# 编译器专题实验报告

#### 目录

##### 实验二:词法部分

###### 实验内容（必做）：

【问题描述】：请根据给定的文法设计并实现词法分析程序，从源程序中识别出单词，记录其单词类别和单词值，输入输出及处理要求如下：

1. 单词类别码 单词的字符/字符串形式(中间仅用一个空格间隔)

2. 单词的类别码请统一按如下形式定义：

词法分析类别码定义如下：

| **单词名称** | **类别码** | **单词名称** | **类别码** | **单词名称** | **类别码** | **单词名称** | **类别码** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Ident** | IDENFR | ! | NOT | \* | MULT | = | ASSIGN |
| **IntConst** | INTCON | && | AND | / | DIV | ; | SEMICN |
| **FormatString** | STRCON | || | OR | % | MOD | , | COMMA |
| main | MAINTK | while | WHILETK | < | LSS | ( | LPARENT |
| const | CONSTTK | getint | GETINTTK | <= | LEQ | ) | RPARENT |
| int | INTTK | printf | PRINTFTK | > | GRE | [ | LBRACK |
| break | BREAKTK | return | RETURNTK | >= | GEQ | ] | RBRACK |
| continue | CONTINUETK | + | PLUS | == | EQL | { | LBRACE |
| if | IFTK | - | MINU | != | NEQ | } | RBRACE |
| else | ELSETK | void | VOIDTK |  |  |  |  |

【输入形式】test.txt中的符合文法要求的测试程序。

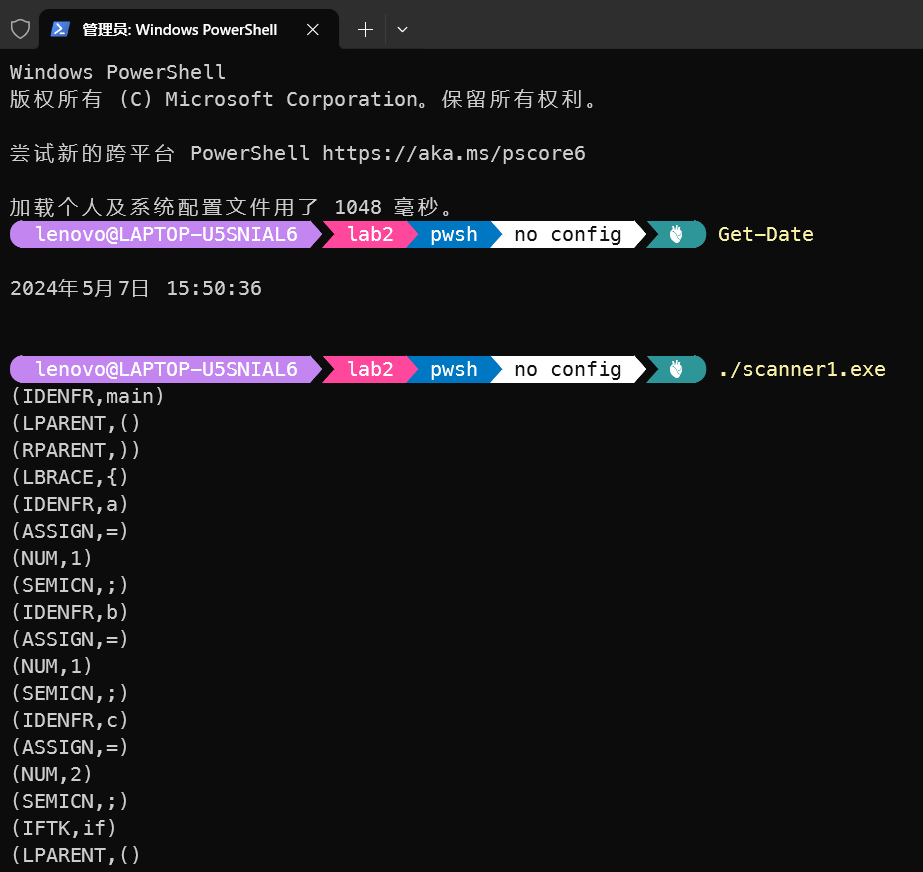
【输出形式】将词法分析的结果输出到命令行界面。

【特别提醒】

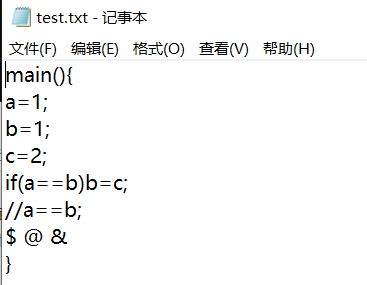
1. 读取的字符串要原样保留着便于输出，特别是数字，这里输出的并不是真正的单词值，其实是读入的字符串，单词值需另行记录。（存储时直接复制字符串来保存）
2. 本次作业只考核对正确程序的处理，但需要为今后可能出现的错误情况预留接口。（错误情况目前考虑了不可识别的符号）
3. 在今后的错误处理作业中，需要输出错误的行号，在词法分析的时候注意记录该信息。（行号用line来记录，设置了一个变量决定输出的时候是否输出行号）
4. 单词的类别和单词值以及其他关注的信息，在词法分析阶段获取后，后续的分析阶段会使用，请注意记录；当前要求的输出只是为了便于评测，完成编译器中无需出现这些信息，请设计为方便打开/关闭这些输出的方案。

###### 实验内容（选做）：

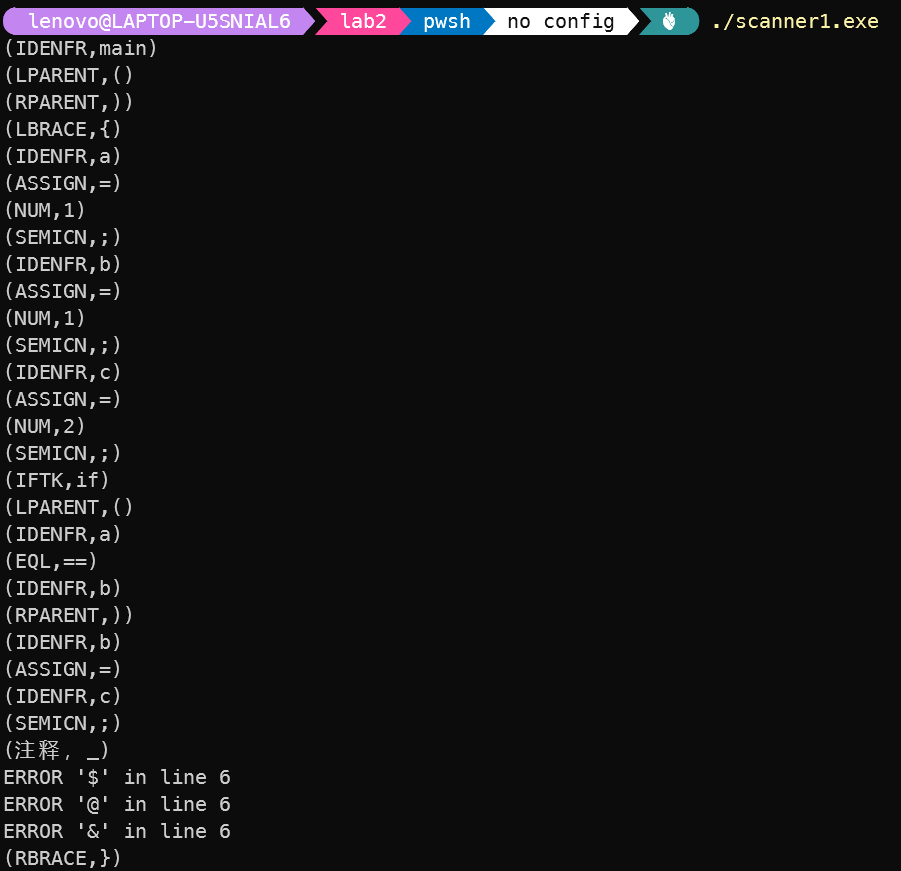
实验结果： （截图。实验结束截图提供实验完成时间，这一点也比较重要，截图中体现提示符，提示符揭示了自己是否独立完成，是合作模式还是独立模式等等。）：



以上是命令行输入，显示了完成实验的时间



以上是test.txt文件的内容，为实验的样例输入。



以上是实验的输出内容。

另外遇到的问题和解决思路（可选）：

解决思路：

1. 为每一类符号设计识别方法和处理方法，在主函数中，不断读入字符，用识别方法进行判断，用处理方法进行处理并输出结果。
2. 在判断’==’(相等符号)和‘=’(输入符)时，采取读到’=’后，再读入一个符号进行判断。
3. 对错误的处理是判断：当前输入的符号不在预设的符号中，则判断为错误并输出行号

遇到的问题：

应该将不同的符号进行更好的分类，比如要将’||’,’&&’等逻辑运算符号和算数运算符放到一类进行处理。

代码很原创（√）：

代码：

1. // 添加头文件
2. #include <stdio.h>
3. #include <stddef.h>
4. #include <ctype.h>
5. #include <string.h>
7. // 变量声明
8. **FILE** \*fp;
9. **char** ch; // 记录当前字符
10. **int** line = 0;
11. **char** category[10];
12. **char** value[10];
14. **const** **char** \*key[13] = {"main", "int", "if", "else", "while", "break", "continue", "float", "void", "const", "printf", "getint", "return"}; // 1.加入其他关键字
15. **const** **char** \*key\_name[13] = {"MAINTK", "INTCON", "IFTK", "ELSETK", "WHILETK", "BREAKTK", "CONTINUETK", "FLOATCON", "VOIDTK", "CONSTTK", "PRINTFTK", "GETINTTK", "RETURNTK"};
16. **const** **char** \*other\_table[11] = {",", ";", "(", ")", "[", "]", "{", "}", "!", "||", "&&"}; // 其他字符
17. **const** **char** \*other\_name[11] = {"COMMA", "SEMICN", "LPARENT", "RPARENT", "LBRACK", "RBRACK", "LBRACE", "RBRACE", "NOT", "OR", "AND"};
18. **const** **char** \*maths\_calcu\_table[4] = {"+", "-", "\*", "/"}; // 3.加入运算符
19. **const** **char** \*maths\_calcu\_name[4] = {"PLUS", "MINU", "MULT", "DIV"};
20. **const** **char** \*relation\_calcu\_table[7] = {"<", "<=", ">", ">=", "=", "!=", "=="}; // 4.加入关系符
21. **const** **char** \*relation\_calcu\_name[7] = {"LSS", "LEQ", "GRE", "GEQ", "ASSIGN", "NEQ", "EQL"};
23. // 函数声明
24. **char** alphaprocess(**char** buffer);
25. **int** search(**char** searchchar[]);
26. **char** numberprocess(**char** buffer);
27. **char** otherprocess(**char** buffer);
28. **char** relation\_calcuProcess(**char** buffer);
29. **char** maths\_calcuProcess(**char** buffer);
30. **char** stringprocess(**char** buffer);
32. **bool** isrelation\_calcu(**char** buffer);
33. **bool** ismaths\_calcu(**char** buffer);
34. **bool** isnumber(**char** buffer);
35. **bool** isalpha(**char** buffer);
36. **bool** isstring(**char** buffer);
37. **bool** isother(**char** buffer);
39. **void** output();
40. **void** ERROR(**char** \*s);
42. // void 改int
43. **int** main()
44. {
45. **if** ((fp = fopen("test.txt", "r")) == NULL)
46. printf("error");
47. **else**
48. {
49. ch = fgetc(fp); // 从指定的文件流中读取一个字符
50. **while** (ch != EOF)
51. {
52. **if** (ch == ' ' || ch == '\n' || ch == '\t')
53. { // 5.忽略空格、制表符(新加)和换行符
54. **if** (ch == '\n')
55. line++;
56. ch = fgetc(fp);
57. }
58. **else** **if** (isstring(ch)) // 处理字符串
59. ch = stringprocess(ch);
60. **else** **if** (isalpha(ch)) // 处理标识符和关键字
61. ch = alphaprocess(ch);
62. **else** **if** (isnumber(ch)) // 处理常数
63. ch = numberprocess(ch);
64. **else** **if** (isother(ch))
65. ch = otherprocess(ch);
66. **else** **if** (isrelation\_calcu(ch)) // 处理关系运算符
67. ch = relation\_calcuProcess(ch);
68. **else** **if** (ismaths\_calcu(ch)) // 处理算术运算符
69. ch = maths\_calcuProcess(ch);
70. **else**
71. {
72. ERROR(&ch);
73. ch = fgetc(fp);
74. }
75. }
76. }
77. fclose(fp); //
78. **return** 0;
79. }
81. **void** ERROR(**char** \*s)
82. {
83. printf("ERROR '%s' in line %d\n", s, line);
84. }
86. **void** output()
87. {
88. printf("(%s,%s)\n", category, value);
89. }
91. **bool** isstring(**char** buffer)
92. {
93. **if** (buffer == 34)
94. **return** **true**;
95. **return** **false**;
96. }
98. **char** stringprocess(**char** buffer)
99. {
100. **int** i = 0;
101. value[i] = buffer;
102. buffer = fgetc(fp);
103. i++;
104. **while** (!isstring(buffer))
105. {
106. value[i] = buffer;
107. buffer = fgetc(fp);
108. i++;
109. }
110. value[i] = buffer;
111. buffer = fgetc(fp);
112. i++;
113. value[i] = '\0';
114. strcpy(category, "STRCON");
115. output();
116. **return** buffer;
117. }
119. **bool** isalpha(**char** buffer)
120. {
121. **if** ((buffer >= 'a' && buffer <= 'z') || (buffer >= 'A' && buffer <= 'Z') || buffer == '\_')
122. **return** **true**;
123. **return** **false**;
124. }
126. // 处理自定义变量和关键字
127. **char** alphaprocess(**char** buffer)
128. {
129. **int** atype; // 保留字数组中的位置
130. **int** i = 0;
131. **while** (isalpha(buffer) || isnumber(buffer))
132. {
133. value[i] = buffer;
134. buffer = fgetc(fp);
135. i++;
136. }
137. value[i] = '\0';
138. atype = search(value);
139. **if** (atype != 0) // 关键字
140. {
141. strcpy(category, key\_name[atype]);
142. }
143. **else** // 变量
144. {
145. strcpy(category, "IDENFR");
146. }
147. output();
148. **return** buffer; // 返回buffer值
149. }
151. **int** search(**char** searchchar[])
152. {
153. **int** i;
154. **int** p;
155. **for** (i = 0; i < 13; i++)
156. {
157. **if** (strcmp(key[i], searchchar) == 0)
158. {
159. p = i;
160. **break**;
161. }
162. **else**
163. p = 0;
164. }
165. **return** p;
166. }
168. **bool** isnumber(**char** buffer)
169. {
170. **if** (buffer >= '0' && buffer <= '9')
171. **return** **true**;
172. **return** **false**;
173. }
175. // 识别正整数小数，实数
176. **char** numberprocess(**char** buffer)
177. {
178. strcpy(category, "NUM");
179. **int** i = 0, flag = 0;
180. **while** (isnumber(buffer) || buffer == '.' || buffer == 'e' || buffer == 'E' || buffer == '+' || buffer == '-')
181. {
182. **if** (buffer == 'e' || buffer == 'E') // 科学计数法
183. {
184. flag = 1;
185. value[i] = buffer;
186. }
187. **else** **if** (buffer == '+' || buffer == '-')
188. {
189. **if** (flag == 1)
190. {
191. value[i] = buffer;
192. }
193. **else**
194. **break**;
195. }
196. **else**
197. {
198. value[i] = buffer;
199. }
200. buffer = fgetc(fp);
201. i++;
202. }
203. value[i] = '\0';
204. output();
205. **return** buffer;
206. }
208. **bool** isother(**char** buffer)
209. {
210. **if** (buffer == '&' || buffer == '|' || buffer == ';' || buffer == ',' || buffer == '(' || buffer == ')' || buffer == '{' || buffer == '}' || buffer == '[' || buffer == ']')
211. **return** **true**;
212. **return** **false**;
213. }
215. // 输出标识符
216. **char** otherprocess(**char** buffer)
217. {
218. **int** i = 0;
219. **while** (isother(buffer))
220. {
221. **if** (buffer == '(' || buffer == ')' || buffer == '{' || buffer == '}' || buffer == '[' || buffer == ']')
222. {
223. value[i] = buffer;
224. buffer = fgetc(fp);
225. i++;
226. **break**;
227. }
228. value[i] = buffer;
229. buffer = fgetc(fp);
230. i++;
231. }
232. value[i] = '\0';
234. **for** (i = 0; i < 11; i++)
235. {
236. **if** (strcmp(value, other\_table[i]) == 0)
237. {
238. strcpy(category, other\_name[i]);
239. output();
240. **return** buffer;
241. }
242. }
243. ERROR(value);
244. **return** buffer;
245. }
247. **bool** isrelation\_calcu(**char** buffer) // 关系运算符
248. {
249. **if** (buffer == '>' || buffer == '<' || buffer == '=' || buffer == '!')
250. **return** **true**;
251. **return** **false**;
252. }
254. **char** relation\_calcuProcess(**char** buffer)
255. {
256. **int** i = 0;
257. **int** flag = 0;
258. **while** ((isrelation\_calcu(buffer)))
259. {
260. // 假如后面都是关系运算符，则一直加入，直到后面不是关系运算符为止
261. value[i] = buffer;
262. i++;
263. buffer = fgetc(fp);
264. }
265. value[i] = '\0';
267. **if** (value[0] == '!' && value[1] == '\0')
268. { // 正确识别“！”与“！=”
269. strcpy(category, "NOT");
270. output();
271. **return** buffer;
272. }
274. // 检查该操作符是否正确
275. **for** (i = 0; i < 7; i++)
276. {
277. **if** (strcmp(value, relation\_calcu\_table[i]) == 0)
278. {
279. strcpy(category, relation\_calcu\_name[i]);
280. output();
281. **return** buffer;
282. }
283. }
284. ERROR(value);
285. **return** buffer;
286. }
288. **bool** ismaths\_calcu(**char** buffer)
289. {
290. **if** (buffer == '+' || buffer == '-' || buffer == '\*' || buffer == '/')
291. **return** **true**;
292. **return** **false**;
293. }
295. **char** maths\_calcuProcess(**char** buffer)
296. {
297. **int** i = 0;
298. **while** ((ismaths\_calcu(buffer)))
299. {
300. // 假如后面都是算术运算符，则一直加入，直到后面不是算术运算符为止
301. value[i] = buffer;
302. i++;
303. buffer = fgetc(fp);
304. }
305. value[i] = '\0';
307. // 检查是否是注释
308. **bool** flag1 = **false**; // 注释“//”
309. **bool** flag2 = **false**; // 注释“/\*\*/”
311. **if** (value[0] == '/' && value[1] == '/')
312. flag1 = **true**;
314. **if** (value[0] == '/' && value[1] == '\*')
315. flag2 = **true**;
317. **if** (flag1)
318. {
319. **while** (buffer != '\n')
320. {
321. buffer = fgetc(fp);
322. }
323. printf("(注释，\_) \n");
324. **return** buffer;
325. }
327. **if** (flag2)
328. {
329. **int** flag3 = 0;
330. **while** (buffer != '/' && flag3 == 0)
331. {
332. **if** (buffer == '\n')
333. line++;
334. **if** (buffer == '\*')
335. flag3 = 1;
336. **else**
337. flag3 = 0;
338. buffer = fgetc(fp);
339. }
340. buffer = fgetc(fp);
341. printf("(注释，\_) \n");
342. **return** buffer;
343. }
345. // 检查该操作符是否在预设的符号之中
346. **for** (i = 0; i < 4; i++)
347. {
348. **if** (strcmp(value, maths\_calcu\_table[i]) == 0)
349. {
350. strcpy(category, maths\_calcu\_name[i]);
351. output();
352. **return** buffer;
353. }
354. }
355. ERROR(value);
356. **return** buffer;
357. }