词法分析器实验报告

运行方式

```
(bash) ~/$ flex mycompiler.l
(bash) ~/$ gcc lex.yy.c -o MyCompiler
(bash) ~/$ ./MyCompiler ./test.sy > ans.out
```

- 使用$flex$对代码进行转化。
- 使用gcc对生成的$lex.yy.c$文件进行编译，
- 程序接受一个命令行参数，指明需要进行词法分析文件所在的路径。

程序内容

1）处理注释

对于注释的处理尽量放到开头以提高优先级，避免注释的内容被其他的规则匹配，导致出现错误。

- 匹配单行注释。

  ```
  "//".*
  ```

  最开头匹配两个斜杠，然后匹配任意多个字符。由于 . 不会匹配回车，所以这样能保证匹配的为单行的注释。

- 匹配多行注释

  ```
  "/*"([^*]|[\r\n]|("*"+([^*/]|[\r\n])))*"*"+"/"
  ```

  先匹配注释的开头，为了避免**多个注释被匹配到一起**的情况，对注释的内容需要进行特殊处理。

  首先是不能有*，如果出现*号，那么后面必须不能是/，防止将多个注释作为同一个注释一起匹配而导致错误。

2）处理算符

```
"="|"+"|"-"|"*"|"/"|"%"|"=="|"!="|"<"|">"|"
<="|">="|"!"|"&&"|"||"|"&"|"|"|"^"
```

使用双引号对符号进行转义，所有的符号之间是或的关系，任意一个都能匹配上。

3）处理界符

```
","|";"|"{"|"}"|"["|"]"|"("|")"
```

使用双引号对符号进行转义，所有的符号之间是或的关系，任意一个都能匹配上。

4）处理关键字

```
"int"|"main"|"return"|"if"|"break"|"continue"|"while"|"const"|"el
se"
```

使用双引号对关键字进行转义，所有的关键字之间是或的关系，任意一个都能匹配上。

5）处理标识符

先进行辅助定义：

```
digit [0-9]
letter [A-Za-z_]
```

然后借助辅助定义对标识符进行匹配：

```
{letter}({letter}|{digit})*
```

最开始一定是一个字母或者下划线，接下来可以没有后续，也可以有一个或多个字母、数字或下划线。

6）处理常量

先进行辅助定义：

```
numberD (-?)([1-9][0-9]+)|([0-9])
numberO (-?)(0[0-7]+)
numberH (-?)("0x"|"0X")([0-9a-fA-F])+
```

然后通过辅助定义进行匹配：

```
{numberD}|{numberO}|{numberH}
```

- *numberD*用于匹配十进制整数，多位数的情况下第一位不能是0，一位数的情况下第一位可以是任何数。
- *numberO*用于匹配八进制整数，一定以0开头，接下来可以是任何数。
- *numberH*用于匹配十六进制整数，一定以“0x”或“0X”开头，接下来可以是任何数。

匹配到常量的时候，需要对该常量的位数进行判定，如果该常量的长度大于10位，需要报*Warning*，但是该常量不会被认为是错误的，仍存将其标识为*C*。

7）处理空格

```
[ \t\r]
```

匹配空格，同时对\r进行处理，防止系统不同带来的差异。

8）处理其他符号

在最后通过 . 对其余无法识别的字符进行匹配。