词法分析器实验报告

-----2020200982 闫世杰

sysy 语言集

关键字(K): 直接精确匹配

基本类型: const int void

if 语句: if else

while 语句: while break continue

函数: main return

标识符(I): 字母/下划线开头+字母/下划线/数字

(由于没有字符串以及 char 类型变量,因此无需区分标识符与字符串,极大简化了难度

正则式: [A-Za-z_]+[A-Za-z0-9_]*

常量(C): 仅含数字类型,仅考虑正数,正负性留在语法分析

十进制: [1-9][0-9]*

十六进制: 0[x|X][1-9a-fA-F][0-9a-fA-F]*

八进制: 0[1-7][0-7]* (以 0 开头数字统一匹配成八进制,因此 010 算作是合法八进制

科学计数法: [1-9](.[1-9][0-9]*)[e|E][+-]*[0-9]* (小数部分可有可无

特殊判断: 0 (数字位数>1 时,首位不能为 0,正则匹配时需要以[1-9]+开头,这样就漏去了数字 0,但是也没有想到较好的方法统一处理,所以采取特殊匹配

运算符(O): 直接精确匹配

数值计算: +-*/%

逻辑运算: && || !

关系运算: == != >= <= > <

赋值运算: =

界符(D): 直接精确匹配

括号符: {} [] ()

边界符: , ;

其他(T): 不符合匹配

不在符号表的字符串: [@#$:”?\`^.]+[@#$^”?\`.a-zA-Z_0-9]*

数字开头的非标识符: [0-9]+{all}+

(all 指所有的字符, 为保证数字匹配,将此匹配法则放在常量匹配之后

注释:

//: //后的整行都是注释, 直接匹配 //.*\n 即可,较为简单

/**/: 匹配/*后进入注释状态,取消一切匹配操作,只进行匹配单个字符及\n

匹配*/后退出注释状态,进入常规模式

最初的想法是匹配/*(.*\n)*.**/把所有的注释一次性匹配下来,然后再解析注释来改变行列,但是发现在处理代码紧挨着注释的情况时细节较为繁琐

然后又转换思路,设置全局变量,匹配/*时将设置 flag=1 表示注释,匹配*/更改 flag=0 表示正常匹配,但是后来发现由于 flex 遵循最长匹配原则,/* ... // */ 这种情况下,会将*/匹配为注释,不能正确处理

最后查资料发现,flex 提供状态识别机制,利用该机制,在遇到/*和*/选择更改状态,并且只在正常状态下进行其余匹配即可,由于注释状态下只进行单个字符匹配,在最长匹配机制下,会优先匹配*/退出注释状态,因此可以正常完成状态转换