

## 1. 实验题目:

## 1.5⑤ 一元稀疏多项式计算器

## 【问题描述】

设计一个一元稀疏多项式简单计算器。

## 【基本要求】

一元稀疏多项式简单计算器的基本功能是:

- (1) 输入并建立多项式;
- (2) 输出多项式,输出形式为整数序列: $n, c_1, e_1, c_2, e_2, \dots, c_n, e_n$ , 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  $a-b$ 。

## 【测试数据】

- (1)  $(2x+5x^8-3.1x^{11})+(7-5x^8+11x^9)=(-3.1x^{11}+11x^9+2x+7)$
- (2)  $(6x^{-3}-x+4.4x^2-1.2x^3)-(-6x^{-3}+5.4x^2-x^2+7.8x^{15})$   
 $=(-7.8x^{15}-1.2x^9+12x^{-3}-x)$
- (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

## 【实现提示】

用带头结点的单链表存储多项式,多项式的项数存放在头结点。

• 81 •

## 【选作内容】

- (1) 计算多项式在  $x$  处的值。
- (2) 求多项式  $a$  的导函数  $a'$ 。
- (3) 多项式  $a$  和  $b$  相乘, 建立乘积多项式  $ab$ 。
- (4) 多项式的输出形式为类数学表达式。例如, 多项式  $-3x^8+6x^3-18$  的输出形式为  $-3x^8+6x^3-18$ ,  $x^{15}+(-8)x^7-14$  的输出形式为  $x^{15}-8x^7-14$ 。注意, 系数值为 1 的非零次项的输出形式中略去系数 1, 如项  $1x^8$  的输出形式为  $x^8$ , 项  $-1x^3$  的输出形式为  $-x^3$ 。
- (5) 计算器的仿真界面。

## 2. 运行结果:

(1)

```
Microsoft Visual Studio 调试控制台
input your function:+
input items of pa,pb in order:3 3
input your expn:2 1 5 8 -3.1 11
input your expn:7 0 -5 8 11 9
第一个多项式为: -3.10 x^11 + 5.00 x^8+ 2.00 x^1
第二个多项式为: 11.00 x^9 + -5.00 x^8+ 7.00
运算结果为: -3.10 x^11 + 11.00 x^9+ 2.00 x^1+ 7.00
```

(2)

```
Microsoft Visual Studio 调试控制台
input your function:-
input items of pa,pb in order:4 4
input your expn:6 -3 -1 1 4.4 2 -1.2 9
input your expn:-6 -3 5.4 2 -1 2 7.8 15
第一个多项式为: -1.20 x^9 + 4.40 x^2+ -1.00 x^1+ 6.00 x^-3
第二个多项式为: 7.80 x^15 + -1.00 x^2+ 5.40 x^2+ -6.00 x^-3
运算结果为: -7.80 x^15 + -1.20 x^9+ -1.00 x^1+ 12.00 x^-3
```

相等的时候只后移 pb 指针，pa 指针不动即可

(3)

```
Microsoft Visual Studio 调试控制台
input your function:+
input items of pa,pb in order:6 2
input your expn:1 0 1 1 1 2 1 3 1 4 1 5
input your expn:-1 3 -1 4
第一个多项式为: 1.00 x^5 + 1.00 x^4+ 1.00 x^3+ 1.00 x^2+ 1.00 x^1+ 1.00
第二个多项式为: -1.00 x^4 + -1.00 x^3
运算结果为: 1.00 x^5 + 1.00 x^2+ 1.00 x^1+ 1.00
E:\课程实验\data_stru\Unary_polynomial\x64\Debug\Unary_polynomial.exe (进程 25088) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

(4)

```
Microsoft Visual Studio 调试控制台
input your function:+
input items of pa,pb in order:2 2
input your expn:1 1 1 3
input your expn:-1 1 -1 3
第一个多项式为: 1.00 x^3 + 1.00 x^1
第二个多项式为: -1.00 x^3 + -1.00 x^1
运算结果为: 0
E:\课程实验\data_stru\Unary_polynomial\x64\Debug\Un
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。 . . .
```

(5)

```
Microsoft Visual Studio 调试控制台

input your function:+
input items of pa,pb in order:2 2
input your expn:1 1 1 100
input your expn:1 100 1 200
第一个多项式为: 1.00 x^100 + 1.00 x^1
第二个多项式为: 1.00 x^200 + 1.00 x^100
运算结果为: 1.00 x^200 + 2.00 x^100+ 1.00 x^1
```

(6)

```
Microsoft Visual Studio 调试控制台

input your function:+
input items of pa,pb in order:3 1
input your expn:1 1 1 2 1 3
input your expn:0 0
第一个多项式为: 1.00 x^3 + 1.00 x^2+ 1.00 x^1
第二个多项式为: 0.00 x^0
运算结果为: 1.00 x^3 + 1.00 x^2+ 1.00 x^1+ 0.00
```

(7)

```
Microsoft Visual Studio 调试控制台

input your function:+
input items of pa,pb in order:1 3
input your expn:0 0
input your expn:1 1 1 2 1 3
第一个多项式为: 0.00 x^0
第二个多项式为: 1.00 x^3 + 1.00 x^2+ 1.00 x^1
运算结果为: 1.00 x^3 + 1.00 x^2+ 1.00 x^1+ 0.00
```

### 3. 代码分析:

ListNode 为多项式链表的结点, polynomial 为多项式;

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  using namespace std;
4  struct ListNode {
5      float coef; //系数
6      int expn;
7      ListNode* next;
8  };
9  struct polynomial {
10     ListNode* head;
11     ListNode* tail;
12 };
```

根据输入项数从键盘读取输入为结点参数, 并建立多项式链表, p 为链表指针:

```

13 void CreatePolyn(polynomial* p, int m) {
14     float coef;
15     int expn;
16     p->head = (ListNode*)malloc(sizeof(ListNode));
17     p->tail = (ListNode*)malloc(sizeof(ListNode));
18     printf("input your expn:");
19     if (m == 0) {
20         p->head->next = NULL;
21         p->tail = NULL;
22         printf("\n");
23     }
24     for (int i = 0; i < m; i++) {
25         scanf_s("%f %d", &coef, &expn);
26         ListNode* node = (ListNode*)malloc(sizeof(ListNode));
27         // if (node == NULL) return;
28         node->coef = coef;
29         node->expn = expn;
30         if (i == 0) {
31             p->head->next = node;
32             p->tail = node;
33             node->next = NULL;
34         }
35         else {
36             p->tail->next = node;
37             node->next = NULL;
38             p->tail = node;
39         }
40     }
41     return;
42 }

```

顺序打印链表:

```

56 void PrintPolyn(polynomial* p) {
57     // printf("p->head:%d\n", p->head);
58     // printf("p->head->next:%d\n", p->head->next);
59     // printf("p->tail:%d\n", p->tail);
60     // printf("input unary_polynomial:");
61     ListNode* trav = (ListNode*)malloc(sizeof(ListNode));
62     // ListNode* next = (ListNode*)malloc(sizeof(ListNode));
63     trav = p->head->next;
64     // printf("trava is:%d", trav);
65     if (trav == NULL) {
66         printf("0\n");
67         return;
68     }
69     printf("%.2f x^%d ", trav->coef, trav->expn);
70     trav = trav->next;
71     while (trav != NULL) {
72         if (trav->expn == 0) printf("+ %.2f ", trav->coef);
73         else printf("+ %.2f x^%d", trav->coef, trav->expn);
74         trav = trav->next;
75     }
76     printf("\n");
77 }
78
79 int PolynLength(polynomial* p) { //返回多项式项数
80     int item = 0;
81     ListNode* trav = (ListNode*)malloc(sizeof(ListNode));
82     trav = p->head;
83     while (trav != NULL) {
84         trav = trav->next;
85         item++;
86     }
87     return item;
88 }

```

链表相加——分三种情况考虑，指数相等时类似状态机转入另外两个状态或结束循环：



```

90 void AddPolyn(polynomial* pa, polynomial* pb) {
91     ListNode* trava = (ListNode*)malloc(sizeof(ListNode));
92     ListNode* trava_ah = (ListNode*)malloc(sizeof(ListNode));
93     ListNode* travb = (ListNode*)malloc(sizeof(ListNode));
94     ListNode* travb_ah = (ListNode*)malloc(sizeof(ListNode));
95     ListNode* tema = (ListNode*)malloc(sizeof(ListNode));
96     ListNode* temb = (ListNode*)malloc(sizeof(ListNode));
97     trava_ah = pa->head;
98     // travb_ah = pb->head;
99     trava = pa->head->next;
100    travb = pb->head->next;
101    //用头结点不用考虑pa只有一项
102    while (trava != NULL && travb != NULL) {
103        if (trava->expn > travb->expn) { //trava指数小于travb, 将travb插入到trava与trava_ah之间, 并释放travb_ah,
104            temb = travb->next;
105            trava_ah->next = travb;
106            travb->next = trava;
107            trava_ah = travb;
108            travb = temb;
109        }
110        //trava->next==travb->expn isn't thoroughly considered
111        else if (trava->expn < travb->expn) {
112            while (trava->next != NULL && trava->next->expn <= travb->expn) {
113                trava_ah = trava;
114                trava = trava->next; //trava<travb<trava->next||trava->next=NULL, trava->travb->trava->next;
115            }
116            if (trava->expn == travb->expn) {
117                goto equal;
118            }
119            tema = trava->next;
120            trava->next = travb;
121            temb = travb->next;
122            travb->next = tema;
123            trava_ah = travb; //travb成为trava->next的ah
124            trava = tema;
125            travb = temb;
126        }
127        else {
128            equal:
129            trava->coef += travb->coef;
130            if (trava->coef == 0) {
131                tema = trava->next;
132                trava_ah->next = tema;
133                free(trava);
134                trava = tema;
135            }
136            else {
137                trava_ah = trava;
138                trava = trava->next;
139            }
140            temb = travb->next;
141            free(travb);
142            travb = temb;
143        }
144    }
145    //trava=NULL或travb=NULL
146    if (trava == NULL) {
147        trava_ah->next = travb;
148    }
149    free(pb->head);
150    free(pb);
151 }

```

链表相减，与链表相加差不多，区别如下：

1. 插入时需要把系数\* (-1)
2. 指数相同为相减

```

173 void SubPolyn(polynomial* pa, polynomial* pb) {
174     ListNode* trava = (ListNode*)malloc(sizeof(ListNode));
175     ListNode* trava_ah = (ListNode*)malloc(sizeof(ListNode));
176     ListNode* travb = (ListNode*)malloc(sizeof(ListNode));
177     // ListNode* travb_ah = (ListNode*)malloc(sizeof(ListNode));
178     ListNode* tema = (ListNode*)malloc(sizeof(ListNode));
179     ListNode* temb = (ListNode*)malloc(sizeof(ListNode));
180     trava_ah = pa->head;
181     // travb_ah = pb->head;
182     trava = pa->head->next;
183     travb = pb->head->next;
184     //用头结点不用考虑pa只有一项
185     while (trava != NULL && travb != NULL) {
186         if (trava->expn > travb->expn) { //trava指数小于travb, 将travb插入到trava与trava_ah之间, 并释放travb_ah,
187             travb->coef = travb->coef * (-1);
188             temb = travb->next;
189             trava_ah->next = travb;
190             travb->next = trava;
191             trava_ah = travb;
192             trava = temb;
193         }
194         //trava->next==travb->expn isn't thoroughly considered
195         else if (trava->expn < travb->expn) {
196             while (trava->next != NULL && trava->next->expn <= travb->expn) {
197                 trava_ah = trava;
198                 trava = trava->next; //trava<travb<trava->next||trava->next=NULL, trava->travb->trava->next;
199             }
200             if (trava->expn == travb->expn) {
201                 goto equal;
202             }
203             travb->coef *= (-1);
204             tema = trava->next;
205             trava->next = travb;
206             temb = travb->next;
207             travb->next = tema;
208             trava_ah = travb; //travb成为trava->next的ah
209             trava = tema;
210             trava = temb;
211         }
212         else {
213             equal:
214             trava->coef -= travb->coef;
215             if (trava->coef == 0) {
216                 tema = trava->next;
217                 trava_ah->next = tema;
218                 free(trava);
219                 trava = tema;
220             }
221             else {
222                 trava_ah = trava;
223                 trava = trava->next;
224             }
225             temb = travb->next;
226             free(travb);
227             travb = temb;
228         }
229     }
230     //trava=NULL或travb=NULL
231     if (trava == NULL) {
232         trava_ah->next = travb;
233     }
234     free(pb->head);
235     free(pb);
236 }

```

翻转链表，输入时为按指数升序，翻转后按指数降序来输出：

```

153 int invert_list(polynomial* p) { //when p至少有一项
154     if (p->head->next == NULL) return 0;
155     ListNode* trav1 = (ListNode*)malloc(sizeof(ListNode));
156     ListNode* trav2 = (ListNode*)malloc(sizeof(ListNode));
157     ListNode* stor = (ListNode*)malloc(sizeof(ListNode));
158     trav1 = p->head->next;
159     trav2 = trav1->next;
160     while (trav2 != NULL) {
161         stor = trav2->next;
162         trav2->next = trav1;
163         trav1 = trav2;
164         trav2 = stor;
165     }
166     stor = p->head->next;
167     p->head->next = trav1;
168     p->tail = stor;
169     stor->next = NULL;
170     return 1;
171 }

```

该程序主函数：

```

237
238 int main() {
239     polynomial* pa = (polynomial*)malloc(sizeof(polynomial));
240     polynomial* pb = (polynomial*)malloc(sizeof(polynomial));
241     char func; //指示"+","-"法
242     int item_a, item_b;
243     printf("input your function:");
244     scanf_s("%c", &func);
245     printf("input items of pa,pb in order:");
246     scanf_s("%d %d", &item_a, &item_b);
247     CreatePolyn(pa, item_a);
248     CreatePolyn(pb, item_b);
249     invert_list(pa);
250     printf("第一个多项式为: ");
251     PrintPolyn(pa);
252     invert_list(pa);
253     invert_list(pb);
254     printf("第二个多项式为: ");
255     PrintPolyn(pb);
256     invert_list(pb);
257     if (func == '+') AddPolyn(pa, pb);
258     else SubPolyn(pa, pb);
259     invert_list(pa);
260     printf("运算结果为: ");
261     PrintPolyn(pa);
262     return 0;
263 }

```