**数据结构实验报告—二叉排序树基本操作的实现**

设计者姓名：张帆

设计者班级：2班

设计者学号：20192131077

上机环境：DEV-C++

设计日期：2021-01-01

1. **实验题目**

必做内容一：

选择二叉链式存储结构作为二叉排序树的存储结构，设计一个程序实现二叉

排序树的基本操作（包括：建立、插入、删除、查找等）

程序的菜单功能项如下：

1——建立一棵二叉排序树

2——在二叉排序树中插入一个值（递归算法）

3——在二叉排序树中插入一个值（非递归算法）

4——在二叉排序树中查找一个值（递归算法）

5——在二叉排序树中查找一个值（非递归算法）

6——在二叉排序树中删除一个值

7——输出二叉树

8——退出

必做内容二：

实现学生通讯录管理的几个操作功能（新建、插入、删除、从文件读取、写 入文件和查询、屏幕输出等功能）。通讯录中学生的信息有学号、姓名、出生日

期、性别、电话和地址等。

利用二叉链式（二叉排序树）存储结构来实现

系统的菜单功能项如下：

1——新建学生通讯录

2——向学生通讯录插入学生信息

3——在通讯录删除学生信息

4——从文件中读取通讯录信息

5——向文件写入学生通讯录信息

6——在通讯录中查询学生信息

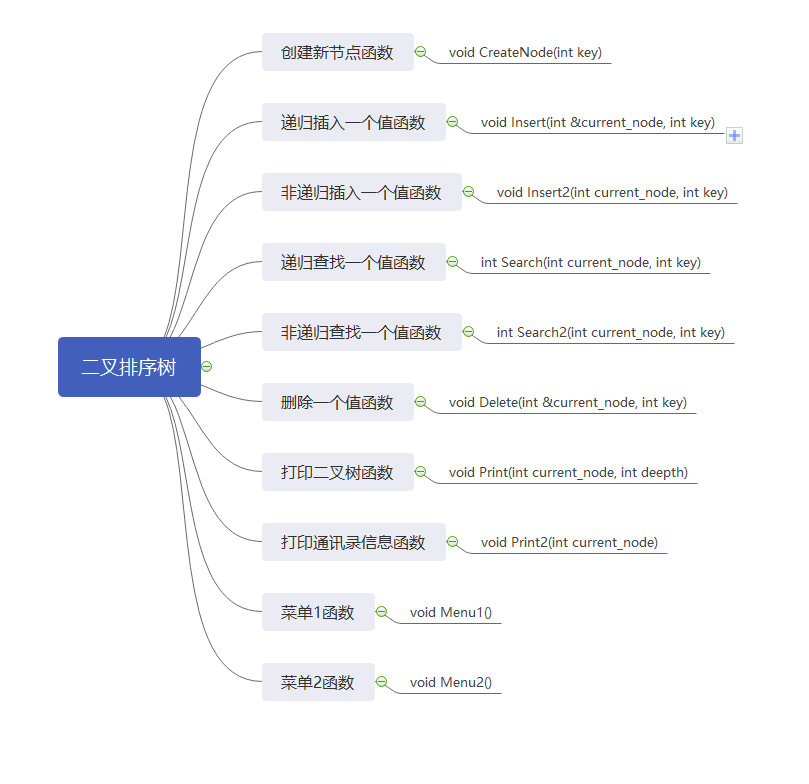
7——在屏幕中输出全部学生信息

8——退出

1. **实验项目目的**

使学生掌握二叉排序树的基本操作，熟练使用二叉排序树进行操作

1. **实验项目的程序结构**



**四、实验项目包含的各个文件中的函数的功能描述**

1. #include <iostream>
2. #include <cstdio>
3. #include <cstdlib>
4. #include <cstring>
5. #include <fstream>
6. #include <algorithm>
7. **using** **namespace** std;
8. **const** **int** NodeNumber = 100;  //结点个数
10. **int** root, idx;  //根结点，结点指针
12. **class** Node
13. {
14. **private**:
15. string sex, name, address, phone\_number;  // 姓名、性别、电话、地址
17. **struct** Birthday  //出生日期
18. {
19. **int** year, month, day;
20. **friend** istream &operator >> (istream &in, Birthday &x)
21. {
22. in >> x.year >> x.month >> x.day;
23. **return** in;
24. }
25. **friend** ostream &operator << (ostream &out, **const** Birthday &x)
26. {
27. out << x.year << "-" << x.month << "-" << x.day;
28. **return** out;
29. }
30. }birthday;
32. **public**:
33. **int** leftChild, rightChild;  //左右孩子
34. **int** key;  //值（学号）
35. **int** cnt;  //相同值（学号）的个数
36. Node() {}
37. ~Node() {}
38. **void** Input()
39. {
40. cout << "请输入学号：";
41. cin >> key;
42. cout << key << endl;
43. cout << "请输入姓名：";
44. cin >> name;
45. cout << name << endl;
46. cout << "请输入性别：";
47. cin >> sex;
48. cout << sex << endl;
49. cout << "请输入电话：";
50. cin >> phone\_number;
51. cout << phone\_number << endl;
52. cout << "请输入地址：";
53. cin >> address;
54. cout << address << endl;
55. cout << "请输入出生日期：(格式yy mm dd)";
56. cin >> birthday;
57. cout << birthday << endl;
58. cout << "信息录入完毕！" << endl;
59. }
61. **void** Output()
62. {
63. cout << "学号：" << key << " 姓名：" << name << " 性别：" << sex << endl;
64. cout << "电话：" << phone\_number << " 地址：" << address <<  " 出生日期：" << birthday << endl;
65. }
67. Node operator=(**const** Node &x)
68. {
69. name = x.name, sex = x.sex, phone\_number = x.phone\_number, address = x.address;
70. birthday.year = x.birthday.year, birthday.month = x.birthday.month, birthday.day = x.birthday.day;
71. key = x.key, cnt = x.cnt;
72. **return** \***this**;
73. }
74. }BinarySortTree[NodeNumber];

77. **void** CreateNode(**int** key)  //创建一个新结点
78. {
79. BinarySortTree[++idx].key = key;
80. BinarySortTree[idx].cnt = 1;
81. }

84. **void** Insert(**int** ¤t\_node, **int** key)  //在二叉搜索树中递归插入一个值
85. {
86. **if** (current\_node == 0)
87. {
88. CreateNode(key);
89. current\_node = idx;
90. }
91. **else** **if** (BinarySortTree[current\_node].key == key)  BinarySortTree[current\_node].cnt++;
92. **else** **if** (BinarySortTree[current\_node].key > key)  Insert(BinarySortTree[current\_node].leftChild, key);
93. **else** Insert(BinarySortTree[current\_node].rightChild, key);
94. }

97. **void** Insert2(**int** current\_node, **int** key)  //在二叉搜索树中非递归插入一个值
98. {
99. **while** (1)
100. {
101. **if** (BinarySortTree[current\_node].key == key)
102. {
103. BinarySortTree[current\_node].cnt++;
104. **return**;
105. }
106. **else** **if** (BinarySortTree[current\_node].key > key)
107. {
108. **if** (BinarySortTree[current\_node].leftChild == 0)
109. {
110. CreateNode(key);
111. BinarySortTree[current\_node].leftChild = idx;
112. **return**;
113. }
114. **else** current\_node = BinarySortTree[current\_node].leftChild;
115. }
116. **else**
117. {
118. **if** (BinarySortTree[current\_node].rightChild == 0)
119. {
120. CreateNode(key);
121. BinarySortTree[current\_node].rightChild = idx;
122. **return**;
123. }
124. **else** current\_node = BinarySortTree[current\_node].rightChild;
125. }
126. }
127. }

130. **int** Search(**int** current\_node, **int** key)  //在二叉搜索树中递归搜索一个值
131. {
132. **if** (current\_node == 0)  **return** 0;
133. **if** (BinarySortTree[current\_node].key == key) **return** current\_node;
134. **else** **if** (BinarySortTree[current\_node].key > key) **return** Search(BinarySortTree[current\_node].leftChild, key);
135. **else** **return** Search(BinarySortTree[current\_node].rightChild, key);
136. }

139. **int** Search2(**int** current\_node, **int** key)  //在二叉搜索树中非递归搜索一个值
140. {
141. **while** (1)
142. {
143. **if** (BinarySortTree[current\_node].key == key) **return** current\_node;
144. **else** **if** (BinarySortTree[current\_node].key > key)
145. {
146. **if** (BinarySortTree[current\_node].leftChild == 0) **return** 0;
147. **else** current\_node = BinarySortTree[current\_node].leftChild;
148. }
149. **else**
150. {
151. **if** (BinarySortTree[current\_node].rightChild == 0) **return** 0;
152. **else** current\_node = BinarySortTree[current\_node].rightChild;
153. }
154. }
155. }

158. **void** Delete(**int** ¤t\_node, **int** key)  //在二叉搜索树中删除一个值
159. {
160. **if** (current\_node == 0) **return**;
161. **if** (BinarySortTree[current\_node].key == key)
162. {
163. **if** (BinarySortTree[current\_node].cnt > 1)
164. {
165. BinarySortTree[current\_node].cnt--;
166. **return**;
167. }**else** **if** (BinarySortTree[current\_node].leftChild || BinarySortTree[current\_node].rightChild)
168. {
169. **if** (BinarySortTree[current\_node].leftChild && BinarySortTree[current\_node].rightChild)
170. {
171. **int** temp = BinarySortTree[current\_node].rightChild;
172. **while** (BinarySortTree[temp].leftChild != 0) temp = BinarySortTree[temp].leftChild;
173. BinarySortTree[current\_node] = BinarySortTree[temp];
174. Delete(BinarySortTree[current\_node].rightChild, BinarySortTree[temp].key);
175. }
176. **else** **if** (BinarySortTree[current\_node].leftChild) current\_node = BinarySortTree[current\_node].leftChild;
177. **else** current\_node = BinarySortTree[current\_node].rightChild;
178. }       **else**
179. {
180. current\_node = 0;
181. }
182. }
183. **else** **if** (BinarySortTree[current\_node].key > key) Delete(BinarySortTree[current\_node].leftChild, key);
184. **else** Delete(BinarySortTree[current\_node].rightChild, key);
185. }

188. **void** Print(**int** current\_node, **int** deepth)  //凹入表打印树
189. {
190. **for** (**int** i = 0; i < deepth; i++) cout << "     ";
191. cout << BinarySortTree[current\_node].key << "(" << BinarySortTree[current\_node].cnt << ")" << endl;
192. **if** (BinarySortTree[current\_node].leftChild != 0) Print(BinarySortTree[current\_node].leftChild, deepth + 1);
193. **if** (BinarySortTree[current\_node].rightChild != 0) Print(BinarySortTree[current\_node].rightChild, deepth + 1);
194. }

197. **void** Print2(**int** current\_node)  //打印通讯录信息
198. {
199. BinarySortTree[current\_node].Output();
200. **if** (BinarySortTree[current\_node].leftChild != 0) Print2(BinarySortTree[current\_node].leftChild);
201. **if** (BinarySortTree[current\_node].rightChild != 0) Print2(BinarySortTree[current\_node].rightChild);
202. }

205. **void** Menu1()  //必做内容一功能菜单
206. {
207. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
208. cout << "\*                1、建立一棵二叉排序树                \*" << endl;
209. cout << "\*         2、在二叉排序树中插入一个值（递归算法）     \*" << endl;
210. cout << "\*        3、在二叉排序树中插入一个值（非递归算法）    \*" << endl;
211. cout << "\*         4、在二叉排序树中查找一个值（递归算法）     \*" << endl;
212. cout << "\*        5、在二叉排序树中查找一个值（非递归算法）    \*" << endl;
213. cout << "\*             6、在二叉排序树中删除一个值             \*" << endl;
214. cout << "\*                  7、输出二叉树                      \*" << endl;
215. cout << "\*                     8、退出                         \*" << endl;
216. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
217. cout << "请输入操作指令：" << endl;
218. }

221. **void** FirstControl()  //必做内容一：
222. {
223. **while** (1)
224. {
225. Menu1();
226. //指令，结点数量，待查找的值，搜索位置，删除的结点，插入的结点
227. **int** operator\_index, node\_num, find\_element, position, delete\_element, insert\_node;
228. cin >> operator\_index;
229. cout << operator\_index << endl;
230. **switch** (operator\_index)
231. {
232. **case** 1:
233. cout << "请输入二叉排序树的结点个数：";
234. cin >> node\_num;
235. cout << node\_num << endl;
236. **for** (**int** i = 0; i < node\_num; i++)
237. {
238. **int** key;
239. cin >> key;
240. cout << key << " ";
241. Insert(root, key);
242. }
243. cout << endl;
244. **break**;
245. **case** 2:
246. cout << "插入前的二叉搜索树：" << endl;
247. Print(root, 0);
248. cout << "请输入你要插入的结点：";
249. cin >> insert\_node;
250. cout << insert\_node << endl;
251. Insert(root, insert\_node);
252. cout << "插入后的二叉搜索树：" << endl;
253. Print(root, 0);
254. **break**;
255. **case** 3:
256. cout << "插入前的二叉搜索树：" << endl;
257. Print(root, 0);
258. cout << "请输入你要插入的结点：";
259. cin >> insert\_node;
260. cout << insert\_node << endl;
261. Insert2(root, insert\_node);
262. cout << "插入后的二叉搜索树：" << endl;
263. Print(root, 0);
264. **break**;
265. **case** 4:
266. cout << "输入你要查找的结点：";
267. cin >> find\_element;
268. cout << find\_element << endl;
269. position = Search(root, find\_element);
270. **if** (position) cout << "搜索成功！编号为" << position << endl;
271. **else** cout << "搜索失败！" << endl;
272. **break**;
273. **case** 5:
274. cout << "输入你要查找的结点：";
275. cin >> find\_element;
276. cout << find\_element << endl;
277. position = Search2(root, find\_element);
278. **if** (position) cout << "搜索成功！编号为" << position << endl;
279. **else** cout << "搜索失败！" << endl;
280. **break**;
281. **case** 6:
282. cout << "删除前的二叉搜索树：" << endl;
283. Print(root, 0);
284. cout << "请输入你要删除的结点：";
285. cin >> delete\_element;
286. cout << delete\_element << endl;
287. Delete(root, delete\_element);
288. cout << "删除后的二叉搜索树：" << endl;
289. Print(root, 0);
290. **break**;
291. **case** 7:
292. Print(root, 0);
293. **break**;
294. **case** 8:
295. **return**;
296. **default**:
297. cout << operator\_index << "不是内部指令！" << endl;
298. }
299. cout << endl;
300. }
301. }

304. **void** Menu2()  //必做内容二功能菜单
305. {
306. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
307. cout << "\*         1、新建学生通讯录          \*" << endl;
308. cout << "\*     2、向学生通讯录插入学生信息    \*" << endl;
309. cout << "\*       3、在通讯录删除学生信息      \*" << endl;
310. cout << "\*      4、从文件中读取通讯录信息     \*" << endl;
311. cout << "\*     5、向文件写入学生通讯录信息    \*" << endl;
312. cout << "\*      6、在通讯录中查询学生信息     \*" << endl;
313. cout << "\*     7、在屏幕中输出全部学生信息    \*" << endl;
314. cout << "\*              8、退出               \*" << endl;
315. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
316. cout << "请输入操作指令：" << endl;
317. }

320. **void** SecondControl()  //必做内容二：
321. {
322. **while** (1)
323. {
324. Menu2();
325. **int** operator\_index, student\_num, delete\_student, find\_student;
326. cin >> operator\_index;
327. cout << operator\_index << endl;
328. **switch** (operator\_index)
329. {
330. **case** 1:
331. cout << "请输入添加学生的个数：";
332. cin >> student\_num; cout << student\_num << endl;
333. **for** (**int** i = 0; i < student\_num; i++)
334. {
335. BinarySortTree[idx + 1].Input();
336. Insert(root, BinarySortTree[idx + 1].key);
337. }
338. cout << "通讯录创建完毕！" << endl;
339. **break**;
340. **case** 2:
341. cout << "请输入你要插入学生的信息：" << endl;
342. BinarySortTree[idx + 1].Input();
343. Insert(root, BinarySortTree[idx + 1].key);
344. cout << "插入成功！" << endl;
345. **break**;
346. **case** 3:
347. cout << "请输入你要删除学生的学号：";
348. cin >> delete\_student; cout << delete\_student << endl;
349. Delete(root, delete\_student);
350. cout << "删除完毕！" << endl;
351. **break**;
352. **case** 4:
353. cout << "通讯录信息已读取成功！" << endl;
354. **break**;
355. **case** 5:
356. cout << "通讯录信息已写入文件中！" << endl;
357. **break**;
358. **case** 6:
359. cout << "请输入你要查找的学生的学号：";
360. cin >> find\_student; cout << find\_student << endl;
361. find\_student = Search(root, find\_student);
362. **if** (find\_student)
363. {
364. cout << "查找成功！" << endl;
365. BinarySortTree[find\_student].Output();
366. } **else** cout << "查找失败！" << endl;
367. **break**;
368. **case** 7:
369. cout << "通讯录中全部的学生信息：" << endl;
370. Print2(root);
371. **break**;
372. **case** 8:
373. **return**;
374. **default**:
375. cout << operator\_index << "不是内部指令！" << endl;
376. }
377. cout << endl;
378. }
379. }
381. **int** main(**int** argc, **char**\*\* argv)
382. {
383. freopen("in.txt", "r", stdin);
384. freopen("out.txt", "w", stdout);
385. FirstControl();
386. **for** (**int** i = 1; i <= idx; i++) BinarySortTree[i].leftChild = BinarySortTree[i].rightChild = 0;
387. root = 0, idx = 0;
388. SecondControl();
389. **return** 0;
390. }

**五、算法描述或流程图**

二叉排序树性质：

若它的左子树不空，则左子树上所有节点的值均小于它的根节点的值

若它的右子树不空，则右子树上所有节点的值均大于其根节点的值

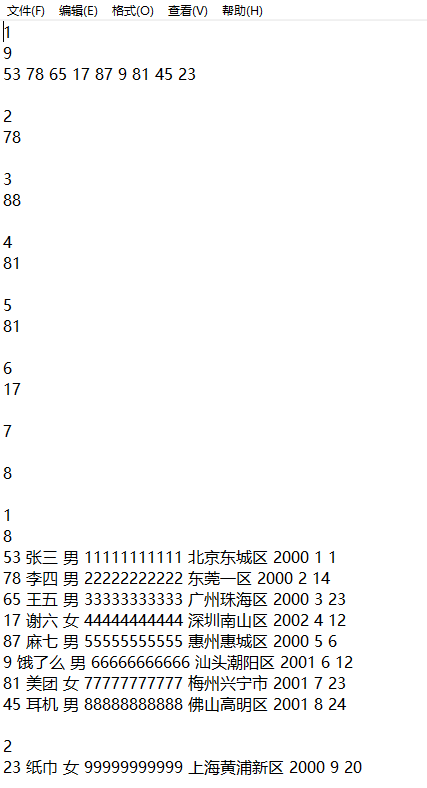
插入新元素时，可以从根节点开始，遇键值较大者就向左，遇键值较小者就向右，一直到末端，就是插入点

对于二叉排序树中的节点A，对它的删除分为两种情况：

1. 如果A只有一个子节点，就直接将A的子节点连至A的父节点上，并将A删除；
2. 如果A有两个子节点，我们就以右子树内的最小节点取代A

**六、实验数据和实验结果分析**

**输入文件：**





**输出文件：**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、建立一棵二叉排序树 \*

\* 2、在二叉排序树中插入一个值（递归算法） \*

\* 3、在二叉排序树中插入一个值（非递归算法） \*

\* 4、在二叉排序树中查找一个值（递归算法） \*

\* 5、在二叉排序树中查找一个值（非递归算法） \*

\* 6、在二叉排序树中删除一个值 \*

\* 7、输出二叉树 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

1

请输入二叉排序树的结点个数：9

53 78 65 17 87 9 81 45 23

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、建立一棵二叉排序树 \*

\* 2、在二叉排序树中插入一个值（递归算法） \*

\* 3、在二叉排序树中插入一个值（非递归算法） \*

\* 4、在二叉排序树中查找一个值（递归算法） \*

\* 5、在二叉排序树中查找一个值（非递归算法） \*

\* 6、在二叉排序树中删除一个值 \*

\* 7、输出二叉树 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

2

插入前的二叉搜索树：

53(1)

17(1)

9(1)

45(1)

23(1)

78(1)

65(1)

87(1)

81(1)

请输入你要插入的结点：78

插入后的二叉搜索树：

53(1)

17(1)

9(1)

45(1)

23(1)

78(2)

65(1)

87(1)

81(1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、建立一棵二叉排序树 \*

\* 2、在二叉排序树中插入一个值（递归算法） \*

\* 3、在二叉排序树中插入一个值（非递归算法） \*

\* 4、在二叉排序树中查找一个值（递归算法） \*

\* 5、在二叉排序树中查找一个值（非递归算法） \*

\* 6、在二叉排序树中删除一个值 \*

\* 7、输出二叉树 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

3

插入前的二叉搜索树：

53(1)

17(1)

9(1)

45(1)

23(1)

78(2)

65(1)

87(1)

81(1)

请输入你要插入的结点：88

插入后的二叉搜索树：

53(1)

17(1)

9(1)

45(1)

23(1)

78(2)

65(1)

87(1)

81(1)

88(1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、建立一棵二叉排序树 \*

\* 2、在二叉排序树中插入一个值（递归算法） \*

\* 3、在二叉排序树中插入一个值（非递归算法） \*

\* 4、在二叉排序树中查找一个值（递归算法） \*

\* 5、在二叉排序树中查找一个值（非递归算法） \*

\* 6、在二叉排序树中删除一个值 \*

\* 7、输出二叉树 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

4

输入你要查找的结点：81

搜索成功！编号为7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、建立一棵二叉排序树 \*

\* 2、在二叉排序树中插入一个值（递归算法） \*

\* 3、在二叉排序树中插入一个值（非递归算法） \*

\* 4、在二叉排序树中查找一个值（递归算法） \*

\* 5、在二叉排序树中查找一个值（非递归算法） \*

\* 6、在二叉排序树中删除一个值 \*

\* 7、输出二叉树 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

5

输入你要查找的结点：81

搜索成功！编号为7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、建立一棵二叉排序树 \*

\* 2、在二叉排序树中插入一个值（递归算法） \*

\* 3、在二叉排序树中插入一个值（非递归算法） \*

\* 4、在二叉排序树中查找一个值（递归算法） \*

\* 5、在二叉排序树中查找一个值（非递归算法） \*

\* 6、在二叉排序树中删除一个值 \*

\* 7、输出二叉树 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

6

删除前的二叉搜索树：

53(1)

17(1)

9(1)

45(1)

23(1)

78(2)

65(1)

87(1)

81(1)

88(1)

请输入你要删除的结点：17

删除后的二叉搜索树：

53(1)

23(1)

9(1)

45(1)

78(2)

65(1)

87(1)

81(1)

88(1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、建立一棵二叉排序树 \*

\* 2、在二叉排序树中插入一个值（递归算法） \*

\* 3、在二叉排序树中插入一个值（非递归算法） \*

\* 4、在二叉排序树中查找一个值（递归算法） \*

\* 5、在二叉排序树中查找一个值（非递归算法） \*

\* 6、在二叉排序树中删除一个值 \*

\* 7、输出二叉树 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

7

53(1)

23(1)

9(1)

45(1)

78(2)

65(1)

87(1)

81(1)

88(1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、建立一棵二叉排序树 \*

\* 2、在二叉排序树中插入一个值（递归算法） \*

\* 3、在二叉排序树中插入一个值（非递归算法） \*

\* 4、在二叉排序树中查找一个值（递归算法） \*

\* 5、在二叉排序树中查找一个值（非递归算法） \*

\* 6、在二叉排序树中删除一个值 \*

\* 7、输出二叉树 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

8

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、新建学生通讯录 \*

\* 2、向学生通讯录插入学生信息 \*

\* 3、在通讯录删除学生信息 \*

\* 4、从文件中读取通讯录信息 \*

\* 5、向文件写入学生通讯录信息 \*

\* 6、在通讯录中查询学生信息 \*

\* 7、在屏幕中输出全部学生信息 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

1

请输入添加学生的个数：8

请输入学号：53

请输入姓名：寮犱笁

请输入性别：鐢?

请输入电话：11111111111

请输入地址：鍖椾含涓滃煄鍖?

请输入出生日期：(格式yy mm dd)2000-1-1

信息录入完毕！

请输入学号：78

请输入姓名：鏉庡洓

请输入性别：鐢?

请输入电话：22222222222

请输入地址：涓滆帪涓€鍖?

请输入出生日期：(格式yy mm dd)2000-2-14

信息录入完毕！

请输入学号：65

请输入姓名：鐜嬩簲

请输入性别：鐢?

请输入电话：33333333333

请输入地址：骞垮窞鐝犳捣鍖?

请输入出生日期：(格式yy mm dd)2000-3-23

信息录入完毕！

请输入学号：17

请输入姓名：璋㈠叚

请输入性别：濂?

请输入电话：44444444444

请输入地址：娣卞湷鍗楀北鍖?

请输入出生日期：(格式yy mm dd)2002-4-12

信息录入完毕！

请输入学号：87

请输入姓名：楹讳竷

请输入性别：鐢?

请输入电话：55555555555

请输入地址：鎯犲窞鎯犲煄鍖?

请输入出生日期：(格式yy mm dd)2000-5-6

信息录入完毕！

请输入学号：9

请输入姓名：楗夸簡涔?

请输入性别：鐢?

请输入电话：66666666666

请输入地址：姹曞ご娼槼鍖?

请输入出生日期：(格式yy mm dd)2001-6-12

信息录入完毕！

请输入学号：81

请输入姓名：缇庡洟

请输入性别：濂?

请输入电话：77777777777

请输入地址：姊呭窞鍏村畞甯?

请输入出生日期：(格式yy mm dd)2001-7-23

信息录入完毕！

请输入学号：45

请输入姓名：鑰虫満

请输入性别：鐢?

请输入电话：88888888888

请输入地址：浣涘北楂樻槑鍖?

请输入出生日期：(格式yy mm dd)2001-8-24

信息录入完毕！

通讯录创建完毕！

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、新建学生通讯录 \*

\* 2、向学生通讯录插入学生信息 \*

\* 3、在通讯录删除学生信息 \*

\* 4、从文件中读取通讯录信息 \*

\* 5、向文件写入学生通讯录信息 \*

\* 6、在通讯录中查询学生信息 \*

\* 7、在屏幕中输出全部学生信息 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

2

请输入你要插入学生的信息：

请输入学号：23

请输入姓名：绾稿肪

请输入性别：濂?

请输入电话：99999999999

请输入地址：涓婃捣榛勬郸鏂板尯

请输入出生日期：(格式yy mm dd)2000-9-20

信息录入完毕！

插入成功！

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、新建学生通讯录 \*

\* 2、向学生通讯录插入学生信息 \*

\* 3、在通讯录删除学生信息 \*

\* 4、从文件中读取通讯录信息 \*

\* 5、向文件写入学生通讯录信息 \*

\* 6、在通讯录中查询学生信息 \*

\* 7、在屏幕中输出全部学生信息 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

3

请输入你要删除学生的学号：65

删除完毕！

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、新建学生通讯录 \*

\* 2、向学生通讯录插入学生信息 \*

\* 3、在通讯录删除学生信息 \*

\* 4、从文件中读取通讯录信息 \*

\* 5、向文件写入学生通讯录信息 \*

\* 6、在通讯录中查询学生信息 \*

\* 7、在屏幕中输出全部学生信息 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

4

通讯录信息已读取成功！

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、新建学生通讯录 \*

\* 2、向学生通讯录插入学生信息 \*

\* 3、在通讯录删除学生信息 \*

\* 4、从文件中读取通讯录信息 \*

\* 5、向文件写入学生通讯录信息 \*

\* 6、在通讯录中查询学生信息 \*

\* 7、在屏幕中输出全部学生信息 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

5

通讯录信息已写入文件中！

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、新建学生通讯录 \*

\* 2、向学生通讯录插入学生信息 \*

\* 3、在通讯录删除学生信息 \*

\* 4、从文件中读取通讯录信息 \*

\* 5、向文件写入学生通讯录信息 \*

\* 6、在通讯录中查询学生信息 \*

\* 7、在屏幕中输出全部学生信息 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

6

请输入你要查找的学生的学号：23

查找成功！

学号：23 姓名：绾稿肪 性别：濂?

电话：99999999999 地址：涓婃捣榛勬郸鏂板尯 出生日期：2000-9-20

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、新建学生通讯录 \*

\* 2、向学生通讯录插入学生信息 \*

\* 3、在通讯录删除学生信息 \*

\* 4、从文件中读取通讯录信息 \*

\* 5、向文件写入学生通讯录信息 \*

\* 6、在通讯录中查询学生信息 \*

\* 7、在屏幕中输出全部学生信息 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

7

通讯录中全部的学生信息：

学号：53 姓名：寮犱笁 性别：鐢?

电话：11111111111 地址：鍖椾含涓滃煄鍖?出生日期：2000-1-1

学号：17 姓名：璋㈠叚 性别：濂?

电话：44444444444 地址：娣卞湷鍗楀北鍖?出生日期：2002-4-12

学号：9 姓名：楗夸簡涔?性别：鐢?

电话：66666666666 地址：姹曞ご娼槼鍖?出生日期：2001-6-12

学号：45 姓名：鑰虫満 性别：鐢?

电话：88888888888 地址：浣涘北楂樻槑鍖?出生日期：2001-8-24

学号：23 姓名：绾稿肪 性别：濂?

电话：99999999999 地址：涓婃捣榛勬郸鏂板尯 出生日期：2000-9-20

学号：78 姓名：鏉庡洓 性别：鐢?

电话：22222222222 地址：涓滆帪涓€鍖?出生日期：2000-2-14

学号：87 姓名：楹讳竷 性别：鐢?

电话：55555555555 地址：鎯犲窞鎯犲煄鍖?出生日期：2000-5-6

学号：81 姓名：缇庡洟 性别：濂?

电话：77777777777 地址：姊呭窞鍏村畞甯?出生日期：2001-7-23

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 1、新建学生通讯录 \*

\* 2、向学生通讯录插入学生信息 \*

\* 3、在通讯录删除学生信息 \*

\* 4、从文件中读取通讯录信息 \*

\* 5、向文件写入学生通讯录信息 \*

\* 6、在通讯录中查询学生信息 \*

\* 7、在屏幕中输出全部学生信息 \*

\* 8、退出 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

请输入操作指令：

8

1. **实验体会**

这次实验，让我更加理解了二叉排序树的内部结构，对二叉排序树如何操作有了一个更清晰的认知。