**数据结构第二次上机实验报告**

学号：161730126 姓名：李双玖

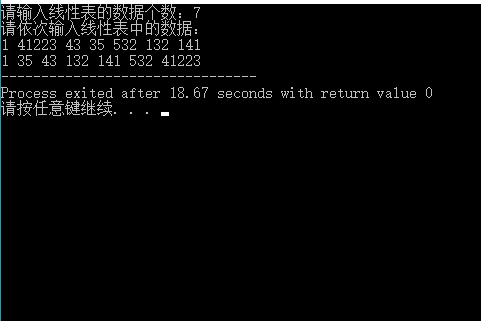
1. 调试成功程序及说明

**1、题目：设元素值为整型的线性表L，分别采用顺序结构和链式结构存储，编写程序，用插入排序算法实现线性表的表排序。**

**1）顺序结构：**

算法思想：插入排序。下标i从1到n-1遍历，将i位置的数插入前面已排好序的线性表中，即从后往前依次比较，比前一个小则交换，直到比前一个大。算法时间复杂度T(n^2)。

运行结果：



结果分析：正确

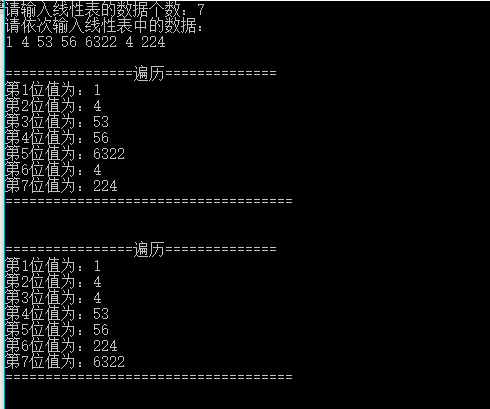
源程序：

1. #include<stdio.h>
2. #include<stdlib.h>
4. **int** main() {
6. **int** n = 0;
7. printf("请输入线性表的数据个数：");
8. scanf("%d", &n);
10. **int** \*L = (**int**\*)malloc(**sizeof**(**int**)\*n);
11. printf("请依次输入线性表中的数据：\n");
12. **int** i = 0;
13. **while**(i<n) {
14. scanf("%d", &L[i]);
15. i++;
16. }
18. **int** j = 0;
19. **int** temp = 0;
20. **for**(i=1; i<n; i++) {
21. **for**(j=i; j>0 && L[j-1] > L[j]; j--) {
22. temp = L[j-1];
23. L[j-1] = L[j];
24. L[j] = temp;
25. }
26. }
28. **for**(i=0; i<n; i++) printf("%d ", L[i]);
29. free(L);
30. }

**2）链式结构：**

算法思想：插入排序。从头到尾遍历链表，依次提出当前结点，插到前面已排好序的线性表中的相应位置。算法时间复杂度T(n^2)。

运行结果：



结果分析：正确

源程序：

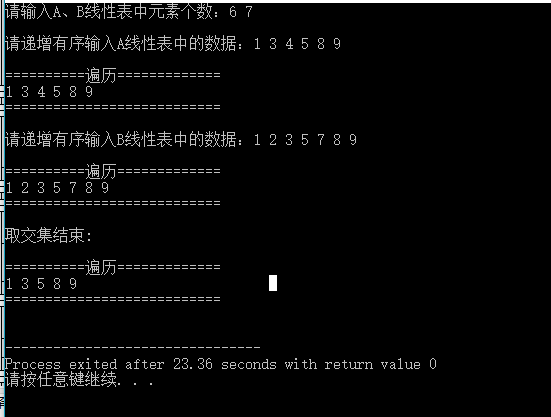
1. #include<stdio.h>
2. #include<stdlib.h>
4. **typedef** **struct** LNode {
5. **int** data;
6. LNode \*next;
7. }LNode, \*LinkList;
9. // 遍历函数
10. **void** ListTraverse(LinkList list) {
11. printf("\n================遍历==============\n");
12. LNode \*p = list->next;
13. **int** i = 1;
14. **while**(p) {
15. printf("第%d位值为：%d\n", i, p->data);
16. p = p->next;
17. i++;
18. }
19. printf("====================================\n\n");
20. }
22. // 在已排好序的链表中插入一个结点
23. **void** InsertList(LinkList &list, LNode \*temp) {
24. LNode \*prior = list, \*p = list->next;
25. **while**(p && p->data < temp->data) {
26. prior = p;
27. p = p->next;
28. }
29. prior->next = temp;
30. temp->next = p;
31. }
33. **int** main() {
35. **int** n = 0;
36. printf("请输入线性表的数据个数：");
37. scanf("%d", &n);
39. LinkList list = (LNode\*)malloc(**sizeof**(LNode));
40. list->data = 0;
41. list->next = NULL;
43. // 输入数据
44. LNode \*temp, \*p = list;
45. printf("请依次输入线性表中的数据：\n");
46. **int** i = 0;
47. **while**(i<n) {
48. temp = (LNode\*)malloc(**sizeof**(LNode));
49. scanf("%d", &temp->data);
50. p->next = temp;
51. p = p->next;
52. i++;
53. }
54. p->next = NULL;
55. ListTraverse(list);
57. // 依次取出结点并插入已排序的链表
58. p = list->next;
59. list->next = NULL;
60. **while**(p) {
61. temp = p;
62. p = p->next;
63. InsertList(list, temp);
64. }
65. ListTraverse(list);
67. //销毁链表
68. p = list;
69. **while**(p) {
70. temp = p;
71. p = p->next;
72. free(temp);
73. }
74. }

**2、题目：设元素值为整型的线性表A,、B，分别采用顺序结构和链式结构存储，编写程序，实现下列功能：假设以递增有序排列的线性表A和B 分别表示两个集合，现要求在A的空间上构成一个新线性表C，其元素为A和B元素的交集，且表C中的元素也是依值递增有序排列。**

**1）顺序结构：**

算法思想：用i、j下标分别控制A、B顺序表。在i、j都未到表尾时，对A表中i位置数和B表中j位置数进行比较：如果相等，则i++，j++；如果A[i]大，则j++；如果B[j]大，则i++。算法时间复杂度T(A.length+B.length)。

运行结果：



结果分析：正确

源程序：

1. #include<stdio.h>
2. #include<stdlib.h>
4. **typedef** **struct**{
5. **int** \*list;
6. **int** length;
7. }SqList ;
9. **void** ListTraverse(SqList L) {
10. printf("\n==========遍历=============\n");
11. **for**(**int** i=0; i<L.length; i++) {
12. printf("%d ", L.list[i]);
13. }
14. printf("\n===========================\n\n");
15. }
17. **int** main() {
19. // 初始化
20. SqList La, Lb, Lc;
21. printf("请输入A、B线性表中元素个数：");
22. scanf("%d%d", &La.length, &Lb.length);
23. La.list = (**int**\*)malloc(**sizeof**(**int**) \* La.length);
24. Lb.list = (**int**\*)malloc(**sizeof**(**int**) \* Lb.length);
26. printf("\n请递增有序输入A线性表中的数据：");
27. **for**(**int** i=0; i<La.length; i++) {
28. scanf("%d", &La.list[i]);
29. }
30. ListTraverse(La);
32. printf("请递增有序输入B线性表中的数据：");
33. **for**(**int** i=0; i<Lb.length; i++) {
34. scanf("%d", &Lb.list[i]);
35. }
36. ListTraverse(Lb);

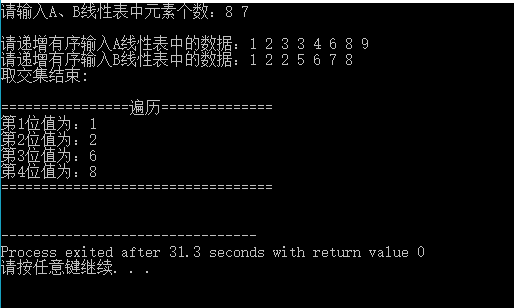
39. // 将A、B取交
40. **int** i=0, j=0;
41. **while**(i<La.length && j<Lb.length) {
42. **if**(La.list[i] == Lb.list[j]) {
43. i++;
44. j++;
45. }
46. **else** **if**(La.list[i] < Lb.list[j]) {
47. **for**(**int** k=i; k<La.length-1; k++) {
48. La.list[k] = La.list[k+1];
49. }
50. La.length--;
51. }
52. **else** {
53. j++;
54. }
55. }
56. **if**(i < La.length) {
57. La.length = i;
58. }
59. printf("取交集结束:\n");
60. Lc.list = La.list;
61. Lc.length = La.length;
62. ListTraverse(Lc);

65. // 销毁线性表
66. free(La.list);
67. free(Lb.list);
68. }

**2）链式结构：**

算法思想：同顺序结构，只是遍历用链表方法而已。算法时间复杂度T(A.length+B.length)。

运行结果：



结果分析：正确

源程序：

1. #include<stdio.h>
2. #include<stdlib.h>
4. **typedef** **struct** LNode {
5. **int** data;
6. LNode \*next;
7. }\*LinkList;
9. //遍历函数
10. **void** ListTraverse(LinkList list) {
11. printf("\n================遍历==============\n");
12. LNode \*p = list->next;
13. **int** i = 1;
14. **while**(p) {
15. printf("第%d位值为：%d\n", i, p->data);
16. p = p->next;
17. i++;
18. }
19. printf("==================================\n\n");
20. };
22. **int** main() {
24. // 初始化
25. LinkList listA = (LNode\*)malloc(**sizeof**(LNode));
26. listA->next = NULL;
27. LinkList listB = (LNode\*)malloc(**sizeof**(LNode));
28. listB->next = NULL;
29. printf("请输入A、B线性表中元素个数：");
30. scanf("%d%d", &listA->data, &listB->data);
32. LNode \*p, \*temp;
33. printf("\n请递增有序输入A线性表中的数据：");
34. p = listA;
35. **for**(**int** i=0; i<listA->data; i++) {
36. temp = (LNode\*)malloc(**sizeof**(LNode));
37. scanf("%d", &temp->data);
38. p->next = temp;
39. p = p->next;
40. }
41. p->next = NULL;
42. //ListTraverse(listA);
44. printf("请递增有序输入B线性表中的数据：");
45. p = listB;
46. **for**(**int** i=0; i<listB->data; i++) {
47. temp = (LNode\*)malloc(**sizeof**(LNode));
48. scanf("%d", &temp->data);
49. p->next = temp;
50. p = p->next;
51. }
52. p->next = NULL;
53. //ListTraverse(listB);

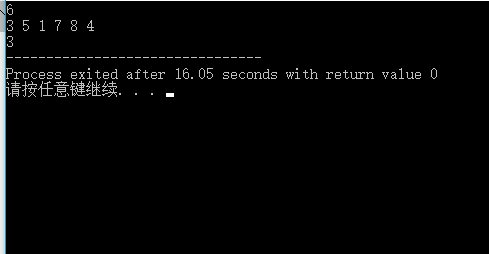
56. // 将B、C取交集
57. LNode \*pA = listA->next, \*pB = listB->next;
58. LNode \*priorA = listA, \*priorB = listB;
59. **while**(pA && pB) {
60. **if**(pA->data == pB->data) {
61. priorB->next = pB->next;
62. free(pB);
63. pB = priorB->next;
65. priorA = pA;
66. pA = priorA->next;
67. }
68. **else** **if**(pA->data < pB->data) {
69. priorA->next = pA->next;
70. free(pA);
71. pA = priorA->next;
72. }
73. **else** {
74. priorB->next = pB->next;
75. free(pB);
76. pB = priorB->next;
77. }
78. }
79. **while**(pA) {
80. temp = pA;
81. pA = pA->next;
82. free(pA);
83. }
84. priorA->next = NULL;
85. printf("取交集结束:\n");
86. ListTraverse(listA);

89. // 销毁链表
90. p = listA;
91. **while**(p) {
92. temp = p;
93. p = p->next;
94. free(temp);
95. }
97. p = listB;
98. **while**(p) {
99. temp = p;
100. p = p->next;
101. free(temp);
102. }
103. }

**3、题目：问题描述：先输入一个十进制整数n，再输入n个正整数，求它们相邻数之差可知是否为上升或下降，由上升转下降或由下降转上升为折点，求折点数。**

算法思想：如果一个点的值比左右两个都大或都小，则为折点。算法时间复杂度T(n)。

运行结果：



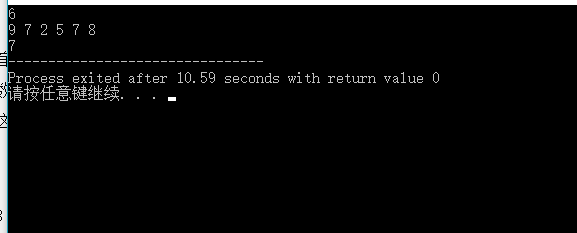
结果分析：正确

源程序：

1. #include<stdio.h>
2. #include<stdlib.h>
4. **int** main() {
6. **int** n = 0;
7. scanf("%d", &n);
9. **int**\* a = (**int**\*)malloc(**sizeof**(**int**) \* n);
10. **for**(**int** i=0; i<n; i++) {
11. scanf("%d", &a[i]);
12. }
14. **int** account = 0;
15. **for**(**int** i=1; i<n-1; i++) {
16. **if**((a[i]>a[i-1] && a[i]>a[i+1]) || (a[i]<a[i-1] && a[i]<a[i+1])) {
17. account++;
18. }
19. }
20. printf("%d", account);
22. free(a);
23. }
25. **题目：首先输入正整数n，接着输入n个正整数，如果其中存在一个数，比该数大的个数等于比该数小的个数，则输出该数，如果不存在则输出-1。**

算法思想：先用插入法排序，然后将n/2位置赋给mid，再遍历一遍数组，分别得出比mid大和比mid小的数个数，最后进行比较。算法时间复杂度T(n^2+n)。

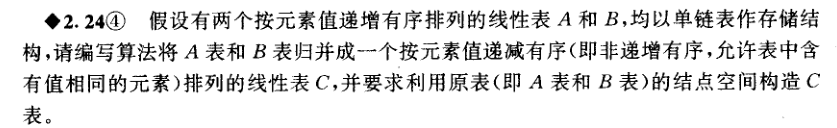
运行结果：



结果分析：正确

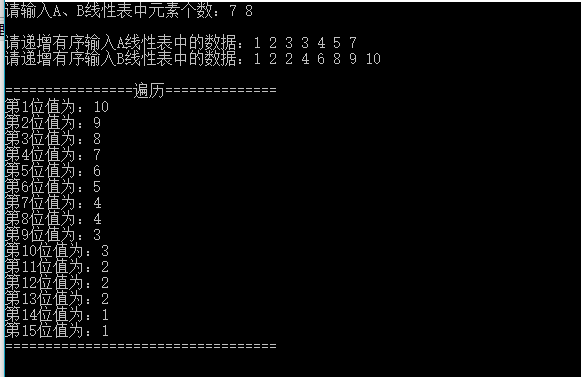
源程序：

1. #include<stdio.h>
2. #include<stdlib.h>
4. **int** main() {
6. **int** n = 0;
7. scanf("%d", &n);
8. **int** \*a = (**int**\*)malloc(**sizeof**(**int**) \* n);
9. **for**(**int** i=0; i<n; i++) {
10. scanf("%d", &a[i]);
11. }
13. **int** temp = 0;
14. **for**(**int** i=1; i<n; i++) {
15. **for**(**int** j=i; j>0 && a[j-1]>a[j]; j--) {
16. temp = a[j-1];
17. a[j-1] = a[j];
18. a[j] = temp;
19. }
20. }
22. **int** mid = a[n/2], low = 0, high = 0;
23. **for**(**int** i=0; i<n; i++) {
24. **if**(a[i] < mid) low++;
25. **else** **if**(a[i] > mid) high++;
26. }
27. **if**(low == high) printf("%d", mid);
28. **else** printf("-1");
29. }
30. **题目：**



算法思想：先将A、B线性表合并，然后在将合并后的表倒置。算法时间复杂度T((A.length+B.length)\*2)。

运行结果：



结果分析：正确

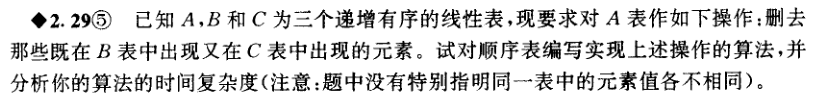
源程序：

1. #include<stdio.h>
2. #include<stdlib.h>
4. **typedef** **struct** LNode {
5. **int** data;
6. LNode \*next;
7. }\*LinkList;
9. //遍历函数
10. **void** ListTraverse(LinkList list) {
11. printf("\n================遍历==============\n");
12. LNode \*p = list->next;
13. **int** i = 1;
14. **while**(p) {
15. printf("第%d位值为：%d\n", i, p->data);
16. p = p->next;
17. i++;
18. }
19. printf("==================================\n\n");
20. };
22. **int** main() {
24. // 初始化
25. LinkList listA = (LNode\*)malloc(**sizeof**(LNode));
26. listA->next = NULL;
27. LinkList listB = (LNode\*)malloc(**sizeof**(LNode));
28. listB->next = NULL;
29. printf("请输入A、B线性表中元素个数：");
30. scanf("%d%d", &listA->data, &listB->data);
32. LNode \*p, \*temp;
33. printf("\n请递增有序输入A线性表中的数据：");
34. p = listA;
35. **for**(**int** i=0; i<listA->data; i++) {
36. temp = (LNode\*)malloc(**sizeof**(LNode));
37. scanf("%d", &temp->data);
38. p->next = temp;
39. p = p->next;
40. }
41. p->next = NULL;
42. //ListTraverse(listA);
44. printf("请递增有序输入B线性表中的数据：");
45. p = listB;
46. **for**(**int** i=0; i<listB->data; i++) {
47. temp = (LNode\*)malloc(**sizeof**(LNode));
48. scanf("%d", &temp->data);
49. p->next = temp;
50. p = p->next;
51. }
52. p->next = NULL;
53. //ListTraverse(listB);

56. // 开始递增合并操作
57. LNode \*pA = listA->next, \*pB = listB->next;
58. LNode \*priorA = listA;
59. **while**(pA && pB) {
60. **if**(pA->data > pB->data) {
61. priorA->next = pB;
62. listB->next = pB->next;
63. pB->next = pA;
64. priorA = pB;
65. pB = listB->next;
66. }
67. **else** {
68. priorA = pA;
69. pA = pA->next;
70. }
71. }
72. **if**(pB) {
73. priorA->next = pB;
74. listB->next = NULL;
75. }
76. free(listB);

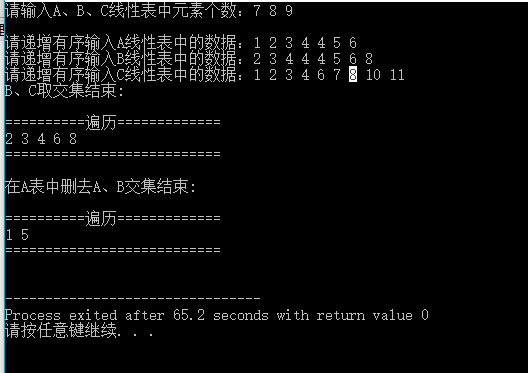
79. // 开始倒置
80. p = listA->next;
81. listA->next = NULL;
82. **while**(p) {
83. temp = p;
84. p = p->next;
85. temp->next = listA->next;
86. listA->next = temp;
87. }
88. ListTraverse(listA);

91. // 销毁链表
92. p = listA;
93. **while**(p) {
94. temp = p;
95. p = p->next;
96. free(temp);
97. }
98. }
99. **题目：**



算法思想：首先用第二题的思想求B、C的交集，然后在转换该思想由删去不同数变成删去相同数。算法时间复杂度T(A.length+B.length+C.length)。

运行结果：



结果分析：正确

源程序：

1. #include<stdio.h>
2. #include<stdlib.h>
4. **typedef** **struct**{
5. **int** \*list;
6. **int** length;
7. }SqList ;
9. **void** ListTraverse(SqList L) {
10. printf("\n==========遍历=============\n");
11. **for**(**int** i=0; i<L.length; i++) {
12. printf("%d ", L.list[i]);
13. }
14. printf("\n===========================\n\n");
15. }
17. **int** main() {
19. // 初始化
20. SqList La, Lb, Lc;
21. printf("请输入A、B、C线性表中元素个数：");
22. scanf("%d%d%d", &La.length, &Lb.length, &Lc.length);
23. La.list = (**int**\*)malloc(**sizeof**(**int**) \* La.length);
24. Lb.list = (**int**\*)malloc(**sizeof**(**int**) \* Lb.length);
25. Lc.list = (**int**\*)malloc(**sizeof**(**int**) \* Lc.length);
27. printf("\n请递增有序输入A线性表中的数据：");
28. **for**(**int** i=0; i<La.length; i++) {
29. scanf("%d", &La.list[i]);
30. }
31. //ListTraverse(La);
33. printf("请递增有序输入B线性表中的数据：");
34. **for**(**int** i=0; i<Lb.length; i++) {
35. scanf("%d", &Lb.list[i]);
36. }
37. //ListTraverse(Lb);
39. printf("请递增有序输入C线性表中的数据：");
40. **for**(**int** i=0; i<Lc.length; i++) {
41. scanf("%d", &Lc.list[i]);
42. }
43. //ListTraverse(Lc);

46. // 将B、C取交，并将交集放在B中
47. **int** i=0, j=0;
48. **while**(i<Lb.length && j<Lc.length) {
49. **if**(Lb.list[i] == Lc.list[j]) {
50. i++;
51. j++;
52. }
53. **else** **if**(Lb.list[i] < Lc.list[j]) {
54. **for**(**int** k=i; k<Lb.length-1; k++) {
55. Lb.list[k] = Lb.list[k+1];
56. }
57. Lb.length--;
58. }
59. **else** {
60. j++;
61. }
62. }
63. **if**(i < Lb.length) {
64. Lb.length = i;
65. }
66. printf("B、C取交集结束:\n");
67. ListTraverse(Lb);

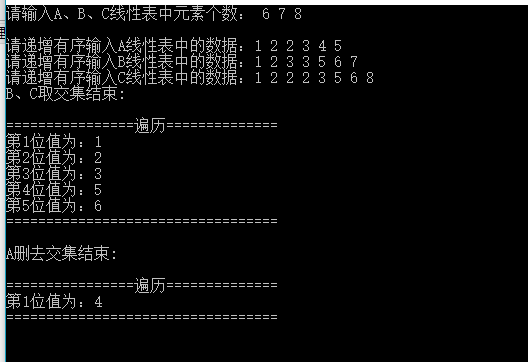
70. // 在A中，将A、B的交集删去
71. i=0, j=0;
72. **while**(i<La.length && j<Lb.length) {
73. **if**(La.list[i] == Lb.list[j]) {
74. **for**(**int** k=i; k<La.length-1; k++) {
75. La.list[k] = La.list[k+1];
76. }
77. La.length--;
78. }
79. **else** **if**(La.list[i] < Lb.list[j]) {
80. i++;
81. }
82. **else** {
83. j++;
84. }
85. }
86. printf("在A表中删去A、B交集结束:\n");
87. ListTraverse(La);

90. // 销毁线性表
91. free(La.list);
92. free(Lb.list);
93. free(Lc.list);
94. }
95. **题目：**



算法思想：算法思想同上一题，只是形式变成链式。算法时间复杂度T(A.length+B.length+C.length)。

运行结果：



结果分析：正确

源程序：

1. #include<stdio.h>
2. #include<stdlib.h>
4. **typedef** **struct** LNode {
5. **int** data;
6. LNode \*next;
7. }\*LinkList;
9. //遍历函数
10. **void** ListTraverse(LinkList list) {
11. printf("\n================遍历==============\n");
12. LNode \*p = list->next;
13. **int** i = 1;
14. **while**(p) {
15. printf("第%d位值为：%d\n", i, p->data);
16. p = p->next;
17. i++;
18. }
19. printf("==================================\n\n");
20. };
22. **int** main() {
24. // 初始化
25. LinkList listA = (LNode\*)malloc(**sizeof**(LNode));
26. listA->next = NULL;
27. LinkList listB = (LNode\*)malloc(**sizeof**(LNode));
28. listB->next = NULL;
29. LinkList listC = (LNode\*)malloc(**sizeof**(LNode));
30. listC->next = NULL;
31. printf("请输入A、B、C线性表中元素个数：");
32. scanf("%d%d%d", &listA->data, &listB->data, &listC->data);
34. LNode \*p, \*temp;
35. printf("\n请递增有序输入A线性表中的数据：");
36. p = listA;
37. **for**(**int** i=0; i<listA->data; i++) {
38. temp = (LNode\*)malloc(**sizeof**(LNode));
39. scanf("%d", &temp->data);
40. p->next = temp;
41. p = p->next;
42. }
43. p->next = NULL;
44. //ListTraverse(listA);
46. printf("请递增有序输入B线性表中的数据：");
47. p = listB;
48. **for**(**int** i=0; i<listB->data; i++) {
49. temp = (LNode\*)malloc(**sizeof**(LNode));
50. scanf("%d", &temp->data);
51. p->next = temp;
52. p = p->next;
53. }
54. p->next = NULL;
55. //ListTraverse(listB);
57. printf("请递增有序输入C线性表中的数据：");
58. p = listC;
59. **for**(**int** i=0; i<listC->data; i++) {
60. temp = (LNode\*)malloc(**sizeof**(LNode));
61. scanf("%d", &temp->data);
62. p->next = temp;
63. p = p->next;
64. }
65. p->next = NULL;
66. //ListTraverse(listC);

69. // 将B、C取交集，并将结果放在B中
70. LNode \*pB = listB->next, \*pC = listC->next;
71. LNode \*priorB = listB, \*priorC = listC;
72. **while**(pC && pB) {
73. **if**(pB->data == pC->data) {
74. priorB = pB;
75. pB = priorB->next;
76. priorC = pC;
77. pC = priorC->next;
78. }
79. **else** **if**(pB->data < pC->data) {
80. priorB->next = pB->next;
81. free(pB);
82. pB = priorB->next;
83. }
84. **else** {
85. priorC = pC;
86. pC = priorC->next;
87. }
88. }
89. **while**(pB) {
90. temp = pB;
91. pB = pB->next;
92. free(pB);
93. }
94. priorB->next = NULL;
95. printf("B、C取交集结束:\n");
96. ListTraverse(listB);

99. // 在A中，删去存在于B、C交集中的数
100. LNode \*pA = listA->next, \*priorA = listA;
101. pB = listB->next, priorB = listB;
102. **while**(pA && pB) {
103. **if**(pA->data == pB->data) {
104. priorA->next = pA->next;
105. free(pA);
106. pA = priorA->next;
107. }
108. **else** **if**(pA->data > pB->data) {
109. priorB = pB;
110. pB = priorB->next;
111. }
112. **else** {
113. priorA = pA;
114. pA = priorA->next;
115. }
116. }
117. printf("A删去交集结束:\n");
118. ListTraverse(listA);

121. // 销毁链表
122. p = listA;
123. **while**(p) {
124. temp = p;
125. p = p->next;
126. free(temp);
127. }
128. p = listB;
129. **while**(p) {
130. temp = p;
131. p = p->next;
132. free(temp);
133. }
134. p = listC;
135. **while**(p) {
136. temp = p;
137. p = p->next;
138. free(temp);
139. }
140. }