->coef != 0 && p->expn == 1){*//系数<0指数为1的情况（x，2x，5x……）* printf("-%dx + ",p->coef);*//注意前面%d后的x* }  
 else if(p->coef == 1 && p->expn != 0){*//系数为1指数不为0的情况（x^2，x^5……）* printf("x^%d + ",p->expn);  
 }

else if(p->coef == 1 && p->expn != 0){*//系数为-1指数不为0的情况（x^2，x^5……）* printf("-x^%d + ",p->expn);  
 }  
 else if(p->coef == 0){*//两个多项式相加的和为0时的情况，不输出* printf("");  
 }  
 else if(p->coef != 1 && p->expn != 0 && p->coef > 1){*//系数不为1指数不为0的情况（2x^3,3x^2……）* printf("%dx^%d + ",p->coef,p->expn);  
 }

else if(p->coef < 0 && p->expn != 0){*//系数不为1指数不为0的情况（2x^3,3x^2……）* printf("-%dx^%d + ",p->coef,p->expn);  
 }  
 p = p->next;*//让p等于它的后继结点，继续循环* }  
 *//判断最后一个多项式的类型 （ps：因为上述循环当中我们定义p->next!=NULL时执  
 // 行，那么当p为最后一个结点时，p->next已经是空了，那么这个时候就会跳出循环，  
 // 最后一个结点的数据就会不运行出来，所以我们需要另写，如果不想写这么多代码的  
 // 话让它在循环内输出也很简单，大家可以动脑筋想一想），下列语句同上* if(p->expn == 0 && p->coef != 0){  
 printf("%d",p->coef);  
 }  
 else if(p->coef == 1 && p->expn == 1){*//系数为1指数为1的情况* printf("x +");  
 }  
 else if(p->expn == 1 && p->coef != 0){  
 printf("%d x",p->coef);  
 }  
 else if(p->coef == 1 && p->expn != 0){  
 printf(" x^%d",p->expn);  
 }  
 else if(p->coef == 0){  
 printf("0"); *//这条语句和上面不一样，因为这是最后一个多项式的输出，如果最后一个多项式的计算结果为0，那么不这样写就会输出（……+）有一个加号，不美观* }  
 else if(p->coef != 1 && p->expn != 0){  
 printf("%dx^%d",p->coef,p->expn);  
 }  
}