

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 #define LIST_INIT_SIZE 100 //存储空间初始分配量
5 #define LIST_INCREMENT 10 //存储空间分配增量
6
7 typedef int ElemType;
8 typedef struct
9 {
10     ElemType *base; //存储空间基址
11     int length; //当前表长
12     int listsize; //当前分配的存储容量
13 } SqList;
14
15 enum Status
16 {
17     OVERFLOW = -1,
18     ERROR = 0,
19     OK = 1
20 };
21
22 //函数声明
23 Status InitList_Sq(SqList& l);
24 Status ListInsert_Sq(SqList& l, int i, ElemType e);
25 Status ListDelete_Sq(SqList& l, int i, ElemType& e);
26 Status GetElem(SqList& la, int i, int& ai);
27 Status ListSort(SqList& l);
28 void MergeList(SqList& la, SqList& lb, SqList& lc);
29 Status PrintList(SqList& l);
30 void PrintStar(void);
31
32 //主函数
33 int main()
34 {
35     char chos;
36     ElemType elem;
37     int pos;
38
39     SqList la;
40     SqList lb;
41
42     SqList lc; //a表和b表的合并
43
44     InitList_Sq(la);
45     InitList_Sq(lb);
46     lc.base = NULL;
47
48     while(1)
49     {
```

```
50     printf("1.Insert a new elem to la.\n");
51     printf("2.Insert a new elem to lb.\n");
52
53     printf("3.Delete ist elem in la.\n");
54     printf("4.Delete ist elem in lb.\n");
55
56     printf("5.Sort la.\n");
57     printf("6.Sort lb.\n");
58
59     printf("7.Merge la & lb to lc.\n");
60
61     printf("8.Print list.\n");
62
63     printf("9.Quit.\n\n");
64
65     printf("Please choose what you want to do: ");
66
67     scanf_s("%c", &chos);
68     if(chos == 10)
69     {
70         scanf_s("%c", &chos);
71     }
72
73     switch(chos)
74     {
75         case '1':
76             {
77                 printf("Please input the position & value of the elem: ");
78                 scanf_s("%d,%d", &pos, &elem);
79
80                 ListInsert_Sq(la, pos, elem);
81             }break;
82         case '2':
83             {
84                 printf("Please input the position & value of the elem: ");
85                 scanf_s("%d,%d", &pos, &elem);
86
87                 ListInsert_Sq(lb, pos, elem);
88             }break;
89         case '3':
90             {
91                 printf("Please input the position of the elem: ");
92                 scanf_s("%d", &pos);
93
94                 if(ListDelete_Sq(la, pos, elem))
95                 {
96                     printf("elem %d in la = %d\n", pos, elem);
97                 }
98             }break;
```

```
99     case '4':
100     {
101         printf("Please input the position of the elem: ");
102         scanf_s("%d", &pos);
103
104         if(ListDelete_Sq(lb, pos, elem))
105         {
106             printf("elem %d in lb = %d\\n", pos, elem);
107         }
108     }break;
109     case '5':
110     {
111         ListSort(la);
112         printf("la has been sorted.\\n");
113     }break;
114     case '6':
115     {
116         ListSort(lb);
117         printf("lb has been sorted.\\n");
118     }break;
119     case '7':
120     {
121         Mergelist(la, lb, lc);
122     }break;
123     case '8':
124     {
125         printf("Please choose which list you want to printf: ");
126         scanf_s("%c", &chos);
127         if(chos == 10)
128         {
129             scanf_s("%c", &chos);
130         }
131
132         switch(chos)
133         {
134             case 'a':
135             {
136                 PrintStar();
137                 printf("la: ");
138                 PrintList(la);
139                 PrintStar();
140             }break;
141             case 'b':
142             {
143                 PrintStar();
144                 printf("lb: ");
145                 PrintList(lb);
146                 PrintStar();
147             }break;
```

```
148         case 'c':
149             {
150                 PrintStar();
151                 printf("lc: ");
152                 PrintList(lc);
153                 PrintStar();
154             }break;
155         default :
156             {
157                 printf("Choose2 error!\n");
158             }break;
159     }
160 }break;
161 case '9':
162     {
163         free(la.base);
164         free(lb.base);
165         if(!lc.base)
166             {
167                 free(lc.base);
168             }
169         exit(0);
170     }break;
171 default :
172     {
173         printf("Choose1 error!\n");
174     }break;
175 }
176
177 //system("pause");
178 putchar(10);
179 }
180
181 return 0;
182 }
183
184 //函数定义
185 Status InitList_Sq(SqList& l)
186 {
187     l.base = (ElemType *) malloc ((LIST_INIT_SIZE) * sizeof(ElemType));
188     if(!l.base) exit(OVERFLOW);
189
190     l.length = 0;
191     l.listsize = LIST_INIT_SIZE;
192
193     return OK;
194 } //InitList_Sq;
195
196 Status ListInsert_Sq(SqList& l, int i, ElemType e)
```

```
197 {
198     if(i<1 || i>l.length+1)
199     {
200         printf("Pos Error!\n");
201         return ERROR;
202     }
203
204     ElemType* p;
205
206     if(l.length >= l.listsize)
207     {
208         ElemType* newbase;
209
210         newbase = (ElemType *)
211                 realloc (l.base, (l.listsize + LIST_INCREMENT) * sizeof
212                         (ElemType));
213         if(!newbase) exit(OVERFLOW);
214         l.base = newbase;
215         l.listsize += LIST_INCREMENT;
216     }
217     if(i < l.length + 1)    //不是在表尾插入元素
218     {
219         ElemType* q;
220         q = &l.base[i - 1];
221
222         for(p = &l.base[l.length - 1]; p>=q; --p)
223         {
224             *(p+1) = *p;
225         }
226         *q = e;
227     }
228     else
229     {
230         l.base[i - 1] = e;
231     }
232
233     ++l.length;
234
235     return OK;
236 } //ListInsert_Sq;
237
238 Status ListDelete_Sq(SqList& l, int i, ElemType& e)
239 {
240     if((i < 1) || (i > l.length))
241     {
242         printf("Pos Error!\n");
243         return ERROR;
244     }
```

```
245
246     ElemType* p;
247     ElemType* q;
248
249     p = &l.base[i - 1];
250     e = *p;
251     q = l.base + l.length - 1;
252
253     for(++p; p <= q; ++p)
254     {
255         *(p-1) = *p;
256     }
257
258     --l.length;
259
260     return OK;
261 } //ListDelete_Sq;
262
263 Status GetElem(SqList& la, int i, int& ai)
264 {
265     if(i < 1 || i > la.length)
266     {
267         return ERROR;
268     }
269
270     ai = la.base[i - 1];
271
272     return OK;
273 } //GetElem;
274
275 //对int表进行冒泡排序
276 Status ListSort(SqList& l)
277 {
278     if(l.base == NULL)
279     {
280         printf("There is no this list!\n");
281         return ERROR;
282     }
283     if(l.length == 0)
284     {
285         printf("There is no elem in this list!\n");
286         return OK;
287     }
288
289     ElemType temp;
290     for(int i = 0; i < l.length - 1; ++i)
291     {
292         for(int j = 0; j < (l.length - 1) - i; ++j)
293         {
```

```
294     if (l.base[j] > l.base[j+1])
295     {
296         temp = l.base[j];
297         l.base[j] = l.base[j+1];
298         l.base[j+1] = temp;
299     }
300 }
301 }
302 return OK;
303 } //ListSort;
304
305 //合并线性表la与lb (递增顺序,无重复元素)
306 void MergeList(SqlList& la, SqlList& lb, SqlList& lc)
307 {
308     printf("la & lb will be sorted first.waiting...\n");
309     ListSort(la);
310     ListSort(lb);
311
312     int i = 1;
313     int j = 1;
314     ElemType ai;
315     ElemType bj;
316
317     if(lc.base)
318     {
319         free(lc.base);
320     }
321     InitList_Sq(lc);
322
323     while((i <= la.length) && (j <= lb.length))
324     {
325         GetElem(la, i, ai);
326         GetElem(lb, j, bj);
327
328         if(ai <= bj)
329         {
330             ListInsert_Sq(lc, lc.length + 1, ai);
331
332             if(ai == bj)
333             {
334                 ++j;
335             }
336
337             ++i;
338         }
339         else
340         {
341             ListInsert_Sq(lc, lc.length + 1, bj);
342
```

```
343         ++j;
344     }
345 }
346
347 while(i <= la.length)
348 {
349     GetElem(la, i++, ai);
350     ListInsert_Sq(lc, lc.length + 1, ai);
351 }
352 while(j <= lb.length)
353 {
354     GetElem(lb, j++, bj);
355     ListInsert_Sq(lc, lc.length + 1, bj);
356 }
357
358 PrintStar();
359 printf("la: ");
360 PrintList(la);
361 printf("lb: ");
362 PrintList(lb);
363 printf("lc: ");
364 PrintList(lc);
365 PrintStar();
366 } //MergeList;
367
368 Status PrintList(SqList& l)
369 {
370     if(l.base == NULL)
371     {
372         printf("The list haven't exist yet!\n");
373         return ERROR;
374     }
375     if(l.length == 0)
376     {
377         printf("There is no elem in this list!\n");
378         return OK;
379     }
380
381     for (ElemType *p = l.base; p <= &l.base[l.length - 1]; ++p)
382     {
383         printf("%d ", *p);
384     }
385
386     putchar(10);
387
388     //printf("PrintList has completed.\n");
389     return OK;
390 } //PrintList;
391
```



```
392 void PrintStar(void)
393 {
394     printf("*****\n");
395 } //PrintStar
```