

《数据结构》上机实验报告

第 1 次上机

学号：

姓名：

学院： 信息科学学院

专业： 计算机科学

教师：

日期： 2018.9.12

1. 实验要求

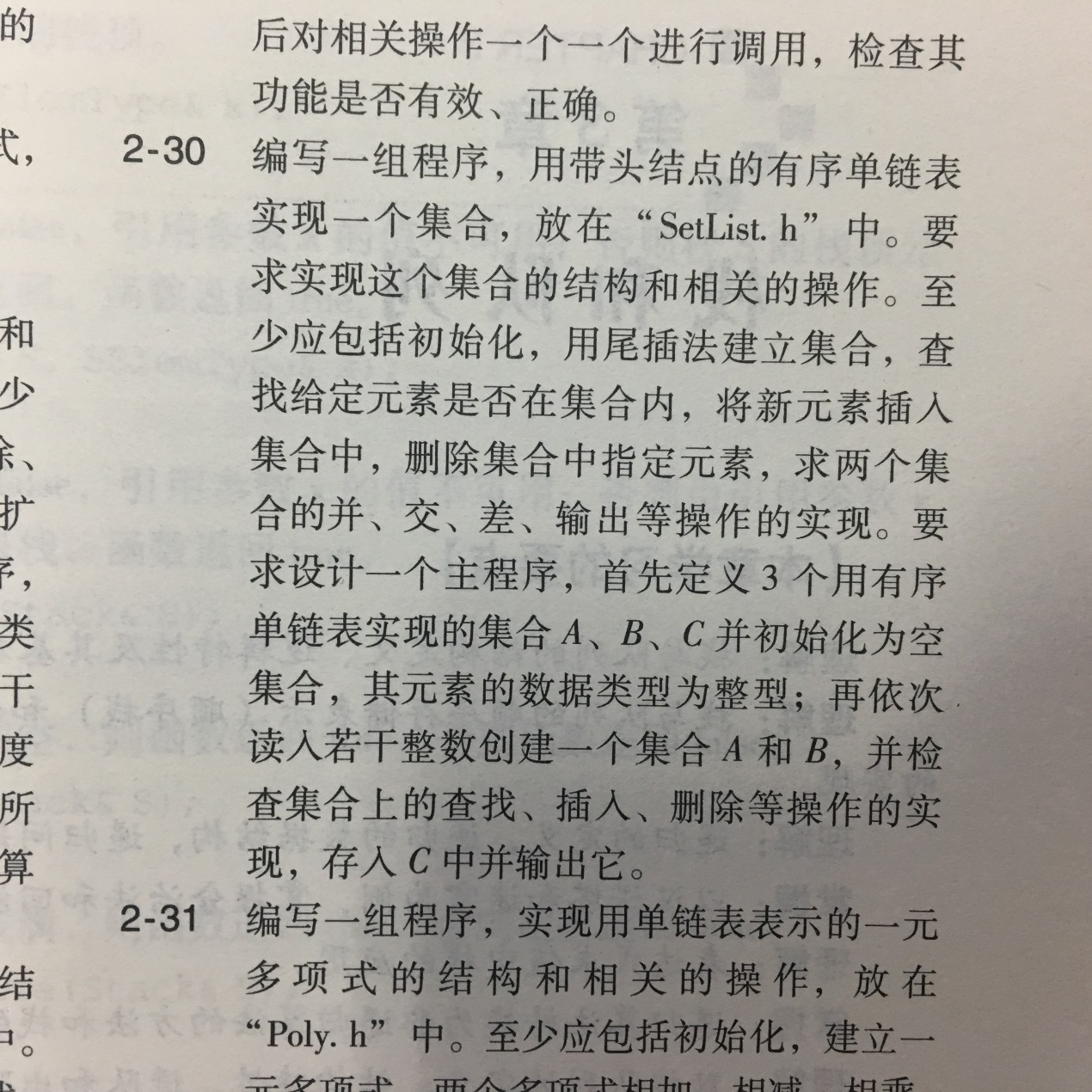
1.上机之前应做好充分准备，认真思考所需的上机题目，提高上机效率。

2.独立上机输入和调试自己所编的程序，切忌抄袭、拷贝他人程序。

3.上机结束后，整理出实验报告。书写报告时，重点放在实验的方法、思路以及总结反思上，以达到巩固课堂学习、提高动手能力的目的。

1. 实验内容

课本练习2-30



1. 实验步骤（写出问题分析或者算法思路）

思路：使用带头节点的链表实现数列是很方便的，而有序集合是没有重复项的有序数列，所以在插入、求集合的并时要注意集合的元素唯一性。

课上和ppt里已经有关于单链表初始化、查找、定位、后插法建表、输出、删除的思想方法和具体实现，这次也借鉴了很多。分别见InitList, Search, Locate, insertRear,Output和Remove函数。注意Remove函数中引用了Locate函数。

集合的单元素插入是这样实现的：先判断输入元素是否已经在集合里了，然后遍历找到输入元素应该插入位置的前一个节点pre，然后使用链表的一般插入方法，在更改两个指针值后，就插入了新的节点。见insetOne函数。

集合的并，思路也比较简单，先插入集合1中的所有不重复元素，然后遍历集合2中的元素，选不重复的元素进行插入新集合中。见Union函数。这个算法是O(n1\*n2)的，可以优化成O(n1+n2)，因为是集合是有序的。

集合的交，首先遍历集合1中的元素，用search函数看它是否也在集合2中，然后再将其插入到新集合里。见Intersection函数。

集合的差，与集合的交类似，首先遍历集合1中的元素，用search函数看它是否不在集合2中，然后再将其插入到新集合里。见Minus函数。

1. 程序清单（源程序代码等）

Seqlist.h:

1. #include<malloc.h>
2. #include<stdio.h>
3. #include<iostream>
4. #define INF ((1<<31)-1)
5. **using** **namespace** std;
6. **typedef** **int** DataType;   //元素的数据类型
8. //typedef char DataType;
9. **typedef** **struct** node
10. {                //链表结点
11. DataType data;                     //结点数据域
12. **struct** node \* link;                 //结点链域
13. } LinkNode, \*LinkList;           //链头指针
14. LinkList first;
15. **void** InitList (LinkList & first)
16. {
17. first=(LinkNode\*)malloc(**sizeof**(LinkNode));
18. first->link=NULL;
19. **return**;
20. }
21. **void** makeEmpty(LinkList &first)
22. {
23. LinkList q;
24. **while** ( first->link != NULL )
25. {   //链不空时
26. q = first->link;     //q指示首元结点
27. first->link = q->link;    //摘下q指示结点
28. free (q);                   //释放它
29. }
30. first->link=NULL;
31. **return**;
32. }
33. LinkNode \*Search ( LinkList first, DataType x )
34. {
35. //在链表中从头搜索其数据值为 x 的结点
36. LinkNode \*p= first->link; //p为检测指针
37. **while** ( p != NULL && p->data != x )
38. p = p->link;
39. **return** p;
40. }
42. LinkNode \*Locate ( LinkList first, **int** i )
43. {
44. //返回表中第 i 个元素的地址,  头结点为 0 号
45. **if** ( i < 0 ) **return** NULL;           // i 值不合理
46. LinkNode \* p = first;   **int** k = 0;
47. **while** ( p != NULL && k < i )
48. { p = p->link;  k++; }       //找第 i 个结点
49. **return** p;
50. //当返回地址非空则为第 i 个结点地址
51. //否则返回NULL
52. }
54. **void** insertRear ( LinkList& first, DataType endTag )    //尾插法，当输入为ndTag时停止插入
55. {
56. DataType val;
57. LinkNode \*s, \*rear = first; //rear指向表尾
58. scanf ( "%d", &val );       //读入一数据
59. **while** ( val != endTag )
60. {
61. /\*if(Search(first,val)!=NULL) //重复的话就不能插入了
62. {
63. scanf("%d",&val);       //读入下一数据
64. continue;
65. }\*/
66. s=(LinkNode\*)malloc(**sizeof**(LinkNode));
67. s->data = val;       //创建新结点并赋值
68. rear->link = s;
69. rear = s;   //插入到表尾
70. scanf ( "%d", &val );       //读入下一数据
72. }
73. rear->link = NULL;   //表收尾
74. }
76. **void** insertOne ( LinkList& first,DataType val )
77. {
78. **if**(Search(first,val)!=NULL) //重复的话就不能插入了
79. **return**;
80. LinkNode \*s;
81. s = (LinkNode \*) malloc ( **sizeof** (LinkNode ));
82. s->data = val;                   //创建新结点
83. LinkNode \*pre;
84. pre=first->link;
85. **if**(pre==NULL)
86. {
87. s->link = first->link;    //插入到表前端
88. first->link = s;
89. **return**;
90. }
91. **while**(pre->link!=NULL && pre->link->data < val)
92. pre=pre->link;
93. s->link=pre->link;
94. pre->link=s;
95. **return** ;
96. }
98. **bool** Remove ( LinkList& first, **int** i, DataType& x )
99. {
100. //在表中删除第 i 个元素，通过 x 返回其值
101. LinkNode \*pre=Locate(first,i-1);
102. **if**(pre==NULL || pre->link==NULL)
103. **return** **false**;
104. **else**
105. pre->link=pre->link->link;
107. **return** **true**;
108. }
110. **void** Union(LinkList &L1, LinkList &L2,LinkList &L3)  //并 保存在L3中
111. {
112. InitList(L3);
113. LinkList p;
114. p=L1->link;
115. **while**(p!=NULL)
116. {
117. **if**(Search(L3,p->data)==NULL)
118. insertOne(L3,p->data);
119. p=p->link;
120. }
121. p=L2->link;
122. **while**(p!=NULL)
123. {
124. **if**(Search(L3,p->data)==NULL)
125. insertOne(L3,p->data);
126. p=p->link;
127. }
128. **return**;
129. }
131. **void** Intersection(LinkList &L1, LinkList &L2,LinkList &L3)  //交 保存在L3中
132. {
133. InitList(L3);
134. LinkList p;
135. p=L1->link;
136. **while**(p!=NULL)
137. {
138. **if**(Search(L2,p->data)!=NULL)
139. insertOne(L3,p->data);
140. p=p->link;
141. }
142. **return**;
143. }
144. **void** Minus(LinkList &L1, LinkList &L2,LinkList &L3)  //差L1-L2 保存在L3中
145. {
146. InitList(L3);
147. LinkList p;
148. p=L1->link;
149. **while**(p!=NULL)
150. {
151. **if**(Search(L2,p->data)==NULL)
152. insertOne(L3,p->data);
153. p=p->link;
154. }
155. **return**;
156. }
157. **void** Output(LinkList &first)
158. {
159. LinkNode \*p=first;
160. **if**(first->link==NULL)
161. printf("The set is empty\n");
162. **else**
163. {
164. p=first->link;
165. **while**(p)
166. {
167. printf("%d ",p->data);
168. p=p->link;
169. }
170. printf("\n");
171. }
172. **return**;
173. }
174. **void** printList ( LinkList& first ) {    //输出链表
175. **for** ( LinkNode \*p = first; p; p = p->link )
176. printf ( "%d ", p->data );
177. printf ( "\n" );
178. }

main.cpp:

1. #include "SetList.h"
2. #include<iostream>
3. #include <stdio.h>
4. #include <fstream>
5. **using** **namespace** std;
6. **int** main ()
7. {
8. freopen("test.txt","r",stdin);
9. LinkList A, B, C, D;
10. InitList(A);
11. InitList(B);
12. InitList(C);
13. LinkNode \*tmp;
14. **int** endtag,tem;
15. scanf("%d",&endtag);
17. insertRear(A,endtag);
18. insertRear(B,endtag);
19. insertRear(C,endtag);
21. printf("输入的A,B,C分别是:\n");
22. Output(A);
23. Output(B);
24. Output(C);
26. printf("在A中查找1：\t");
27. tmp=Search(A,1);
28. **if**(tmp!=NULL)printf("%d\n",tmp->data);
29. **else** printf("没有找到%d\n",1);
31. printf("在B中查找1：\t");
32. tmp=Search(B,1);
33. **if**(tmp!=NULL)printf("%d\n",tmp->data);
34. **else** printf("没有找到%d\n",1);
36. printf("A与B的并：\t");
37. Union(A,B,D);
38. Output(D);
40. printf("B与C的并：\t");
41. Union(B,C,D);
42. Output(D);
44. printf("B与C的差（B-C）：\t");
45. Minus(B,C,D);
46. Output(D);
48. printf("C与A的差（C-A）：\t");
49. Minus(C,A,D);
50. Output(D);
52. printf("A与A的差（A-A）：\t");
53. Minus(A,A,D);
54. Output(D);

57. printf("B与C的交：\t");
58. Intersection(B,C,D);
59. Output(D);
61. printf("A与C的交：\t");
62. Intersection(A,C,D);
63. Output(D);
65. printf("在A中删除第1个元素：\t");
66. Remove(A,1,tem);
67. Output(A);
69. printf("在A中有序插入 4 和 6：\t");
70. insertOne(A,4);
71. insertOne(A,6);
72. Output(A);
74. printf("在B中有序插入 3 和 9：\t");
75. insertOne(B,3);
76. insertOne(B,9);
77. Output(B);
79. printf("在B中删除第2个和第3个元素：\t");
80. Remove(B,2,tem);
81. Remove(B,2,tem);    //不是3！
82. Output(B);
84. **return** 0;
85. }

test.txt:

-1

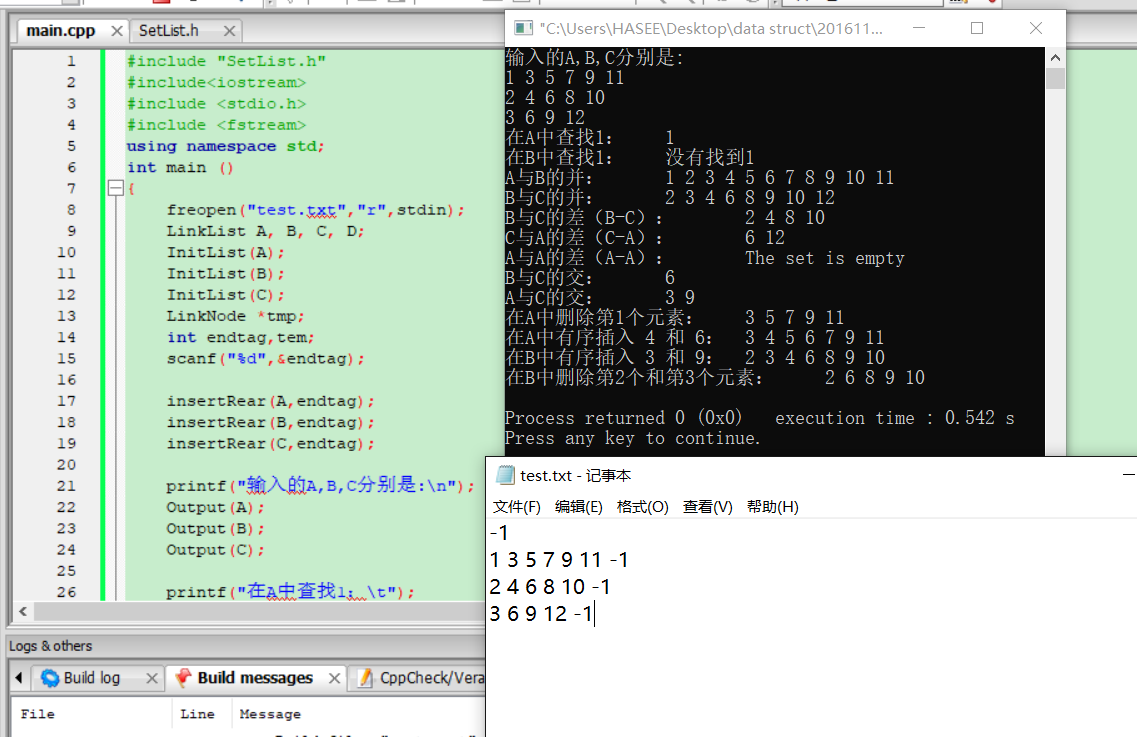
1 3 5 7 9 11 -1

2 4 6 8 10 -1

3 6 9 12 -1

1. 运行结果（程序运行时的结果说明或运行截图等）

使用了freopen函数来读入测试数据“tset.txt”。



1. 总结（实验中遇到的问题、取得的经验、感想等）

1 使用引用型参数&可以直接对原变量（指针等）进行修改。

引用型参数“&”是把形参 first 看作是实际变量（一个指针）的别名，在函数体内对 first 的操作将直接对实际变量的操作。

2 单链表的链首插入、删除、扩充都是O(1)的。但查找、定位是O（n）的，可以用在经常需要链首插入、删除和修改的地方，比如栈等。

3 在集合的并中，可以考虑运用其有序性来加快运行速度。

4 链表与顺序表对空间的要求是不同的，顺序表需要的是一整块连续的空间，而链表则不需要连续的空间，它只需要记录下后续节点的地址就可以了。

5 每一次读入新数据前，一定要进行初始化。