

杭州电子科技大学

HANGZHOU DIANZI UNIVERSITY

数据结构课程设计报告

(2021-2022-2 学期)



题目	一元稀疏多项式计算器			
学院	理学院			
专业	信息与计算科学			
组号	第十六组	学号	姓名	分工
		20071226	童繁	流程图
		20071227	王瀚功	测试
		20071228	王赛豪	文案
		20071229	吴政豪	调试
		20071230	武琦	代码
时间	2022 年 4 月 25 日			

目录

1. 需求分析.....	1
2. 概要设计.....	1
3. 详细设计.....	2
3.1 创建链表.....	3
3.1.1 输入多项式.....	3
3.1.2 多项式整理.....	4
3.2 输出多项式.....	5
3.3 加法.....	7
3.4 减法.....	9
3.4.1 多项式系数变负.....	9
3.4.2 减法.....	10
3.5 乘法.....	10
3.6 赋值运算.....	12
3.7 求一阶导.....	13
4. 测试案例.....	14
5. 功能测试.....	17
6. 附录	20

1. 需求分析

- (1) 输入并建立多项式;
- (2) 输出多项式, 输出形式为类数学表达式, 序列按指数降序排列;
- (3) 多项式 A 和 B 相加, 建立多项式 $A+B$, 输出相加的多项式;
- (4) 多项式 A 和 B 相减, 建立多项式 $A-B$, 输出相减的多项式;
- (5) 多项式 A 和 B 相乘, 建立多项式 $A \times B$, 输出相乘的多项式;
- (6) 计算多项式在 x 处的值;
- (7) 求多项式 A 的导函数 A' ;
- (8) 设计一个菜单, 具有上述操作要求的基本功能。

2. 概要设计

要设计一个一元稀疏多项式简单计算器, 可以用两个带表头节点的单链表 A、B 分别存储两个多项式, 进行初步整理——排序和同指数项相加。下列是具体运算算法:

1. 做加法时, 将 A 链和 B 链的所有结点都添加到新链 C 中;
2. 做减法时, 将 B 链的所有结点的系数变为相反数后与 A 链进行加法运算, 再将 B 链变回原链表以便重复使用;
3. 做乘法时, 遍历 A 链的所有结点, 遍历 B 链的所有结点, 也就是双重循环, 将 $A \times B$ 的所有结点添加到新链 C 中;
4. 求导时, 将求导后的所有结点添加到新链 C 中。

进行以上运算时, 首先建立一个新的链表 C, 实行相应的算法, 在运算结束后输出的新链表 C 即为最终结果。而将 x 代入多项式求值时, 只需将 x 代入每项求值, 再求和后输出最终结果。

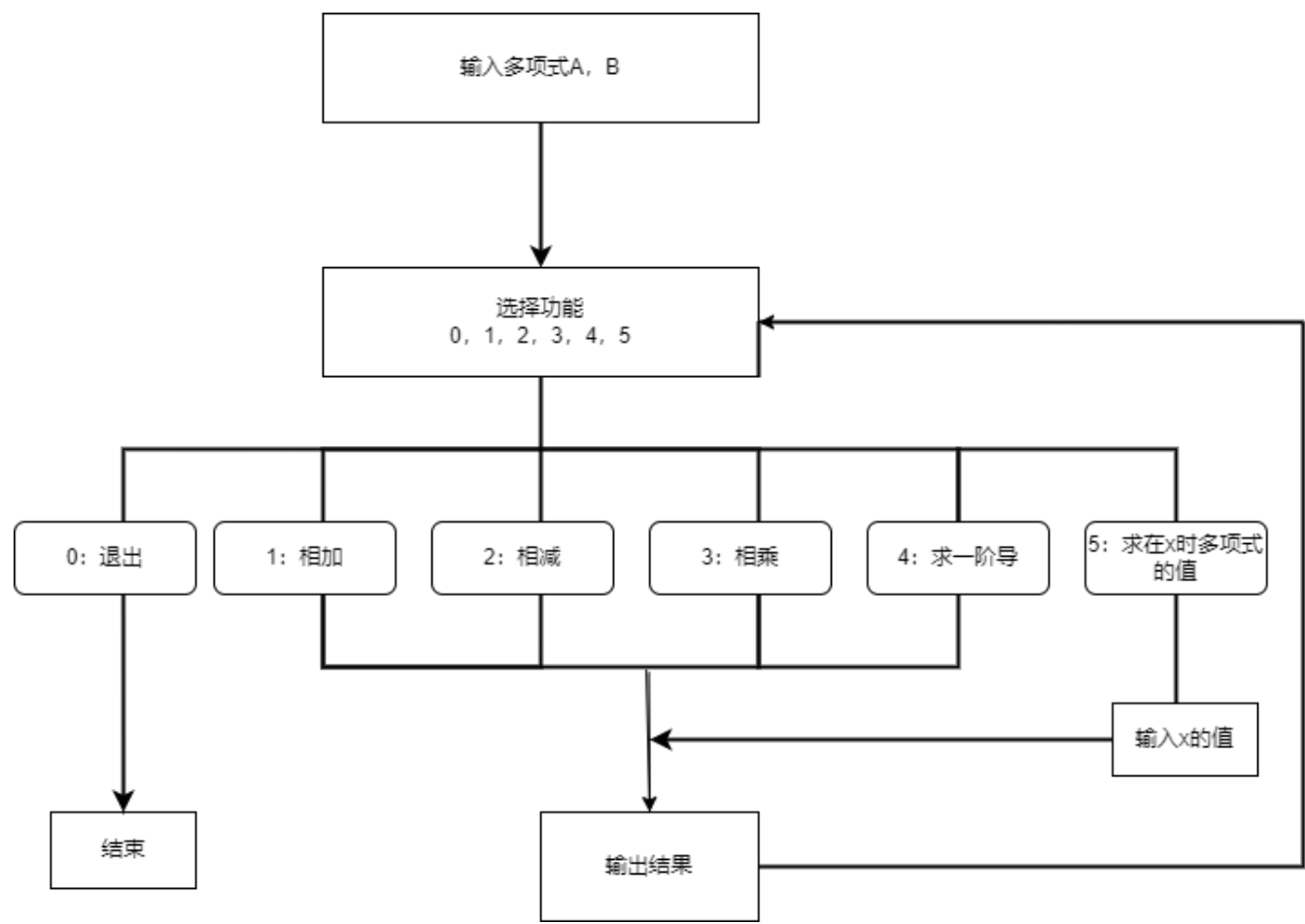


图 1: 主函数流程图

3. 详细设计

要解决上述问题，我们设计了 9 个函数，流程图和函数代码如下：

3.1 创建链表

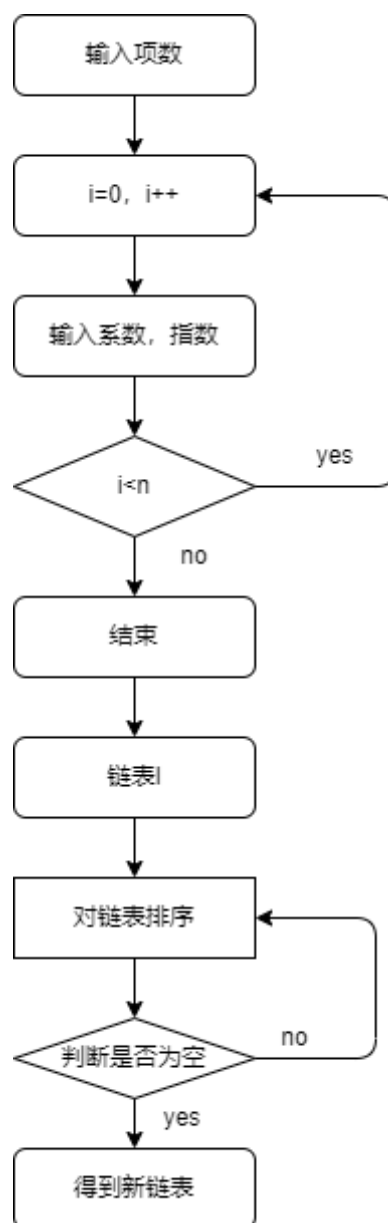


图 2: 链表创建流程图

3.1.1 输入多项式

Listing 1: input

```

1 list input() //函数: 输入多项式
2 {
3     list a,b,l;
4     l=(list)malloc(sizeof(node));
5     l->next=NULL;
6     b=l;

```

```
7   int n,i;
8   printf("请输入项数: ");
9   scanf("%d",&n);//输入项数
10  printf("请输入系数 指数: \n");
11  for(i=0;i<n;i++)
12  {
13      a=(list)malloc(sizeof(node));
14      printf("第%d项:",i+1);
15      scanf("%lf %d",&(a->coe),&(a->exp));//输入系数和指数
16      a->next=NULL;
17      b->next=a;
18      b=a;
19  }
20  return l;
21 }
```

3.1.2 多项式整理

Listing 2: sort

```
1 void sort(list l)//多项式整理
2 {
3     list i,j;
4     double tcoe;
5     int texp;
6     for(i=l->next;i!=NULL;i=i->next)
7     {
8         for(j=i->next;j!=NULL;j=j->next)
9         {
10             if(j->exp>i->exp)
11             {
12                 tcoe=j->coe;
13                 j->coe=i->coe;
14                 i->coe=tcoe;
15                 texp=j->exp;
16                 j->exp=i->exp;
17                 i->exp=texp;
18             }
19             else if(j->exp==i->exp)
20             {
21                 tcoe=j->coe+i->coe;
22                 i->coe=tcoe;
23                 j->coe=0;
24                 i->exp=j->exp;
25                 j->exp=0;
```

```

26     }
27     }
28     }
29 }

```

3.2 输出多项式

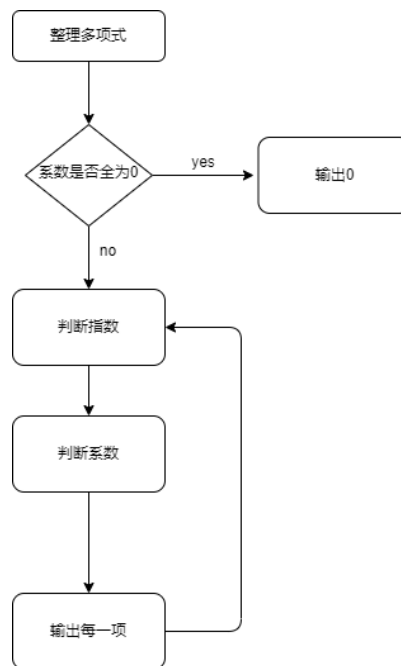


图 3: 多项式输出流程图

Listing 3: output

```

1 void output(list l)//函数: 输出多项式
2 {
3     sort(l);
4     list a,b;
5     int i=0,j=0,k=0;
6     a=l->next;
7     b=l->next;
8     while(b)
9     {
10        j++;
11        if(b->coe==0.0)
12            k++;
13        b=b->next;
14    }
15    if(j==k)
16        printf("0");

```

```
17     else
18     {
19         while(a)
20         {
21             if(a->exp!=0)
22             {
23                 if(a->coe<0)
24                 {
25                     printf("-");
26                     if(a->exp==1)
27                     {
28                         if(a->coe==-1)
29                             printf("x");
30                         else
31                             printf("%.1fx",m,-a->coe);
32                     }
33                     else
34                     {
35                         if(a->coe==-1)
36                             printf("x^%d",a->exp);
37                         else
38                             printf("%.1fx^%d",m,-a->coe,a->exp);
39                     }
40                 }
41                 else if(a->coe>0)
42                 {
43                     if(i!=0)
44                         printf("+");
45                     if(a->exp==1)
46                     {
47                         if(a->coe==1)
48                             printf("x");
49                         else
50                             printf("%.1fx",m,a->coe);
51                     }
52                     else if(a->exp!=1)
53                     {
54                         if(a->coe==1)
55                             printf("x^%d",a->exp);
56                         else
57                             printf("%.1fx^%d",m,a->coe,a->exp);
58                     }
59                 }
60             }
61         }
62     }
63     else
64     {
65         if(a->coe<0)
```



```

64     printf("%.1f",m,a->coe);
65     else if(a->coe>0)
66     {
67         if(i==0)
68             printf("%.1f",m,a->coe);
69         else
70             printf("+%.1f",m,a->coe);
71     }
72 }
73 a=a->next;
74 i++;
75 }
76 }
77 printf("\n");
78 }

```

3.3 加法

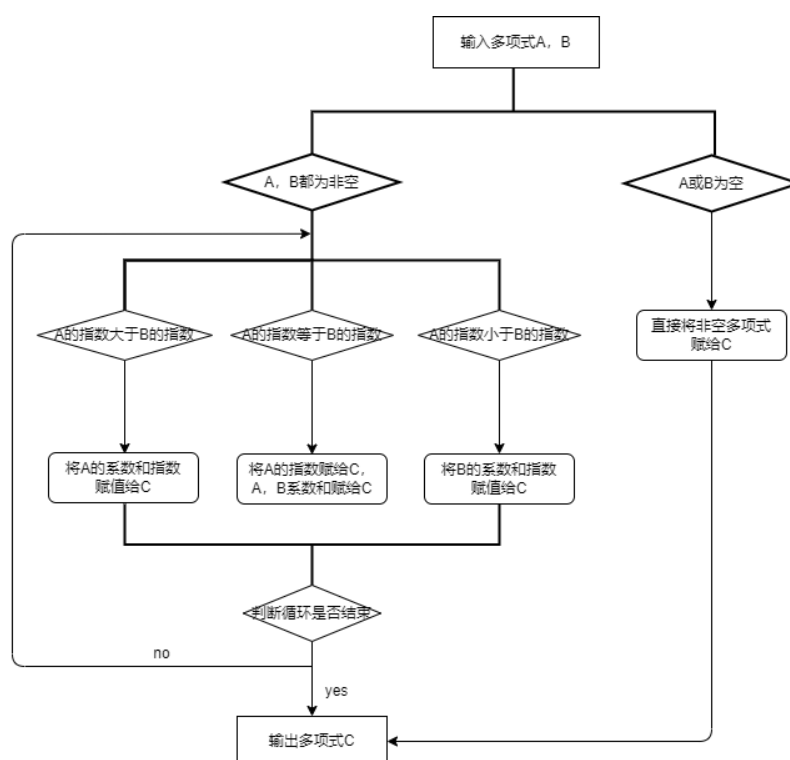


图 4: 加法流程图

Listing 4: plus

```

1 void plus(list a,list b)//函数: 多项式相加
2 {
3     list la,lb,lc,ld,c,d;

```

```
4   c=(list)malloc(sizeof(node));
5   c->next=NULL;
6   d=c;
7   la=a->next;
8   lb=b->next;
9   while(la!=NULL&&lb!=NULL)
10  {
11      lc=(list)malloc(sizeof(node));
12      lc->next=NULL;
13      if(la->exp>lb->exp)
14      {
15          lc->exp=la->exp;
16          lc->coe=la->coe;
17          d->next=lc;
18          d=lc;
19          la=la->next;
20      }
21      else if(la->exp==lb->exp)
22      {
23          double sum=la->coe+lb->coe;
24          if(sum!=0.0)
25          {
26              lc->exp=la->exp;
27              lc->coe=sum;
28              d->next=lc;
29              d=lc;
30          }
31          la=la->next;
32          lb=lb->next;
33      }
34      else
35      {
36          lc->exp=lb->exp;
37          lc->coe=lb->coe;
38          d->next=lc;
39          d=lc;
40          lb=lb->next;
41      }
42  }
43  while(la)
44  {
45      lc=(list)malloc(sizeof(node));
46      lc->next=NULL;
47      lc->exp=la->exp;
48      lc->coe=la->coe;
49      la=la->next;
50      d->next=lc;
```

```

51     d=lc;
52 }
53 while(lb)
54 {
55     lc=(list)malloc(sizeof(node));
56     lc->next=NULL;
57     lc->exp=lb->exp;
58     lc->coe=lb->coe;
59     lb=lb->next;
60     d->next=lc;
61     d=lc;
62 }
63 output(c);
64 }

```

3.4 减法

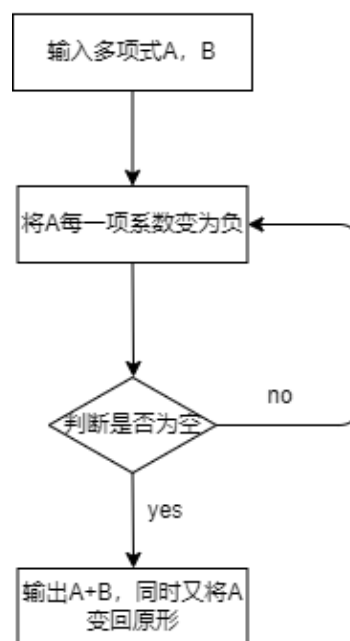


图 5: 减法流程图

3.4.1 多项式系数变负

Listing 5: oppo

```

1 void oppo(list *l)//函数：多项式系数变负
2 {
3     list a=(*l)->next;
4     while(a)

```

```
5  {  
6    a->coe=-a->coe;  
7    a=a->next;  
8  }  
9 }
```

3.4.2 减法

Listing 6: minus

```
1 void minus(list a,list b)//函数：多项式相减  
2 {  
3   oppo(&b);  
4   plus(a,b);  
5   oppo(&b);  
6 }
```

3.5 乘法

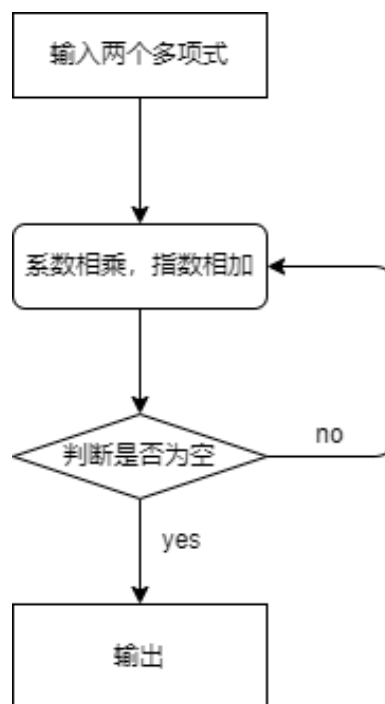


图 6: 乘法流程图

Listing 7: multi

```
1 void multi(list a,list b)//函数：多项式乘法  
2 {
```

```
3  list la,lb,lc,c,ld;
4  la=a->next;
5  lb=b->next;
6  c=(list)malloc(sizeof(node));
7  lc=c;
8  lc->next=NULL;
9  while(la)
10 {
11     lb=b->next;
12     while(lb)
13     {
14         ld=(list)malloc(sizeof(node));
15         ld->next=NULL;
16         ld->coe=la->coe*lb->coe;
17         ld->exp=la->exp+lb->exp;
18         lc->next=ld;
19         lc=ld;
20         lb=lb->next;
21     }
22     la=la->next;
23 }
24 output(c);
25 }
```

3.6 赋值运算

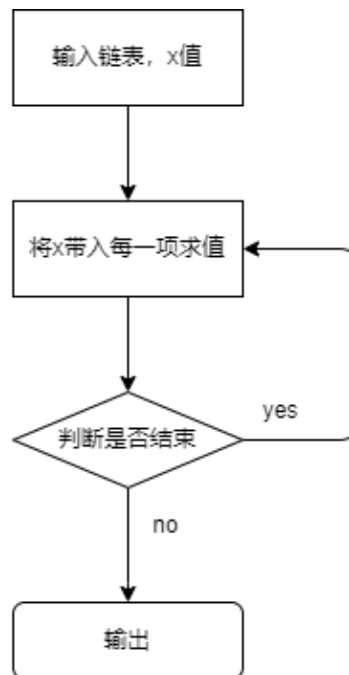


图 7: 赋值运算流程图

Listing 8: fx

```
1 void fx(list a,double x)//函数: 多项式在x处的值
2 {
3     list la;
4     la=a->next;
5     double value=0;
6     while(la)
7     {
8         value+=(la->coe)*pow(x,la->exp);
9         la=la->next;
10    }
11    printf("在%.1f处的值为:%.1f\n",m,x,m,value);
12 }
```

3.7 求一阶导

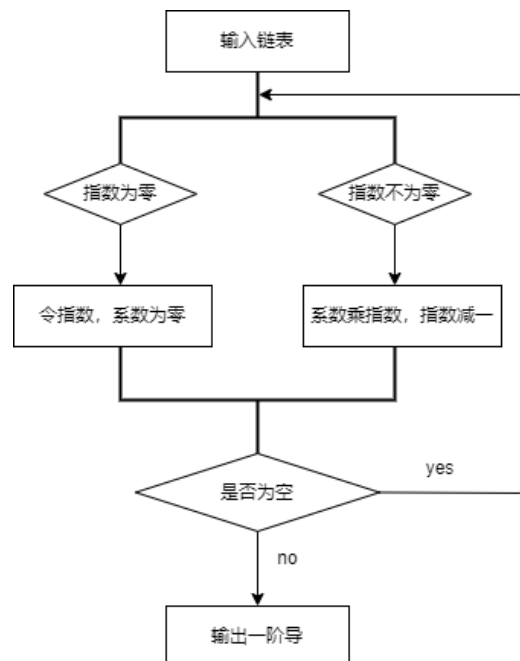


图 8: 求导流程图

Listing 9: diff

```

1 void diff(list a)//函数: 求一阶导
2 {
3     list la,b,lb,lc;
4     la=a->next;
5     b=(list)malloc(sizeof(node));
6     lb=b;
7     lb->next=NULL;
8     while(la)
9     {
10         lc=(list)malloc(sizeof(node));
11         if(la->exp==0)
12         {
13             lc->coe=0.0;
14             lc->exp=0;
15         }
16         else
17         {
18             lc->coe=la->coe*la->exp;
19             lc->exp=la->exp-1;
20         }
21         lc->next=NULL;
22         lb->next=lc;

```

```

23     lb=lc;
24     la=la->next;
25 }
26 printf("的一阶导为:");
27 output(b);
28 }

```

4. 测试案例

例 1: $(2x + 5x^8 - 3.1x^{11}) + (7 - 5x^8 + 11x^9) = -3.1x^{11} + 11x^9 + 2x + 7$

```

请输入全部数保留小数的位数: 1
多项式A:
请输入项数: 3
请输入系数 指数:
第1项: 2 1
第2项: 5 8
第3项: -3.1 11
多项式B:
请输入项数: 3
请输入系数 指数:
第1项: 7 0
第2项: -5 8
第3项: 11 9
多项式A: -3.1x^11+5.0x^8+2.0x
多项式B: 11.0x^9-5.0x^8+7.0
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
1
多项式相加结果为: -3.1x^11+11.0x^9+2.0x+7.0

```

图 9: 测试 1

例 2: $(6x^{-3} - x + 4.4x^2 - 1.2x^9) - (-6x^{-3} + 5.4x^2 - x^2 + 7.8x^{15}) = -7.8x^{15} - 1.2x^9 - x + 12x^{-3}$


```

请输入全部数保留小数的位数: 1
多项式A:
请输入项数: 4
请输入系数(不为0) 指数:
第1项:6 -3
第2项:-1 1
第3项:4.4 2
第4项:-1.2 9
多项式B:
请输入项数: 4
请输入系数(不为0) 指数:
第1项:-6 -3
第2项:5.4 2
第3项:-1 2
第4项:7.8 15
多项式A:-1.2x^9+4.4x^2-x+6.0x^-3
多项式B:7.8x^15+4.4x^2-6.0x^-3
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
2
多项式相减结果为: -7.8x^15-1.2x^9-x+12.0x^-3

```

图 10: 测试 2

$$\text{例 3:}(1+x+x^2+x^3+x^4+x^5)+(-x^3-x^4)=x^5+x^2+x+1$$

```

请输入全部数保留小数的位数: 0
多项式A:
请输入项数: 6
请输入系数 指数:
第1项:1 0
第2项:1 1
第3项:1 2
第4项:1 3
第5项:1 4
第6项:1 5
多项式B:
请输入项数: 2
请输入系数 指数:
第1项:-1 3
第2项:-1 4
多项式A:x^5+x^4+x^3+x^2+x+1
多项式B:-x^4-x^3
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
1
多项式相加结果为: x^5+x^2+x+1

```

图 11: 测试 3

$$\text{例 4: } (x + x^3) + (-x - x^3) = 0$$

```

请输入全部数保留小数的位数: 0
多项式A:
请输入项数: 2
请输入系数 指数:
第1项: 1 3
第2项: 1 1
多项式B:
请输入项数: 2
请输入系数 指数:
第1项: -1 1
第2项: -1 3
多项式A: x^3+x
多项式B: -x^3-x
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
1
多项式相加结果为: 0

```

图 12: 测试 4

$$\text{例 5: } (x + x^{100}) + (x^{100} + x^{200}) = x^{200} + 2x^{100} + x$$

```

请输入全部数保留小数的位数: 0
多项式A:
请输入项数: 2
请输入系数 指数:
第1项: 1 1
第2项: 1 100
多项式B:
请输入项数: 2
请输入系数 指数:
第1项: 1 100
第2项: 1 200
多项式A: x^100+x
多项式B: x^200+x^100
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
1
多项式相加结果为: x^200+2x^100+x

```

图 13: 测试 5

$$\text{例 6: } (x + x^2 + x^3) + 0 = x^3 + x^2 + x$$

```

请输入全部数保留小数的位数: 0
多项式A:
请输入项数: 3
请输入系数 指数:
第1项:1 1
第2项:1 2
第3项:1 3
多项式B:
请输入项数: 1
请输入系数 指数:
第1项:0
0
多项式A:x^3+x^2+x
多项式B:0
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
1
多项式相加结果为: x^3+x^2+x

```

图 14: 测试 6

5. 功能测试

例一: $A: 3x^4 + 0.002x^3 - 2x^2 + 13x + 1$ $B: 5x^6 + 2x^3 - 0.3x^2 - 12$

```

多项式A:3.000x^4+0.002x^3-2.000x^2+13.000x+1.000
多项式B:5.000x^6+2.000x^3-0.300x^2-12.000
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
1
多项式相加结果为: 5.000x^6+3.000x^4+2.002x^3-2.300x^2+13.000x-11.000
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
2
多项式相减结果为: -5.000x^6+3.000x^4-1.998x^3-1.700x^2+13.000x+13.000
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
3
多项式相乘结果为: 15.000x^10+0.010x^9-10.000x^8+71.000x^7+4.104x^6-4.001x^5-9.400x^4-1.924x^3+23.700x^2-156.000x-12.000
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
4
多项式A的一阶导为:12.000x^3+0.006x^2-4.000x+13.000
多项式B的一阶导为:30.000x^5+6.000x^2-0.600x
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
5
请输入x的值:3
多项式A:在3.000处的值为:265.054
多项式B:在3.000处的值为:3684.300
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
0

```

图 15: 功能测试 1

例二: $A: 2x^3 + x^2 + x - 1$ $B: -2x^4 + x^3 + 2x$

```

多项式A:  $2x^3+x^2+x-1$ 
多项式B:  $-2x^4+x^3+2x$ 
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
1
多项式相加结果为:  $-2x^4+3x^3+x^2+3x-1$ 
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
2
多项式相减结果为:  $2x^4+x^3+x^2-x-1$ 
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
3
多项式相乘结果为:  $-4x^7-x^5+7x^4+x^3+2x^2-2x$ 
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
4
多项式A的一阶导为:  $6x^2+2x+1$ 
多项式B的一阶导为:  $-8x^3+3x^2+2$ 
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
5
请输入x的值:-1
多项式A:在-1处的值为:-3
多项式B:在-1处的值为:-5
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
5
请输入x的值:0
多项式A:在0处的值为:-1
多项式B:在0处的值为:0
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
5
请输入x的值:1
多项式A:在1处的值为:3
多项式B:在1处的值为:1
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值

```

图 16: 功能测试 2

例三: $A: 0.3x^2 - 0.04x + 0.005$ $B: -0.3x^2 + 0.04x + 0.005$

```
多项式A:0.300x^2-0.040x+0.005
多项式B:-0.300x^2+0.040x+0.005
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
1
多项式相加结果为: 0.010
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
2
多项式相减结果为: 0.600x^2-0.080x
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
3
多项式相乘结果为: -0.090x^4+0.024x^3-0.002x^2+0.000
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
4
多项式A的一阶导为:0.600x-0.040
多项式B的一阶导为:-0.600x+0.040
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
5
请输入x的值:0
多项式A:在0.000处的值为:0.005
多项式B:在0.000处的值为:0.005
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
5
请输入x的值:-2
多项式A:在-2.000处的值为:1.285
多项式B:在-2.000处的值为:-1.275
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
5
请输入x的值:2
多项式A:在2.000处的值为:1.125
多项式B:在2.000处的值为:-1.115
选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值
```

图 17: 功能测试 3

6. 附录

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4  static int m;//小数位数的全局变量
5  typedef struct node//定义结点结构体
6  {
7      double coe;//系数
8      int exp;//指数
9      struct node *next;//结构体
10 }node,*list;
11
12 list input() //函数：输入多项式
13 {
14     list a,b,l;
15     l=(list)malloc(sizeof(node));
16     l->next=NULL;
17     b=l;
18     int n,i;
19     printf("请输入项数：");
20     scanf("%d",&n);//输入项数
21     printf("请输入系数 指数：\n");
22     for(i=0;i<n;i++)
23     {
24         a=(list)malloc(sizeof(node));
25         printf("第%d项:",i+1);
26         scanf("%lf %d",&(a->coe),&(a->exp));//输入系数和指数
27         a->next=NULL;
28         b->next=a;
29         b=a;
30     }
31     return l;
32 }
33
34 void sort(list l)//多项式整理
35 {
36     list i,j;
37     double tcoe;
38     int texp;
39     for(i=l->next;i!=NULL;i=i->next)
40     {
41         for(j=i->next;j!=NULL;j=j->next)
42         {
43             if(j->exp>i->exp)
44             {
```

```
45         tcoe=j->coe;
46         j->coe=i->coe;
47         i->coe=tcoe;
48         texp=j->exp;
49         j->exp=i->exp;
50         i->exp=texp;
51     }
52     else if(j->exp==i->exp)
53     {
54         tcoe=j->coe+i->coe;
55         i->coe=tcoe;
56         j->coe=0;
57         i->exp=j->exp;
58         j->exp=0;
59     }
60 }
61 }
62 }
63
64 void output(list l)//函数: 输出多项式
65 {
66     sort(l);
67     list a,b;
68     int i=0,j=0,k=0;
69     a=l->next;
70     b=l->next;
71     while(b)
72     {
73         j++;
74         if(b->coe==0.0)
75             k++;
76         b=b->next;
77     }
78     if(j==k)
79         printf("0");
80     else
81     {
82         while(a)
83         {
84             if(a->exp!=0)
85             {
86                 if(a->coe<0)
87                 {
88                     printf("-");
89                     if(a->exp==1)
90                     {
91                         if(a->coe==-1)
```

```
192         printf("x");
193     else
194         printf("%.1fx",m,-a->coe);
195 }
196 else
197 {
198     if(a->coe==-1)
199         printf("x^%d",a->exp);
200     else
201         printf("%.1fx^%d",m,-a->coe,a->exp);
202 }
203 }
204 else if(a->coe>0)
205 {
206     if(i!=0)
207         printf("+");
208     if(a->exp==1)
209     {
210         if(a->coe==1)
211             printf("x");
212         else
213             printf("%.1fx",m,a->coe);
214     }
215     else if(a->exp!=1)
216     {
217         if(a->coe==1)
218             printf("x^%d",a->exp);
219         else
220             printf("%.1fx^%d",m,a->coe,a->exp);
221     }
222 }
223 }
224 else
225 {
226     if(a->coe<0)
227         printf("%.1f",m,a->coe);
228     else if(a->coe>0)
229     {
230         if(i==0)
231             printf("%.1f",m,a->coe);
232         else
233             printf("+%.1f",m,a->coe);
234     }
235 }
236 a=a->next;
237 i++;
238 }
```



```
139     }
140     printf("\n");
141 }
142
143 void plus(list a,list b)//函数: 多项式相加
144 {
145     list la,lb,lc,ld,c,d;
146     c=(list)malloc(sizeof(node));
147     c->next=NULL;
148     d=c;
149     la=a->next;
150     lb=b->next;
151     while(la!=NULL&&lb!=NULL)
152     {
153         lc=(list)malloc(sizeof(node));
154         lc->next=NULL;
155         if(la->exp>lb->exp)
156         {
157             lc->exp=la->exp;
158             lc->coe=la->coe;
159             d->next=lc;
160             d=lc;
161             la=la->next;
162         }
163         else if(la->exp==lb->exp)
164         {
165             double sum=la->coe+lb->coe;
166             if(sum!=0.0)
167             {
168                 lc->exp=la->exp;
169                 lc->coe=sum;
170                 d->next=lc;
171                 d=lc;
172             }
173             la=la->next;
174             lb=lb->next;
175         }
176         else
177         {
178             lc->exp=lb->exp;
179             lc->coe=lb->coe;
180             d->next=lc;
181             d=lc;
182             lb=lb->next;
183         }
184     }
185     while(la)
```

```

186     {
187         lc=(list)malloc(sizeof(node));
188         lc->next=NULL;
189         lc->exp=la->exp;
190         lc->coe=la->coe;
191         la=la->next;
192         d->next=lc;
193         d=lc;
194     }
195     while(lb)
196     {
197         lc=(list)malloc(sizeof(node));
198         lc->next=NULL;
199         lc->exp=lb->exp;
200         lc->coe=lb->coe;
201         lb=lb->next;
202         d->next=lc;
203         d=lc;
204     }
205     output(c);
206 }
207
208 void oppo(list *l)//函数：多项式系数变负
209 {
210     list a=(*l)->next;
211     while(a)
212     {
213         a->coe=-a->coe;
214         a=a->next;
215     }
216 }
217
218 void minus(list a,list b)//函数：多项式相减
219 {
220     oppo(&b);
221     plus(a,b);
222     oppo(&b);
223 }
224
225 void multi(list a,list b)//函数：多项式乘法
226 {
227     list la,lb,lc,c,ld;
228     la=a->next;
229     lb=b->next;
230     c=(list)malloc(sizeof(node));
231     lc=c;
232     lc->next=NULL;

```

```
233 while(la)
234 {
235     lb=b->next;
236     while(lb)
237     {
238         ld=(list)malloc(sizeof(node));
239         ld->next=NULL;
240         ld->coe=la->coe*lb->coe;
241         ld->exp=la->exp+lb->exp;
242         lc->next=ld;
243         lc=ld;
244         lb=lb->next;
245     }
246     la=la->next;
247 }
248 output(c);
249 }
250
251 void fx(list a,double x)//函数: 多项式在x处的值
252 {
253     list la;
254     la=a->next;
255     double value=0;
256     while(la)
257     {
258         value+=(la->coe)*pow(x,la->exp);
259         la=la->next;
260     }
261     printf("在%.1f处的值为:%.1f\n",m,x,m,value);
262 }
263
264 void diff(list a)//函数: 求一阶导
265 {
266     list la,b,lb,lc;
267     la=a->next;
268     b=(list)malloc(sizeof(node));
269     lb=b;
270     lb->next=NULL;
271     while(la)
272     {
273         lc=(list)malloc(sizeof(node));
274         if(la->exp==0)
275         {
276             lc->coe=0.0;
277             lc->exp=0;
278         }
279         else
```

```
280     {
281         lc->coe=la->coe*la->exp;
282         lc->exp=la->exp-1;
283     }
284     lc->next=NULL;
285     lb->next=lc;
286     lb=lc;
287     la=la->next;
288 }
289 printf("的一阶导为:");
290 output(b);
291 }
292
293 int main()
294 {
295     list A,B;
296     int t;
297     printf("请输入全部数保留小数的位数: ");
298     scanf("%d",&m);
299     printf("多项式A:\n");
300     A=input();
301     printf("多项式B:\n");
302     B=input();
303     printf("多项式A:");
304     output(A);
305     printf("多项式B:");
306     output(B);
307     while(1)
308     {
309         printf("选择运算功能: 0:退出 1:加 2:减 3:乘 4:求一阶导 5:在x处的值\n");
310         scanf("%d",&t);
311         switch (t)
312         {
313             case 0:exit(0);
314             case 1:printf("多项式相加结果为: ");plus(A,B);break;
315             case 2:printf("多项式相减结果为: ");minus(A,B);break;
316             case 3:printf("多项式相乘结果为: ");multi(A,B);break;
317             case 4:
318             {
319                 printf("多项式A");diff(A);
320                 printf("多项式B");diff(B);
321             }break;
322             case 5:
323             {
324                 double x;
325                 printf("请输入x的值:");
326                 scanf("%lf",&x);
```

```
327     printf("多项式A:");
328     fx(A,x);
329     printf("多项式B:");
330     fx(B,x);
331     }break;
332     default:printf("输入错误, 请重新输入! \n");
333     }
334 }
335 system("pause");
336 return 0;
337 }
```