

Lab1：词法分析器

21307140 李明俊

一. 实验过程简述

本次实验选择使用antlr来完成，主要有下面两个任务：

1. **Task1**: 获取位置信息，包括: 特定标志符 (leadingSpace, startOfLine)，文件路径，行号，列号
2. **Task2**: 完善token类别

Task1

在第一个任务中，由于标识符，文件路径，行号都是与上下文有关的，在一个单独的token中无法获得，因此需要定义一系列的静态变量，来保证该函数在多次调用时这些静态变量可以保持不变，从而获得上下文信息。

```
static bool leadingSpace = 0;    //判断是否在空格后
static bool startOfLine = 0;    //判断是否是新的一行
static std::string location;    //地址
static int line;                //行号
```

标识符通过处理空格和换行token即可，当出现这种token时用对应静态变量记录一下

```
// 如果是whitespace，记录并跳过
if(tokenTypeName=="whitespace"){
    leadingSpace = 1;
    return;
}
// 如果是newline，记录，更新行号，并跳过
if(tokenTypeName=="Newline"){
    startOfLine = 1;
    line++;
    return;
}
//...
//...
//输出空格和换行信息
int cnt = 0;    //记录标志的个数，便于协调空格
if (startOfLine) {
    outFile << "\t [StartOfLine]";
    startOfLine = 0; //清零，很重要
    cnt++;
}
if (leadingSpace){
    if(cnt==0)
        outFile << "\t [LeadingSpace]";
    else outFile << " [LeadingSpace]";
    leadingSpace = 0; //清零，很重要
```

```
    cnt++;  
}
```

文件路径以及行号是需要通过预处理行获取的。一开始自己写的代码是从第一行中获取文件路径，然后行号就是通过getline的API减去预处理行的数量获得，但是这种做法是错误的，在35号样例中会出错，原因是一个代码中可能有多个文件路径（例如include文件）和不同的对应行号。因此就需要在每一个预处理行都要获取一次路径和行号信息

```
if (tokenTypeName == "LineAfterPreprocessing") {  
    //提取地址  
    std::string s = token->getText();  
    size_t first_quote = s.find("\"");  
    size_t second_quote = s.find("\"", first_quote + 1);  
    location = s.substr(first_quote + 1, second_quote - first_quote - 1);  
    //提取行号  
    line = 0;  
    int ptr = 2;  
    while(s[ptr]==' ') ptr++;  
    while(s[ptr]!=' '){  
        line = line * 10 + (s[ptr]-'0');  
        ptr++;  
    }  
    line = line - 1; //去掉注释行的换行符  
    return;  
}
```

列号可以通过API直接获得，最后将路径，行号，列号信息组合在一起即可

```
std::string locInfo =  
    "Loc=<" + location + ":" +  
    std::to_string(line) + ":" +  
    std::to_string(token->getCharPositionInLine() + 1) + ">";
```

Task2

该任务主要是在 `SYSULexer.g4` 文件中实现的，在该文件中通过正则表达式表示出各种形式的token。

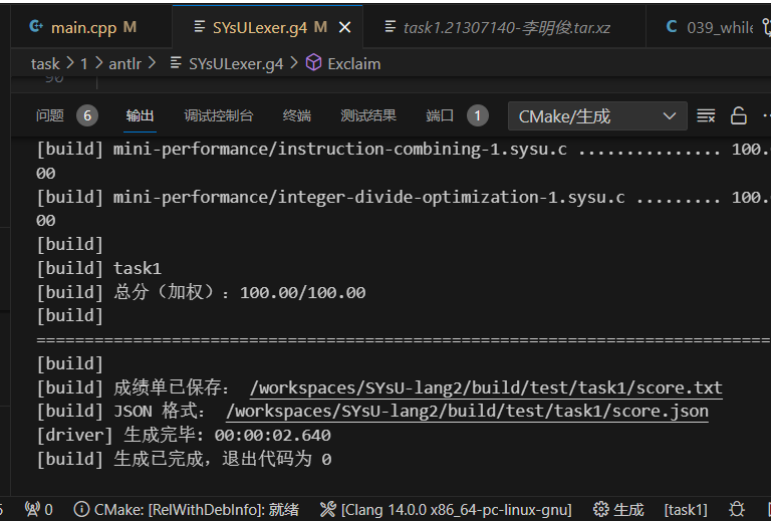
本次实现中添加的token类型如下：

```
"const", "minus", "star", "slash", "percent", "greater", "if", "while",  
"else", "equalequal", "void", "pipepipe", "ampamp", "less", "break",  
"continue", "lessequal", "greaterequal", "exclaimequal", "exclaim",
```

具体内容可见提交代码，这里不再赘述。

二. 实验运行结果

实验代码在本地和评测机中均可以获得满分成绩



```
task > 1 > antlr > SYSULexer.g4 > Exclaim
[build] mini-performance/instruction-combining-1.sysu.c ..... 100.00
[build] mini-performance/integer-divide-optimization-1.sysu.c ..... 100.00
[build] task1
[build] 总分（加权）：100.00/100.00
[build]
[build] 成绩单已保存： /workspaces/SYSU-lang2/build/test/task1/score.txt
[build] JSON 格式： /workspaces/SYSU-lang2/build/test/task1/score.json
[driver] 生成完毕：00:00:02.640
[build] 生成已完成，退出代码为 0
```

yanhuojunjun

提交时间：2024-03-23 23:59:59

提交频率限制：每 15 分钟 1 次

编译原理实验一

提交要求



提交项目里打包好的压缩包，点击提交即可。

须知

- 0. 评测机负责人为郑腾扬，联系方式：zhengty26@mail2.sysu.edu.cn 或QQ群内@我
- 1. 本次不要求提交到排行榜，同学们可以自行决定是否要提交到排行榜
- 2. 有能力本地测评的同学请在本地测试后再提交到评测机，评测机的资源请留给有需要的同学
- 3. 我们将会在每次实验结束后进行代码审查，一经审定为抄袭者，本次实验分数无效

提交记录

+ 上传提交

提交 ID	提交时间	得分	操作
216	2024-03-23 11:52:08	7400 / 7400	 

三. 感想与建议

通过本次实验对词法分析器通过正则表达式获取token的过程有了大致的了解，但是由于我们在这个实验中只是完成了正则表达式部分的编码，剩下的部分是由现成的工具完成的，可能会导致对词法分析底层的过程还是不太清晰，比如可以让学生书写部分链接antlr的相关代码或者相关的cmake代码来解决这一问题？