# **问题重述**

**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,c1,e1,c2,e2,…,cn,en,**

**其中n是多项式的项数，ci,ei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)计算多项式在x处的值；即给定x值，计算多项式值。**

**拓展:多项式乘法；多项式系数的合并和排序（所以不对输入序列有要求）**

# **分析过程**

一：输入多项式的系数和指数，初始化多项式。（其中多项式的结构如下:

typedef struct Node {

float coef; //系数

int exp; //指数

struct Node\* next;

}Node, \* Ploy;

typedef Ploy\* ptrPloy;)）

二:对多项式的按指数进行排序，然后将指数一样的节点进行合并。因此就实现了对序列按指数由小到大依次排序。

三;对多项式的x进行赋值求和并输出结果，就可以直接从头节点开始，遍历一遍多项式链，每到一个节点，就令coef=x(所赋予的值),然后计算coef^exp,将所有节点的值累加起来就是最终所要求的值。

四：多项式的加法与减法原理一样：先只分析加法。由于创建好的两个相加的多项式已经按指数从低到高已经排好序，对两个多项式进行加法的过程就好比对两个多项式的指数进行归并排序，在排序过程中对系数对应的系数进行加法运算，而且这里所说的归并排序只不过所用的归并不用递归，只用一趟就可以得到最终的结果，但系数相加后可能为0，就需要删除这个节点。

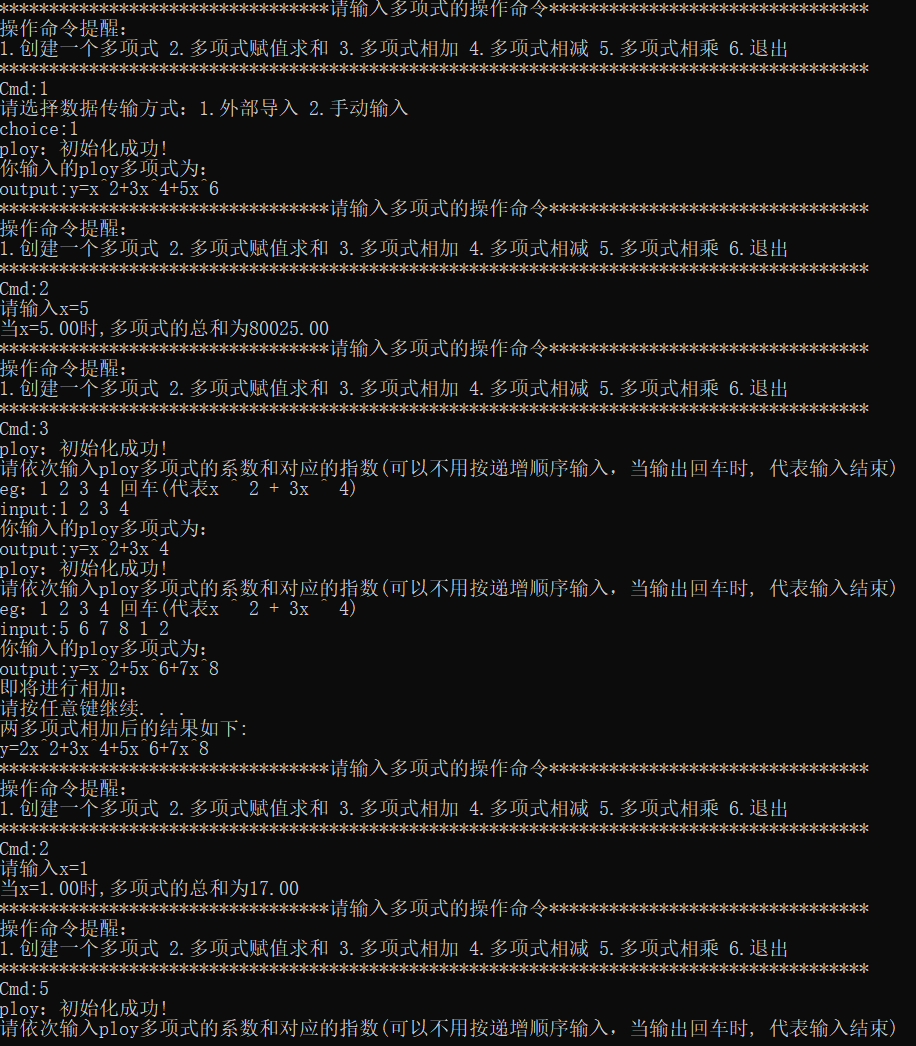
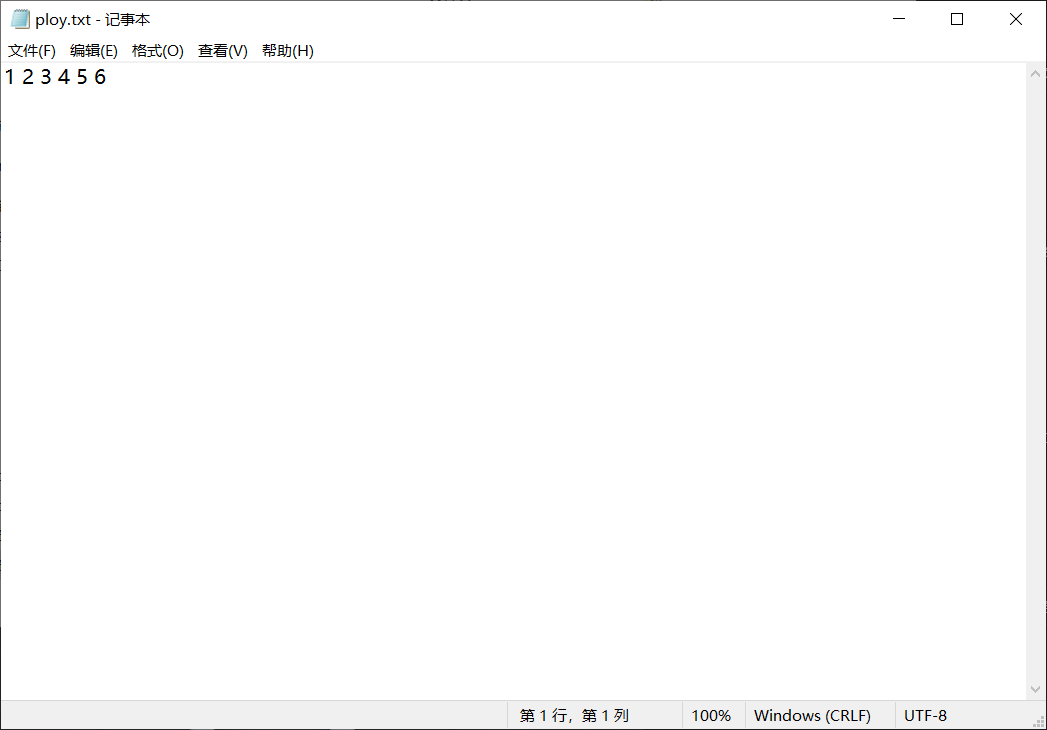
五：多项式的乘法（拓展）：运用双层循环，将多项式1中所有的节点都与多项式2中所有的节点进行乘法运算（系数\*系数，指数\*指数），然后用这些节点重新构建一个多项式，最后对多项式进行排序和合并操作就可以得到最终结果，刚开始再创建多项式时会浪费n\*m(n，m分别为多项式1和2的节点数)，但是经过排序和合并操作就会将多余节点的空间释放掉）。

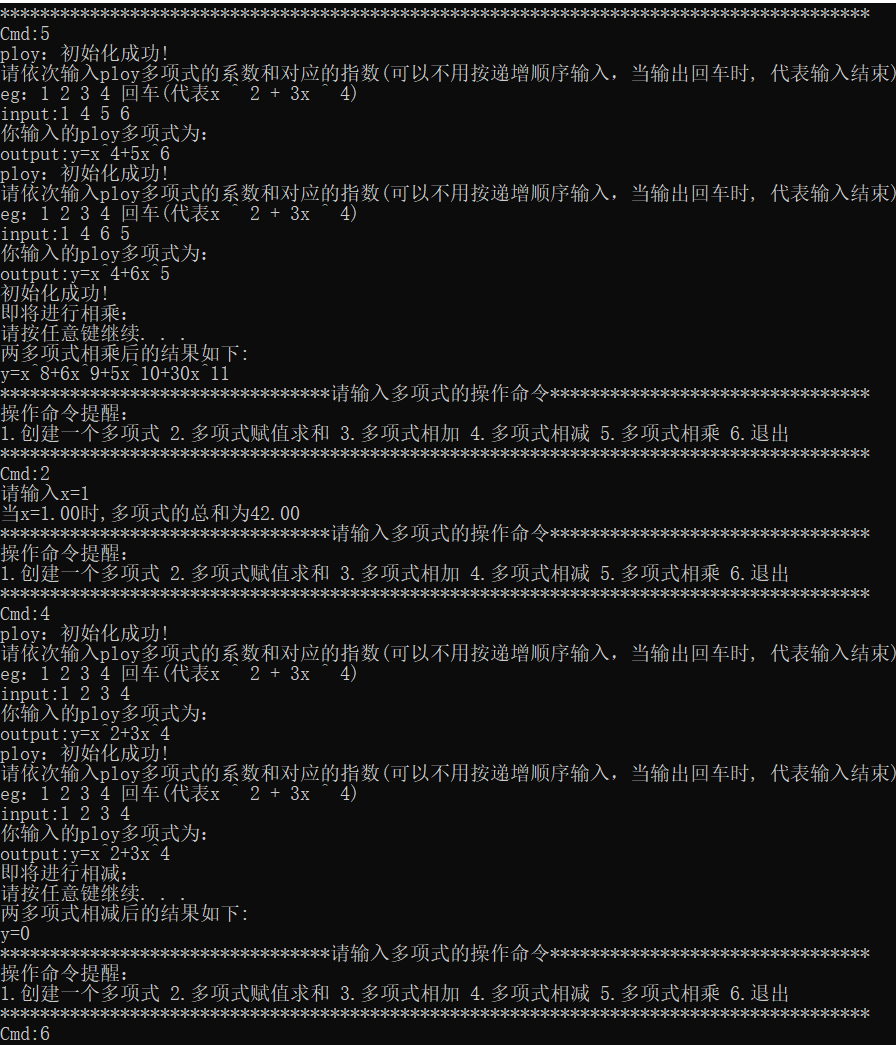
# **源代码**

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include<stdio.h>
3. #include<stdlib.h>
4. #include<math.h>
5. *//采用带链表进行多项式的加法、减法和乘法*
6. *//延伸：倘若输入的指数是无序的，而且可能是重复的，则牵扯到排序与合并。*
7. *//现尝试编写void Sort\_and\_Merge\_Ploy(ptrPloy Head)函数*
8. *//多项式乘法void Mult\_Ploy()*
9. *//多项式求和函数：对多项式赋值后进行求和void Sum\_Ploy()*
10. *//多项式分解函数？*
11. typedef struct Node {
12. float coef;   *//系数*
13. int exp;      *//指数*
14. struct Node\* next;
15. }Node, \* Ploy;
16. typedef Ploy\* ptrPloy;
17. void Init\_Ploy(ptrPloy Head);                       *//初始化操作*
18. void Add\_Ploy(ptrPloy pa, ptrPloy pb, ptrPloy pc);  *//进行多项式的加法*
19. void Sub\_Ploy(ptrPloy pa, ptrPloy pb, ptrPloy pc);  *//进行多项式的减法*
20. void Mult\_Ploy(ptrPloy pa, ptrPloy pb, ptrPloy pc); *//进行多项式的乘法*
21. void Sum\_Ploy(ptrPloy Head, float x,FILE\* fp);      *//对多项式x进行赋值，然后求和*
22. void Import\_Ploy(ptrPloy Head);                     *//输入多项式*
23. void Sort\_and\_Merge\_Ploy(ptrPloy Head);            *//对多项式内部进行排序和合并,其中排序采用快速||冒泡排序*
24. *//对单链表冒泡排序*
25. void Bubble\_Ploy(Ploy head);
26. *//合并同类项*
27. void Merge(Ploy head);
28. void Print\_Ploy(ptrPloy Head,FILE\* fp);                         *//打印多项式*
29. void Create\_Ploy(ptrPloy Head,FILE\* fp);                        *//对初始化，输入，打印操作进行封装，将主函数模块化，向面向对象方法靠近*
30. void Create\_Ploy\_txt(ptrPloy Head, FILE\* fp);                   *//以文本文件形式传输数据*
31. void Final\_Add\_Ans(ptrPloy pa, ptrPloy pb, ptrPloy pc,FILE\* fp);    *//输出最终相加后的结果*
32. void Final\_Sub\_Ans(ptrPloy pa, ptrPloy pb, ptrPloy pc,FILE\* fp);    *//输出最终相减后的结果*
33. void Final\_Mult\_Ans(ptrPloy pa, ptrPloy pb, ptrPloy pc,FILE\* fp);   *//输出最终相乘后的结果*
34. void Command\_Ploy(void);                                   *//命令提醒*
35. int main()
36. {
37. ptrPloy ploy,ploy1, ploy2,ploy3;
38. FILE\* fp;
39. int flag = 1;
40. int tag;
41. int way;
42. if ((fp = fopen("result.txt", "wb+")) == NULL) {
43. perror("the file fail to write");
44. system("pause");
45. exit(1);
46. }
47. while (flag) {
48. int choice;
49. float x=0;
50. Command\_Ploy();
51. scanf\_s("%d", &choice);
52. switch (choice) {
53. case 1:
54. printf("请选择数据传输方式：1.外部导入 2.手动输入\nchoice:");
55. scanf\_s("%d", &way);
56. while (way < 1 || way>2) {
57. printf("请选择正确的传输方式:1.外部导入 2.手动输入\nchoice:");
58. scanf\_s("%d", &way);
59. }
60. way==1?Create\_Ploy\_txt(&ploy,fp):Create\_Ploy(&ploy,fp);
61. tag = 0;
62. break;
63. case 2:
64. printf("请输入x=");
65. scanf\_s("%f", &x);
66. tag == 0 ? Sum\_Ploy(&ploy, x,fp) : Sum\_Ploy(&ploy3, x,fp);
67. break;
68. case 3:
69. fprintf(fp, "多项式加法:\n");
70. Create\_Ploy(&ploy1,fp);
71. Create\_Ploy(&ploy2,fp);
72. Final\_Add\_Ans(&ploy1, &ploy2, &ploy3,fp);
73. tag = 1;
74. break;
75. case 4:
76. fprintf(fp, "多项式减法:\n");
77. Create\_Ploy(&ploy1,fp);
78. Create\_Ploy(&ploy2,fp);
79. Final\_Sub\_Ans(&ploy1, &ploy2, &ploy3,fp);
80. tag = 1;
81. break;
82. case 5:
83. fprintf(fp, "多项式乘法:\n");
84. Create\_Ploy(&ploy1,fp);
85. Create\_Ploy(&ploy2,fp);
86. Init\_Ploy(&ploy);
87. Final\_Mult\_Ans(&ploy1, &ploy2, &ploy3,fp);
88. tag = 1;
89. break;
90. case 6:
91. flag = 0;
92. break;
93. default:
94. break;
95. }
96. }
97. fclose(fp);
98. return 0;
99. }
100. void Init\_Ploy(ptrPloy Head)
101. {
102. Ploy temp;
103. if (\*Head == NULL);
104. {
105. temp = (Ploy)malloc(sizeof(Node));
106. if (\*Head == NULL)
107. {
108. printf("初始化失败！\n");
109. return;
110. }
111. \*Head = temp;
112. (\*Head)->next = NULL;
113. }
114. printf("初始化成功!\n");
115. return;
116. }
117. void Add\_Ploy(ptrPloy pa, ptrPloy pb, ptrPloy pc)
118. {
119. \*pc = \*pa;
120. Ploy at = (\*pa)->next;
121. Ploy bt = (\*pb)->next;
122. Ploy ct = (\*pc);
123. Ploy tag;
124. float temp;
125. while (at && bt) {
126. if (at->exp < bt->exp) {
127. ct = at;
128. at = at->next;
129. }
130. else if (at->exp == bt->exp) {
131. temp = at->coef + bt->coef;
132. if (temp != 0) {
133. at->coef = temp;
134. ct = at;
135. }
136. else {
137. ct->next = at->next;
138. free(at);
139. }
140. at = ct->next;
141. bt = bt->next;
142. }
143. else {
144. tag = bt->next;
145. bt->next = at;
146. ct->next = bt;
147. ct = ct->next;  *// ct往后移*
148. bt = tag;
149. }
150. }
151. if (bt != NULL) {
152. Ploy temp = (Ploy)malloc(sizeof(Node));
153. if (temp) {
154. temp = bt;
155. ct->next = temp;
156. }
157. *//ct->next = bt;*
158. }
159. *//free(pb);*
160. }
161. void Sub\_Ploy(ptrPloy pa, ptrPloy pb, ptrPloy pc)
162. {
163. \*pc = \*pa;
164. Ploy at = (\*pa)->next;
165. Ploy bt = (\*pb)->next;
166. Ploy ct = (\*pc);
167. Ploy tag;
168. float temp;
169. while (at && bt) {
170. if (at->exp < bt->exp) {
171. ct = at;
172. at = at->next;
173. }
174. else if (at->exp == bt->exp) {
175. temp = at->coef - bt->coef;
176. if (temp != 0) {
177. at->coef = temp;
178. ct = at;
179. }
180. else {
181. ct->next = at->next;
182. free(at);
183. }
184. at = ct->next;
185. bt = bt->next;
187. }
188. else {
189. tag = bt->next;
190. bt->next = at;
191. ct->next = bt;
192. bt->coef = -bt->coef;
193. ct = ct->next;  *// ct往后移*
194. bt = tag;
195. }
196. }
197. if (bt != NULL) {
198. Ploy temp = (Ploy)malloc(sizeof(Node));
199. if (temp) {
200. bt->coef = -bt->coef;
201. temp = bt;
202. ct->next = temp;
203. }
204. *//ct->next = bt;*
205. }
206. *//free(pb);*
207. }
208. void Mult\_Ploy(ptrPloy pa, ptrPloy pb, ptrPloy pc)
209. {
210. Ploy at = (\*pa)->next;
211. Ploy bt = (\*pb)->next;
212. *//初始化pc*
213. Ploy p;
214. if (\*pc == NULL);
215. {
216. p = (Ploy)malloc(sizeof(Node));
217. if (\*pc == NULL)
218. {
219. printf("初始化失败！\n");
220. return;
221. }
222. \*pc = p;
223. (\*pc)->next = NULL;
224. }
225. Ploy ct = \*pc;
226. for (at; at!=NULL; at = at->next) {
227. for (bt=(\*pb)->next; bt!=NULL; bt = bt->next) {
228. Ploy temp = (Ploy)malloc(sizeof(Node));
229. temp->coef = at->coef \* bt->coef;
230. temp->exp = at->exp + bt->exp;
231. ct->next = temp;
232. ct = ct->next;
233. }
234. }
235. ct->next = NULL;
236. Sort\_and\_Merge\_Ploy(pc);
237. }
238. void Sum\_Ploy(ptrPloy Head, float x,FILE\* fp)
239. {
240. float sum = 0;
241. Ploy p= (\*Head)->next;
242. while (p) {
243. sum+= p->coef \* pow(x, p->exp);
244. p = p->next;
245. }
246. printf("当x=%.2f时,多项式的总和为%.2f\n", x, sum);
247. fprintf(fp, "当x=%.2f时,多项式的总和为%.2f\n", x, sum);
248. }
249. void Import\_Ploy(ptrPloy Head)
250. {
251. *//printf("请依次输入多项式的系数和对应的指数(当输出回车时,代表输入结束)eg：1 2 3 4 回车(代表x^2+3x^4)\n");*
252. int exp;
253. float coef;
254. char ch = 0;*//用来停止输入*
255. Ploy q = \*Head;
256. Ploy p = (\*Head)->next;
257. while (ch != '\n') {
258. scanf\_s("%f %d", &coef, &exp);
259. p = (Ploy)malloc(sizeof(Node));
260. if (p) {
261. p->coef = coef;
262. p->exp = exp;
263. p->next = q->next;
264. q->next = p;
265. }
266. q = p;
267. p = p->next;
268. ch = getchar();
269. }
270. }
271. void Sort\_and\_Merge\_Ploy(ptrPloy Head)
272. {
273. Ploy head = (\*Head)->next;
274. Bubble\_Ploy(head);
275. *//再Merge*
276. Merge(head);
277. }
278. void Bubble\_Ploy(Ploy head)
279. {
280. Ploy p, q;
281. int exp;
282. float coef;
283. if (head == NULL || head->next == NULL) {
284. return;
285. }
286. for (p = head; p != NULL; p = p->next) {
287. for (q = p->next; q != NULL; q = q->next) {
288. if (p->exp > q->exp) {
289. exp = p->exp;
290. coef = p->coef;
291. p->exp = q->exp;
292. p->coef = q->coef;
293. q->exp = exp;
294. q->coef = coef;
295. }*//if*
296. }*//for*
297. }*//for*
298. }
299. void Merge(Ploy head)
300. {
301. Ploy q = (head);
302. Ploy p = q->next;
303. while (p) {
304. while (p != NULL&&(p->exp == q->exp)) {
305. q->coef += p->coef;
306. p = p->next;
307. }
308. if (q->coef == 0) {
309. if (p == NULL) {
310. q->coef = 0;
311. q->exp = 0;
312. }
313. else {
314. q->coef = p->coef;
315. q->exp = p->exp;
316. if (p->next == NULL) p = NULL;
317. else p = p->next;
318. }
319. q->next = p;
320. }
321. else {
322. if (p == NULL) {
323. q->next = p;
324. }
325. else {
326. q->next = p;
327. q = q->next;
328. p = p->next;
329. }*//else*
330. }*//else*
331. }*//while*
332. }
333. void Print\_Ploy(ptrPloy Head,FILE\* fp)
334. {
335. int flag = 0;
336. Ploy p = (\*Head)->next;
337. printf("y=");
338. fprintf(fp, "y=");
339. while (p) {
340. if (p->coef != 0) {
341. if ((p)->exp == 0) {
342. printf("%.0f", (p)->coef);
343. fprintf(fp, "%.0f", (p)->coef);
344. flag = 1;
345. }
346. else if ((p)->coef > 0) {
347. if ((p)->coef == 1) {
348. if ((p)->exp == 1) {
349. flag == 0 ? printf("x") : printf("+x");
350. flag==0?fprintf(fp, "x") :fprintf(fp, "+x");
351. flag = 1;
352. }
353. else {
354. flag == 0 ? printf("x^%d", (p)->exp) : printf("+x^%d", (p)->exp);
355. flag == 0 ? fprintf(fp,"x^%d", (p)->exp) : fprintf(fp,"+x^%d", (p)->exp);
356. flag = 1;
357. }
358. }
359. else {
360. if (p->exp == 1)
361. {
362. printf("%.0fx", p->coef);
363. fprintf(fp, "%.0fx", p->coef);
364. }
365. else {
366. flag == 0 ? printf("%.0fx^%d", (p)->coef, (p)->exp) : printf("+%.0fx^%d", (p)->coef, (p)->exp);
367. flag == 0 ? fprintf(fp,"%.0fx^%d", (p)->coef, (p)->exp) : fprintf(fp,"+%.0fx^%d", (p)->coef, (p)->exp);
368. }
369. flag = 1;
370. }
371. }
372. else {
373. if ((p)->coef == -1) {
374. if ((p)->exp == 1) {
375. printf("-x");
376. fprintf(fp, "-x");
377. flag = 1;
378. }
379. else {
380. printf("-x^%d", (p)->exp);
381. fprintf(fp,"-x^%d", (p)->exp);
382. flag = 1;
383. }
384. }
385. else {
386. if (p->exp == 1) {
387. printf("%.0fx", p->coef);
388. fprintf(fp,"%.0fx", p->coef);
389. }
390. else {
391. printf("%.0fx^%d", (p)->coef, (p)->exp);
392. fprintf(fp, "%.0fx^%d", (p)->coef, (p)->exp);
393. }
394. flag = 1;
395. }*//else*
396. }*//else*
397. }
398. p = p->next;
399. }
400. *//如果没有输出其他格式，说明多项式为0*
401. if (flag == 0) {
402. printf("0");
403. fprintf(fp, "0");
404. }
405. printf("\n");
406. fprintf(fp, "\n");
407. return;
408. }
409. void Create\_Ploy(ptrPloy Head,FILE\* fp)
410. {
411. printf("ploy：");
412. Init\_Ploy(Head);
413. printf("请依次输入ploy多项式的系数和对应的指数(可以不用按递增顺序输入，当输出回车时, 代表输入结束)\neg：1 2 3 4 回车(代表x ^ 2 + 3x ^ 4)\ninput:");
414. Import\_Ploy(Head);
415. Sort\_and\_Merge\_Ploy(Head);
416. printf("你输入的ploy多项式为：\noutput:");
417. Print\_Ploy(Head,fp);
418. }
419. void Create\_Ploy\_txt(ptrPloy Head, FILE\* fp)
420. {
421. printf("ploy：");
422. Init\_Ploy(Head);
423. FILE\* fp1;
424. int exp;
425. float coef;
426. char ch = 0;*//用来停止输入*
427. Ploy q = \*Head;
428. Ploy p = (\*Head)->next;
429. if ((fp1 = fopen("ploy.txt", "r")) == NULL) {
430. perror("the file ploy.txt fail to read");
431. system("pause");
432. exit(1);
433. }
435. while (!feof(fp1) && !ferror(fp1)) {
436. fscanf(fp1, "%f%d", &coef,&exp);
437. p = (Ploy)malloc(sizeof(Node));
438. if (p) {
439. p->coef = coef;
440. p->exp = exp;
441. p->next = q->next;
442. q->next = p;
443. }
444. q = p;
445. p = p->next;
446. }
447. Sort\_and\_Merge\_Ploy(Head);
448. printf("你输入的ploy多项式为：\noutput:");
449. Print\_Ploy(Head, fp);
450. }
451. void Final\_Add\_Ans(ptrPloy pa, ptrPloy pb, ptrPloy pc,FILE\* fp)
452. {
453. printf("即将进行相加：\n");
454. system("pause");
455. Add\_Ploy(pa, pb, pc);
456. printf("两多项式相加后的结果如下:\n");
457. fprintf(fp,"两多项式相加后的结果如下:\n");
458. Print\_Ploy(pc,fp);
459. }
460. void Final\_Sub\_Ans(ptrPloy pa, ptrPloy pb, ptrPloy pc,FILE\* fp)
461. {
462. printf("即将进行相减：\n");
463. system("pause");
464. Sub\_Ploy(pa, pb, pc);
465. printf("两多项式相减后的结果如下:\n");
466. fprintf(fp, "两多项式相减后的结果如下:\n");
467. Print\_Ploy(pc,fp);
468. }
469. void Final\_Mult\_Ans(ptrPloy pa, ptrPloy pb, ptrPloy pc,FILE\* fp)
470. {
471. printf("即将进行相乘：\n");
472. system("pause");
473. Mult\_Ploy(pa, pb, pc);
474. printf("两多项式相乘后的结果如下:\n");
475. fprintf(fp, "两多项式相乘后的结果如下\n");
476. Print\_Ploy(pc,fp);
477. }
478. void Command\_Ploy(void)
479. {
480. printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*请输入多项式的操作命令\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
481. printf("操作命令提醒：\n1.创建一个多项式 2.多项式赋值求和 3.多项式相加 4.多项式相减 5.多项式相乘 6.退出\n");
482. printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
483. printf("Cmd:");
484. }

# **测试**

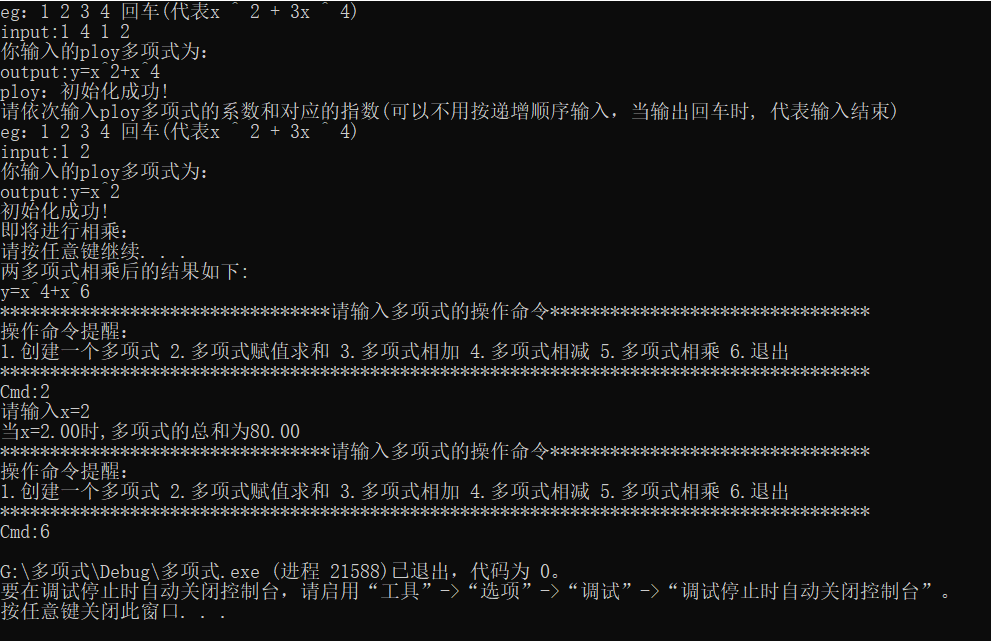
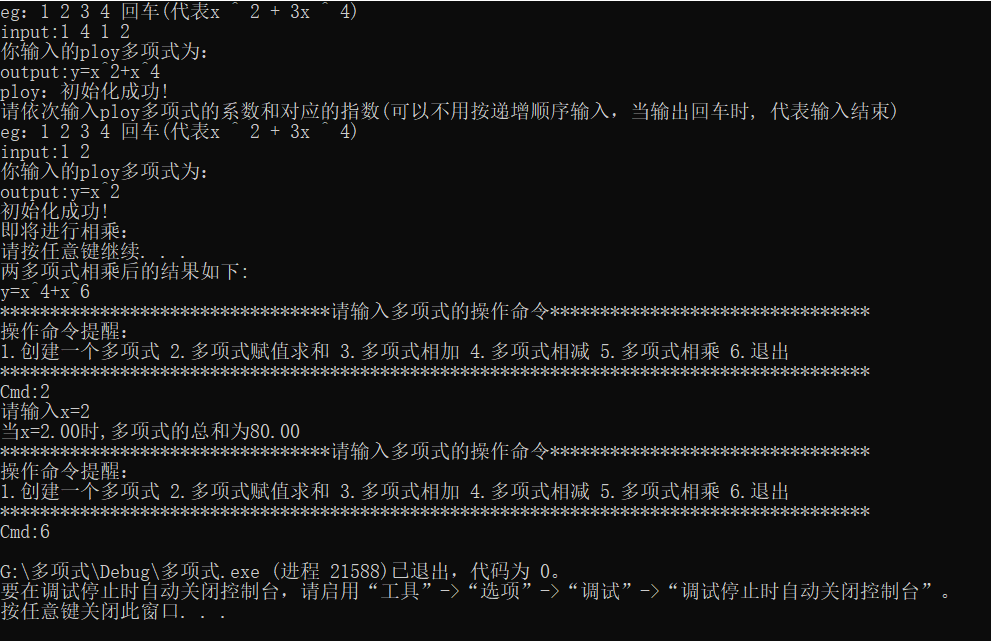
测试1：







测试2：



# **评价**

1.首先完成了题目的要求；

2.在对序列进行排序时，可以使用其他的排序方法，进行改进；

3.设计了简单的交互式界面，便于操作；

4.添加了外部导入文件的传输方式，并将结果写入到txt文件中；

5.依据加法和减法的思想，进行拓展，增加了多项式相乘的算法。