**数据结构实验报告—树的基本操作的实现**

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：2020-12-1

1. **实验题目**

编写一个程序实现大学的数据统计。某大学的组织结构如表 1 所示，该数据

存放在文本文件 abc.txt 中。要求采用树的孩子链存储结构存储它，并完成如下

功能（菜单）：

1） 从 abc.txt 文件读数据到Ｒ数组中。

2） 由数组Ｒ创建树ｔ的孩子链存储结构。

3） 采用括号表示输出树ｔ。

4） 求计算机学院的专业数。

5） 求计算机学院的班数。

6） 求电信学院的学生数。

7） 求销毁树。

该大学的组织结构是一种树结构，不是二叉树。采用树的孩子链存储结构ｔ存储，

结点类型如下

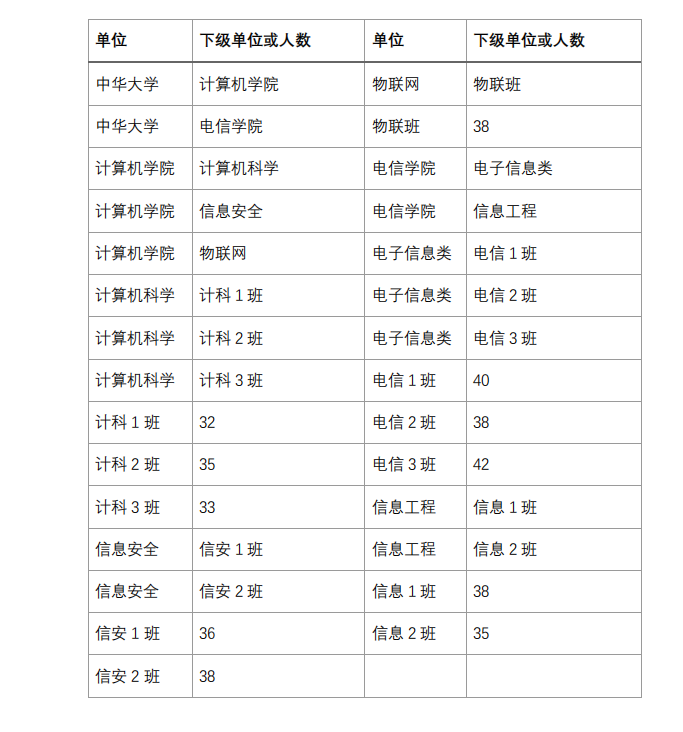
typedef struct node

{

char data[20]; //结点的值：单位名称或人数

struct node \*sons[Maxsons]; //指向孩子结点

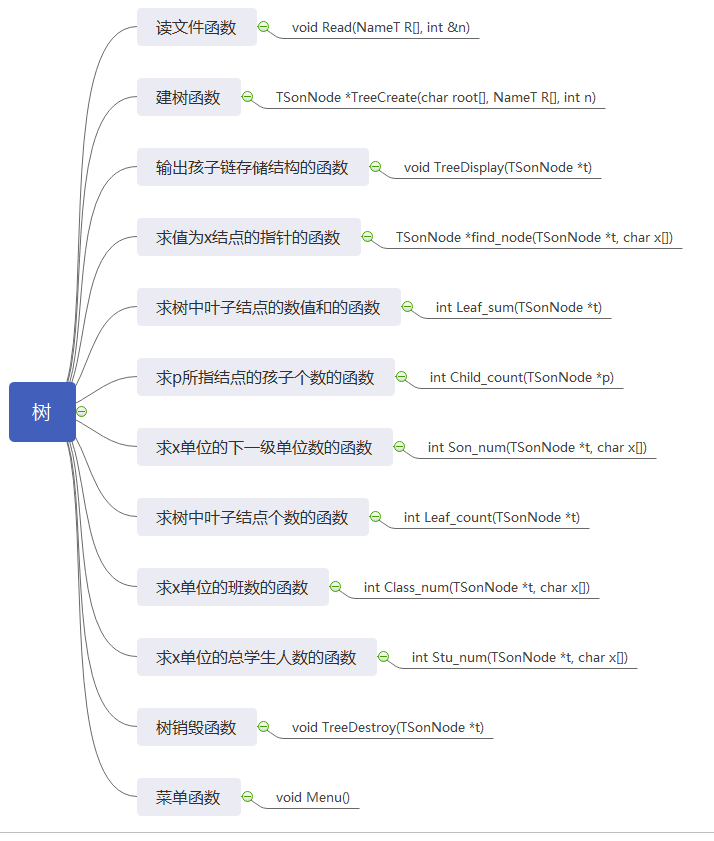
}TSonNode； //声明孩子链存储结构结点类型



1. **实验项目目的**

使学生掌握树的存储结构，熟练使用树的遍历算法进行问题求解

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 <stack>
3. #include <queue>
4. #include <vector>
5. #include <cstdio>
6. #include <cstdlib>
7. #include <cstring>
8. #include <fstream>
9. #include <algorithm>
10. **using** **namespace** std;
11. /\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/
13. **const** **int** Maxsize = 10;
14. **const** **int** Maxsons = 100;
16. **struct** NameT
17. {
18. **char** fname[20];  //单位名称
19. **char** sname[20];  //下级单位名称或人数
20. };
22. **typedef** **struct** node
23. {
24. **char** data[20];  //结点的值：单位名称或人数
25. **struct** node \*sons[Maxsons]; //指向孩子结点
26. }TSonNode;          //声明孩子链存储结构结点类型

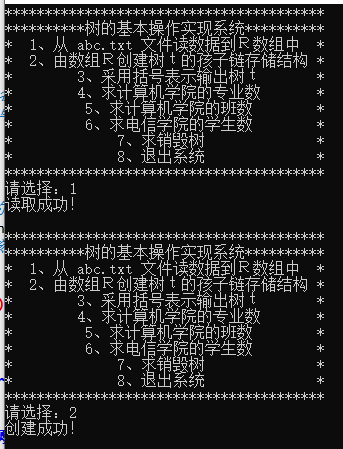
29. **void** Read(NameT R[], **int** &n)  //读abc.txt文件存入R数组中
30. {
31. n = 0;
32. **FILE** \*fp;
33. fp = fopen("abc.txt", "r");
34. **if**(fp == NULL)
35. {
36. cout << "文件打开失败！！！" << endl;
37. **return**;
38. }
39. **while** (!feof(fp))                    //检测流上的文件结束符(文件结束:返回非0值,文件未结束,返回0值)
40. {
41. fscanf(fp, "%s", R[n].fname);    //读fname数据
42. fscanf(fp, "%s", R[n].sname);    //读sname数据
43. n++;
44. }
45. fclose(fp);  //关闭文件
46. }
48. TSonNode \*TreeCreate(**char** root[], NameT R[], **int** n) //创建树
49. {
50. TSonNode \*temp;
51. **int** i = 0, j = 0;
52. temp = (TSonNode \*)malloc(**sizeof**(TSonNode));  //创建根结点
53. strcpy(temp->data, root);
54. **for**(**int** k = 0; k < Maxsize; k++)    //结点所有指针域设置为空
55. {
56. temp->sons[k] = NULL;
57. }
58. **while**(i < n)
59. {
60. **if**(strcmp(R[i].fname, root) == 0)  //找到fname为root的节点
61. {
62. temp->sons[j] = TreeCreate(R[i].sname, R, n); //递归创建子树
63. j++;
64. }
65. i++;
66. }
67. **return** temp;
68. }
70. **void** TreeDisplay(TSonNode \*t) //输出孩子链存储结构
71. {
72. **if**(t != NULL)
73. {
74. printf("%s", t->data);
75. **if**(t->sons[0] != NULL)                //t结点至少有一个孩子
76. {
77. cout << "(";                      //输出一个左括号
78. **for**(**int** i = 0; i < Maxsize; i++)
79. {
80. TreeDisplay(t->sons[i]);
81. **if**(t->sons[i + 1] != NULL)    //如果有下一个孩子
82. cout << ",";              //输出","
83. **else**                          //如果没有下一个孩子
84. **break**;                    //退出循环
85. }
86. cout << ")";                      //输出一个右括号
87. }
88. }
89. }
91. **void** TreeDestroy(TSonNode \*t)  //销毁树t
92. {
93. **if**(t != NULL)
94. {
95. **for**(**int** i = 0; i < Maxsize; i++)
96. {
97. **if**(t->sons[i] != NULL)            //有子树
98. {
99. TreeDestroy(t->sons[i]);
100. }                                //销毁子树
101. **else**
102. {                                //没有子树
103. **break**;
104. }                                //退出循环
105. }
106. free(t);                             //释放根结点
107. }
108. }
110. TSonNode \*find\_node(TSonNode \*t, **char** x[])  //求值为x结点的指针
111. {
112. **int** i;
113. TSonNode \*p;
114. **if**(t == NULL)
115. **return** NULL;
116. **else** **if**(strcmp(t->data, x) == 0)        //找到值为x的结点
117. **return** t;
118. **else**
119. {
120. **for**(i = 0; i < Maxsize; i++)
121. {
122. **if**(t->sons[i] != NULL)
123. {
124. p = find\_node(t->sons[i], x);
125. **if**(p != NULL)
126. **return** p;
127. }
128. }
129. **return** NULL;
130. }
131. }
133. **int** Leaf\_sum(TSonNode \*t)   //求树中叶子结点的数值和
134. {
135. **int** sum = 0;
136. **if**(t == NULL)
137. {
138. **return** 0;
139. }
140. **else**
141. {
142. **if**(t->sons[0] == NULL)                    //t为叶子结点
143. **return** atoi(t->data);                 //字符串形式的数字转为整形
144. **else**                                      //t不为叶子结点
145. {
146. **for**(**int** i = 0; i < Maxsize; i++)
147. {
148. **if**(t->sons[i] != NULL)
149. {
150. sum += Leaf\_sum(t->sons[i]);
151. }
152. **else**
153. {
154. **break**;
155. }
156. }
157. }
158. **return** sum;
159. }
160. }
162. **int** Child\_count(TSonNode \*p)  //求p所指结点的孩子个数
163. {
164. **int** i, sum;
165. TSonNode \*t;
166. **int** num = 0;
167. **for**(i = 0; i < Maxsize; i++)
168. {
169. **if**(p->sons[i] != NULL)
170. {
171. num++;
172. }
173. **else**
174. {
175. **break**;
176. }
177. }
178. **return** num;
179. }
181. **int** Son\_num(TSonNode \*t, **char** x[])  //求x单位的下一级单位数
182. {
183. TSonNode \*p;
184. p = find\_node(t, x);
185. **if**(p == NULL)
186. {
187. **return** 0;
188. }
189. **else**
190. {
191. **return** Child\_count(p);
192. }
193. }
195. **int** Leaf\_count(TSonNode \*t)  //求树中叶子结点个数
196. {
197. **int** num = 0;
198. **if**(t == NULL)
199. {
200. **return** 0;
201. }
202. **else**
203. {
204. **if**(t->sons[0] == NULL)                  //t为叶子结点
205. {
206. num++;
207. }
208. **else**                                    //t不为叶子结点
209. {
210. **for**(**int** i = 0; i < Maxsize; i++)
211. {
212. **if**(t->sons[i] != NULL)
213. {
214. num += Leaf\_count(t->sons[i]);
215. }
216. **else**
217. {
218. **break**;
219. }
220. }
221. }
222. **return** num;
223. }
224. }
226. **int** Class\_num(TSonNode \*t, **char** x[])  //求x单位的班数
227. {
228. TSonNode \*p;
229. p = find\_node(t, x);
230. **if**(p == NULL)
231. {
232. **return** 0;
233. }
234. **else**
235. {
236. **return** Leaf\_count(p);
237. }
238. }
240. **int** Stu\_num(TSonNode \*t, **char** x[])  //求x单位的总学生人数
241. {
242. TSonNode \*p;
243. p = find\_node(t, x);
244. **if**(p == NULL)
245. {
246. **return** 0;
247. }
248. **else**
249. {
250. **return** Leaf\_sum(p);
251. }
252. }

255. **void** Menu()  //菜单函数
256. {
257. **int** n;
258. TSonNode \*t;
259. NameT R[Maxsons];
260. **while**(**true**)
261. {
262. **int** choice;
263. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
264. cout << "\*\*\*\*\*\*\*\*\*\*树的基本操作实现系统\*\*\*\*\*\*\*\*\*\*" << endl;
265. cout << "\*  1、从 abc.txt 文件读数据到Ｒ数组中  \*" << endl;
266. cout << "\*  2、由数组Ｒ创建树ｔ的孩子链存储结构  \*" << endl;
267. cout << "\*        3、采用括号表示输出树ｔ       \*" << endl;
268. cout << "\*        4、求计算机学院的专业数       \*" << endl;
269. cout << "\*         5、求计算机学院的班数        \*" << endl;
270. cout << "\*         6、求电信学院的学生数        \*" << endl;
271. cout << "\*             7、求销毁树               \*" << endl;
272. cout << "\*             8、退出系统              \*" << endl;
273. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
274. cout << "请选择：";
275. cin >> choice;
276. **if** (choice == 1)
277. {
278. Read(R, n);
279. **if** (n == 0)
280. {
281. cout << "树为空！" << endl;
282. }
283. **else** cout << "读取成功！" << endl;
284. }
285. **else** **if** (choice == 2)
286. {
287. t = TreeCreate(R[0].fname, R, n);
288. cout << "创建成功！" << endl;
289. }
290. **else** **if** (choice == 3)
291. {
292. cout << "树的括号输出为：";
293. TreeDisplay(t);
294. cout << endl;
295. }
296. **else** **if** (choice == 4)
297. {
298. cout << "计算机学院专业数为：" << Son\_num(t, "计算机学院") << endl;
299. }
300. **else** **if** (choice == 5)
301. {
302. cout << "计算机学院班数为：" << Class\_num(t, "计算机学院") << endl;
303. }
304. **else** **if** (choice == 6)
305. {
306. cout << "电信学院学生数为：" << Stu\_num(t, "电信学院") << endl;
307. }
308. **else** **if** (choice == 7)
309. {
310. TreeDestroy(t);
311. cout << "销毁成功！" << endl;
312. }
313. **else** **break**;
314. cout << endl;
315. }
316. }
318. **int** main(**int** argc, **char**\*\* argv)
319. {
320. Menu();  //菜单
321. **return** 0;
322. }

**五、算法描述或流程图**

读取文件因为有中文，所以abc.txt文件采用了ANSI编码以便于读入。建树采用了队递归建树的方法。找x单位的总学生人数、找x单位的班级树、找x单位的专业数都是先调用find\_node函数找到这个节点然后再进行操作。

**六、实验数据和实验结果分析**







1. **实验体会**

这次实验，让我更加理解了树与森林的内部的操作，对树与森林有了一个更深入的了解。