**暨南大学本科实验报告专用纸**

课程名称 数据挖掘 成绩评定

实验项目名称 简单的类C语言词法分析器 指导教师

实验项目编号 01 实验项目类型 设计性 实验地点 DF116

学生姓名 xxx 学号 2018xxxxxx

学院 信息科学技术学院 系 计算机科学 专业 计算机科学与技术

实验时间 年月 日 午～ 月 日 午 温度 ℃湿度

**一、实验目的**

1. 理解词法分析在编译程序中的作用；加深对有穷自动机模型的理解；掌握词法分析程序的实现方法和技术。

**二、实验内容**

选择部分C语言的语法成分，设计其词法分析程序，要求能够识别关键字、运算符、分界符、标识符、常量（至少是整型常量，可以自己扩充识别其他常量）等，并能处理注释、部分复合运算符（如>=等）。

**三、实验要求**

（1）待分析的简单的语法

关键字：begin if then while do end

运算符和界符：:= + - \* / < <= > >= <> = ; ( ) #

其他单词是标识符id和整型常数num，通过以下正规式定义：

id=l(l|d)\*

num=dd\*

空格、注释：在词法分析中要去掉。

（2）各种单词符号对应的种别编码

图形用户界面, 应用程序, Teams

描述已自动生成

程序实现词法分析，从文件data.txt中读取一段小程序，分解出一个个的单词，其中有关键词，有界符、运算符等等，代码还需实现去掉空格、回车、注释等等情况，最后的输出结果是以单词二元组（单词种别码，单词自身的值）的形式输出。

**四、实验过程**

本人实验采用Golang 1.16进行编写，算法原理是有限状态机。

**五、实验结果**

文本

描述已自动生成

只展示部分结果。

**六、源代码**

main.go:

1. package main
3. import (
4. "bufio"
5. "fmt"
6. "os"
7. "strings"
8. "unicode"
9. )
11. func main() {
12. lexicalAnalysis("data.txt")
13. }
15. var single\_operators string = ";,(){}[]+-\*#"
17. var special\_operators string = "<>=!:"
19. var keywords []string = []string{
20. "begin", "if", "then", "while", "do", "end",
21. "int", "main", "return", "cout",
22. }
24. var token\_type map[string]**int** = map[string]**int**{
25. "!": -1, "#": 0, "begin": 1, "if": 2, "then": 3,
26. "while": 4, "do": 5, "end": 6, "int": 7, "main": 8,
27. "return": 12, "cout": 13, "=": 17, ":=": 18, "==": 21,
28. "+": 22, "-": 23, "\*": 24, "/": 25, "(": 26,
29. ")": 27, "[": 28, "]": 29, "{": 30, "}": 31,
30. ",": 32, ":": 33, ";": 34, ">": 35, "<": 36,
31. ">=": 37, "<=": 38, "!=": 40,
32. }
34. func displayToken(val **int**, token string) {
35. fmt.Printf("(%d, \"%s\")\n", val, token)
36. }
38. func lexicalAnalysis(path string) {
39. file, err := os.Open(path)
40. **if** err != nil {
41. fmt.Printf("[ERROR] File{'%s'} open failed.\n", path)
42. **return**
43. }
45. fmt.Printf("[INFO] Analysising lexical from '%s'.\n", path)
47. in := bufio.NewScanner(file)
48. line := 1
49. **for** in.Scan() {
50. row := in.Text()
51. length := len(row)
53. **for** i := 0; i < length; i++ {
54. var word rune = rune(row[i])
55. **if** unicode.IsSpace(word) {
56. // Skip whitespace
57. **continue**
58. } **else** **if** strings.ContainsRune(single\_operators, word) {
59. // 单字符运算符直接断定
60. displayToken(token\_type[string(word)], string(word))
61. } **else** **if** strings.ContainsRune(special\_operators, word) {
62. // 特殊字符 后面可能跟等号
63. **if** i+1 < length && row[i+1] == '=' {
64. displayToken(token\_type[string(word)+"="], string(word)+"=")
65. i++
66. } **else** {
67. displayToken(token\_type[string(word)], string(word))
68. }
69. } **else** **if** word == '/' {
70. **if** i < length && row[i+1] == '/' {
71. **break** // 跳过注释
72. } **else** {
73. displayToken(token\_type[string(word)], string(word))
74. }
75. } **else** **if** word == '"' {
76. // 字符串
77. s, e := i, i+1
78. **for** ; e < length && row[e] != '"'; e++ {
79. }
80. **if** e == length {
81. panic("Non-terminated string quote!")
82. }
83. fmt.Printf("(T\_STRING, '%s')\n", row[s:e])
84. i = e - 1
85. } **else** **if** unicode.IsLetter(word) || word == '\_' {
86. // 关键字或标识符
87. s, e := i, i+1
88. **for** ; e < length &&
89. (unicode.IsLetter(rune(row[e])) ||
90. unicode.IsDigit(rune(row[e])) ||
91. row[e] == '\_'); e++ {
92. }
93. isIdentifier := **true**
94. **for** \_, keyword := range keywords {
95. **if** row[s:e] == keyword {
96. displayToken(token\_type[keyword], keyword)
97. isIdentifier = **false**
98. }
99. }
100. **if** isIdentifier {
101. displayToken(10, row[s:e])
102. }
103. i = e - 1
104. } **else** **if** unicode.IsDigit(word) {
105. // 数字
106. s, e := i, i+1
107. **for** ; e < length && unicode.IsDigit(rune(row[e])); e++ {
108. }
109. displayToken(20, row[s:e])
110. i = e - 1
111. } **else** {
112. panic(fmt.Sprintf("Unknown word '%c' at line %d.\n", word, line))
113. }
114. }
116. line++
117. }
119. fmt.Println("[SUCCESS] Analysis completed!")
120. file.Close()
121. }

data.txt:

1. // Author: peanutzhen
2. **int** main() {
3. **int** var1 = 6324;
4. **int** var2 = 250;
6. **int** array[] = {6, 3, 2, 4, 6324};
7. // This is a comment.
9. **do**
10. {
11. begin i := 0;
12. **if** (var1 > i) then
13. argv1 = argv1 - argv2;
14. end
15. } **while** (argv1 < 0);
16. cout < (var1\*var2 + var2/var1);
18. **return** 0;
19. }