数据结构实践课报告

题目：一元稀疏多项式计算器

班级：17052317 姓名：於文卓 学号17061833 完成日期：2018年11月5日

1. **需求分析**

1．输入并建立多项式；

2．输出多项式，输出形式为整数序列：*n*，*c1*，*e*1，*c*2，*e*2，……，*cn*，*en*，其中n是多项式的项数，*c*i和*e*i分别是第*i*项的系数和指数，序列按指数降序排列；

3．多项式a和b相加，建立多项式a＋b；

4．多项式a和b相减，建立多项式a - b；

5. 测试数据：

（1） (2x+5x^8-3.1x^11)+(7-5x^8+11x^9)=(-3.1x^11+11x^9+2x+7)

（2） (6x^-3-x+4.4x^2-1.2x^9+1.2x^9)-(-6x^-3+5.4x^2-x^2+7.8x^15 )

=(-7.8x^15-1.2x^9+12x^-3-x);

（3）(1+x+x^2+x^3+x^4+x^5)+(-x^3-x^4)=(1+x+x^2+x^5);

（4）(x+x^3)+(-x-x^3)=0

（5）(x+x^100)+(x^100+x^200)=(x+2x^100+x^200)

（6）(x+x^2+x^3)+0=x+x^2+x^3

（7）互换上述测试数据中的前后两个多项式

1. **概要设计**
2. **链表的抽象数据类型定义为：**

ADT Node{

数据对象：D＝{ ai | ai∈ElemSet, i=1,2,...,n, n≥0 }

数据关系：R1＝{ <ai-1, ai>|ai-1, ai∈D, i=2,...,n }

基本操作：

InitList(&L)

操作结果：构造一个空的线性表L。

DestroyList(&L)

初始条件：线性表L已存在。

操作结果：销毁线性表L。

ClearList(\*L)

初始条件：线性表L已存在。

操作结果：将线性表L重置为空表。

LocateElem(L, e, cmp())

初始条件：线性表L已存在，compare()是元素判定函数。

操作结果：返回L中第1个与e满足关系cmp()的元素的位序。 若这样的元素不存在，则返回值为0。

SetCurElem(&p, e)

初始条件：线性表L已存在，且非空。

操作结果：用元数e更新p所指结点中元数的值。

GetCurElem(p)

初始条件：线性表L已存在，且非空。

操作结果：返回p所指结点中数据元数的值。

InsFirst (&L, h, s)

初始条件：线性表L已存在，h结点在L中。

操作结果：在L的s所指结点插入在h结点之后，L的长度加1。

DelFirst (&L, h, q)

初始条件：线性表L已存在且非空，q结点在L中且不是尾结点

操作结果：删除链表L中的h结点之后的结点q，L的长度减1。

MakeNode(&p, e)

操作结果：创建了一个结点p，其data部分为e。

FreeNode(&p)

初始条件：结点p存在且非空。

操作结果：释放结点p空间。

Append(LinkList &L,Link s)

初始条件：线性表L已存在。

操作结果：s及s以后的结点链接到了原L之后，L的长度增加链上的结点数。

ListEmpty(L)

初始条件：线性表L已存在。

操作结果：若线性链表L为空表，则返回TRUE,否则返回FALSE。

GetHead(L)

初始条件：线性表L已存在。

操作结果：返回线性链表L中头结点的位置。

NextPos(L, p)

初始条件：线性表L已存在。

操作结果：返回p所指结点的直接后继的位置，若没后继，则返回NULL。

int cmp(a, b)

初始条件：存在两个元数。

操作结果：比较a，b的数值，分别返回-1，0，1。

} ADT LinkList

**2.一元多项式的抽象数据类型定义为：**

ADT Polynomial{

数据对象：D＝{ ai | ai∈TermSet, i=1,2,...,m, m≥0 }

数据关系：R1＝{ <ai-1, ai>|ai-1, ai∈D, 且ai-1中的指数值<ai中的指数值，i=2，……，n}

基本操作：

Creat（&P，m）

操作结果：输入m项的系数和指数，建立一元多项式P。

Add（&Pa，&Pb）

初始条件：一元多项式Pa和Pb已存在。

操作结果：完成多项式相加运算，即：Pa=Pa+Pb，并销毁一元多项式Pb。

Sub (&Pa,&Pb)

初始条件：一元多项式Pa和Pb已存在。

操作结果：完成多项式相减运算，即：Pa=Pa-Pb，并销毁一元多项式Pb。

Value\_x(Pa , x)

初始条件：一元多项式Pa已存在。

操作结果：求多项式在x处的值。

Print (Node\*p)

初始条件：一元多项式p已存在，且已知多项式项数。

操作结果：打印输出一元多项式p的项数、系数和指数。

Advance\_print(p, m)

初始条件：一元多项式p已存在，且已知多项式项数。

操作结果：打印输出一元多项式的类数学表达式。

Sort (&p)

初始条件：一元多项式p已存在。

操作结果：对多项式p进行排序

Repeat(&p)

操作结果：对多项式p自身进行化简

}ADT Polynomial

1. **本程序包含4个模块：**

(1)主程序模块：

int main(){

初始化；

接受命令；

while（命令！=退出）{

处理命令；

接受命令；

}

return 0;

}

(2)一元多项式单元模块——实现一元多项式的抽象数据类型；

（3）链表单元模块——实现链表的抽象数据类型；

(4)结点结构单元模块——定义链表的节点结构。

各模块之间的调用关系如下：

主程序模块 一元多项式单元模块 链表单元模块

结点结构单元模块

1. **实验程序**

**/\***

**\* @Author: 於文卓**

**\* @Date: 2018-11-02 09:54:26**

**\* @Last Modified by: 於文卓**

**\* @Last Modified time: 2018-11-05 23:14:52**

**\*/**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<math.h>**

**typedef struct Node**

**{**

**double xishu;**

**int zhishu;**

**struct Node \*next;**

**} Node;**

**int n;**

**void print(Node \*Head);**

**Node \*create()//返回的是一个指针，这个指针是指向结构体的**

**{**

**Node \*Head;//创建一个头结点**

**Head = (Node \*)malloc(sizeof(Node));**

**Head -> xishu = 0;//给头结点赋值初始化**

**Head -> zhishu = 0;**

**Head -> next = NULL;**

**Node \*p, \*q; //p用来存新的结点，q用来保存当前结点**

**q = Head;**

**printf("请先输入多项式的项数n\n" );**

**scanf("%d", &n);**

**if (n == 0)**

**{**

**p = (Node \*)malloc(sizeof(Node));**

**p->xishu = 0;**

**p->zhishu = 0;**

**p->next = NULL;**

**q->next = p;**

**return Head;**

**}**

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

**{**

**p = (Node \*)malloc(sizeof(Node));**

**printf("请输入第%d项的系数和指数，用空格隔开\n", i);**

**scanf("%lf %d", &(p->xishu), &(p->zhishu));**

**p->next = NULL;**

**q->next = p;//当前节点和新创建的节点连起来**

**q = p;//更新一下当前的节点**

**}**

**return Head;//返回头结点的指针**

**}**

**Node \*sort(Node \*Head) //冒泡排序**

**{**

**if (Head->next == NULL)**

**{**

**return Head;**

**}**

**Node \*pre, \*cur, \*next, \*head, \*temp, \*end;**

**head = Head;**

**end = NULL;**

**while(head->next != end)**

**{**

**for (pre = head, cur = pre->next, next = cur->next; next != end; pre = pre->next, cur = cur->next, next = next->next)**

**{**

**if (cur->zhishu < next->zhishu)//小的往后沉 节点做交换 指向节点的指针也交换**

**{**

**cur->next = next->next;**

**pre->next = next;//这里犯错了 本来是 pre->next = cur->next;**

**next->next = cur;**

**temp = next;**

**next = cur;**

**cur = temp;**

**}**

**}**

**end = cur;**

**}**

**return head;**

**}**

**Node \*add(Node \*A, Node \*B) //设计了一个temp 一个mark 假如从A->B 那么mark先到B上 temp->next = B; temp = mark;通过temp和mark，把两组多项式从大到小串起来。**

**//mark 追着 a 而temp追着mark;**

**{**

**Node \*P = sort(A);**

**Node \*Q = sort(B);**

**Node \*res = P;**

**Node \*temp = P;**

**Node \*mark;**

**Node \*a, \*b;**

**a = P->next;**

**b = Q->next;**

**while(a != NULL && b != NULL)**

**{**

**if (a->zhishu < b->zhishu)**

**{**

**mark = b;//因为最后是指数降序，给指数大的做标记，指针后移**

**b = b->next;**

**}**

**else if (a->zhishu == b->zhishu)//如果相等，系数相加**

**{**

**a->xishu = (a->xishu) + (b->xishu);**

**mark = a;**

**a = a->next;//两个都后移**

**b = b->next;**

**}**

**else**

**{**

**mark = a;**

**a = a->next;**

**}**

**if (mark->xishu != 0)//做连接，把系数不是零的都连起来**

**{**

**temp->next = mark;**

**temp = mark;**

**}**

**}**

**if (a == NULL) //若果a已经到头了 那么a就接到b上**

**{**

**a = b;**

**}**

**while(a)**

**{**

**mark = a;**

**a = a->next;**

**temp->next = mark;**

**temp = mark;**

**}**

**return res;**

**}**

**int count(Node \*a)//输出的n，**

**{**

**Node \*b = a;**

**int cnt = 0;**

**while(b != NULL)**

**{**

**if (b->xishu != 0)//系数不能是0**

**{**

**cnt++;**

**}**

**b = b->next;**

**}**

**return cnt;**

**}**

**Node \*repeat(Node \*A)//先把多项式里指数相同的项合并**

**{**

**Node \*p, \*head;**

**head = A;**

**p = head->next;**

**Node \*q;**

**q = p->next;**

**while(p->next != NULL)**

**{**

**while(q != NULL)**

**{**

**if (p->zhishu == q->zhishu)**

**{**

**p->xishu += q->xishu;**

**p->next = q->next;**

**free(q);**

**}**

**q = q->next;**

**}**

**p = p->next;**

**q = p->next;**

**}**

**return head;**

**}**

**Node \*sub(Node \*A, Node \*B)//减法 基本与加法同理**

**{**

**Node \*P = sort(A);**

**Node \*Q = sort(B);**

**Node \*res = P;**

**Node \*temp = P;**

**Node \*mark;**

**Node \*a, \*b;**

**a = P->next;**

**b = Q->next;**

**while(a != NULL && b != NULL)**

**{**

**if (a->zhishu < b->zhishu)**

**{**

**mark = b;**

**mark->xishu = (-1) \* (mark->xishu);//0减去一个数的时候要注意下**

**b = b->next;**

**}**

**else if (a->zhishu == b->zhishu)**

**{**

**a->xishu = a->xishu - b->xishu;**

**mark = a;**

**a = a->next;**

**b = b->next;**

**}**

**else**

**{**

**mark = a;**

**a = a->next;**

**}**

**if (mark->xishu != 0)**

**{**

**temp->next = mark;**

**temp = mark;**

**}**

**}**

**if (a == NULL) //若果a已经到头了 那么a就接到b上**

**{**

**a = b;**

**Node \*temp = a;**

**while(temp != NULL)**

**{**

**temp->xishu \*= -1;**

**temp = temp->next;**

**}**

**}**

**while(a)**

**{**

**mark = a;**

**a = a->next;**

**temp->next = mark;**

**temp = mark;**

**}**

**return res;**

**}**

**void print(Node \*Head)**

**{**

**int cnt = count(Head);**

**printf("%d,", cnt);**

**Node \*p = sort(Head);//先排下序**

**p = Head->next;**

**if (p == NULL)**

**{**

**printf("0");**

**return;**

**}**

**else if (p->zhishu == 0)//如果第一个数的指数为零，直接输出系数**

**{**

**printf("%.2lf,", p->xishu);**

**}**

**else if(p->xishu == 0)//如果第一个数的系数为零，指针向后移**

**{**

**p = p->next;**

**}**

**else//除此之外，正常输出即可**

**{**

**printf("%.2lf,%d,", p->xishu, p->zhishu);**

**p = p->next;**

**}**

**while(p != NULL)**

**{**

**if (p->xishu == 0 && p->next == NULL)//如果整个多项式结果为零，直接返回。**

**{**

**printf("\n");**

**return;**

**}**

**printf("%.2lf,%d,", p->xishu, p->zhishu);**

**p = p->next;**

**}**

**printf("\n");**

**}**

**void value\_x(Node \*Head)//计算多项式在x处的值**

**{**

**Node \*p = Head->next;**

**printf("x为多少呢\n");**

**double x;**

**double sum = 0;**

**scanf("%lf", &x);**

**while(p != NULL)**

**{**

**sum += (p->xishu) \* (pow(x, p->zhishu));**

**p = p->next;**

**}**

**printf("%lf\n", sum);**

**}**

**void advance\_print(Node \*Head)//选做题，输出语义化的结果**

**{**

**Node \*p = sort(Head);//先排下序**

**p = Head->next;**

**if (p == NULL)**

**{**

**printf("0");**

**return;**

**}**

**else if (p->zhishu == 0)//如果第一个数的指数为零，直接输出系数**

**{**

**printf("%.2lf,", p->xishu);**

**}**

**else if(p->xishu == 0)//如果第一个数的系数为零，指针向后移**

**{**

**p = p->next;**

**}**

**else**

**{**

**if (p->xishu == 1)**

**{**

**printf("x^%d", p->zhishu);**

**}**

**else if (p->xishu == -1)**

**{**

**printf("-x^%d", p->zhishu);**

**}**

**else**

**{**

**printf("%.2lf\*x^%d", p->xishu, p->zhishu);**

**}**

**p = p->next;**

**}**

**while(p != NULL)**

**{**

**if (p->xishu == 0 && p->next == NULL)//如果最后一个数是零**

**{**

**printf("\n");**

**return;**

**}**

**if (p->xishu > 0)**

**{**

**printf("+");**

**}**

**if (p->xishu == 1 && p->zhishu != 0 )**

**{**

**printf("x^%d", p->zhishu);**

**}**

**else if (p->xishu == -1)**

**{**

**printf("-x^%d", p->zhishu);**

**}**

**else if (p->zhishu == 0)**

**{**

**printf("%.2lf\n", p->xishu);**

**}**

**else**

**{**

**printf("%.2lf\*x^%d", p->xishu, p->zhishu);**

**}**

**p = p->next;**

**}**

**printf("\n");**

**}**

**int main()**

**{**

**printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf(" \* 多项式操作程序 \*\n");**

**printf(" \* \*\n");**

**printf(" \* 1:输入多项式 2:输出多项式(标准版) \*\n");**

**printf(" \* \*\n");**

**printf(" \* 3:代入x的值计算a 4:代入x的值计算b \*\n");**

**printf(" \* \*\n");**

**printf(" \* 5:输出a+b 6:输出a-b \*\n");**

**printf(" \* \*\n");**

**printf(" \* 7.加强版输出 8:退出程序 \*\n");**

**printf(" \* \*\n");**

**printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**while(1)**

**{**

**printf("你的选择是\n");**

**Node \*A, \*B, \*res;**

**Node \*addA, \*addB;**

**Node \*subA, \*subB;**

**int choice;**

**scanf("%d", &choice);**

**switch(choice)**

**{**

**case 1: printf("您的选择是1--->输入多项式\n"); printf("请根据提示输入链表A的数据\n"); A = create(); addA = A, subA = A;**

**printf("请根据提示输入链表B的数据\n"); B = create(); addB = B, subB = B; break;**

**case 2: printf("您的选择是2--->标准输出多项式a和b\n"); printf("A\n"); print(A); printf("B\n"); print(B); break;**

**case 3: printf("您的选择是3--->代入x的值计算a\n"); value\_x(A); break;**

**case 4: printf("您的选择是4--->代入x的值计算b\n"); value\_x(B); break;**

**case 5: printf("您的选择是5--->输出a+b\n"); addA = repeat(A); addB = repeat(B); res = add(addA, addB); advance\_print(res); printf("欢迎下次使用\n"); exit(0);**

**case 6: printf("您的选择是6--->输出a-b\n"); subA = repeat(A); subB = repeat(B); res = sub(subA, subB); advance\_print(res); printf("欢迎下次使用\n"); exit(0);**

**case 7: printf("您的选择是7--->输出表达式\n"); printf("A\n"); advance\_print(A); printf("B\n"); advance\_print(B); break;**

**case 8: printf("您的选择是8--->退出\n"); printf("bye~"); exit(0);**

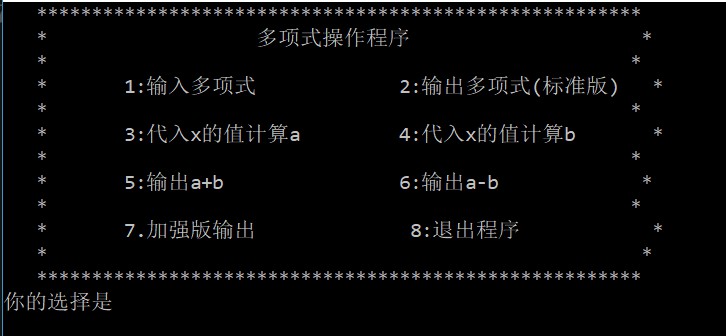
**default: printf("非法输入，请重新选择\n"); break;**

**}**

**}**

**}**

1. **调试分析**
2. **做减法的算法时，要考虑的情况比较多，最容易忽略的是0-A时要变符号**
3. **输入的多项式可能本身就是有相同的项，我这里用了repeat函数，预先把指数相同的项合并了，化简了多项式**
4. **按题目要求，输出需要指数降序，用冒泡排序法实现了sort函数，在输出前先对多项式进行排序**
5. **输出时，也要考虑很多情况，比如第一个数的指数为零，就应该直接输出系数，第一个数的系数为零，那么跳过这个节点，在做语义化输出的时候（加强版输出），不仅要考虑标准输出的情况，还要考虑系数为-1，0，1指数为0的情况，不能遗漏**
6. **用户手册**
7. **本程序运行环境为win10操作系统，执行文件为一元稀疏多项式.exe**
8. **用户界面如下**

****

选择1，输入多项式后，即可选择相应功能。

1. **测试数据**

