编译原理实验报告

task1——词法分析器（antlr）

实验评分截图

```
[build] mini-performance/instruction-combining.sysu.c ................. 100.00/100.00
[build] mini-performance/integer-divide-optimization.sysu.c ........... 100.00/100.00
[build] mini-performance/mm.sysu.c ..................................... 100.00/100.00
[build]
[build] task1
[build] 总分（加权）：100.00/100.00
[build] ================================================================================
[build]
[build] 成绩单已保存： /YatCC/build/test/task1/score.txt
[build] JSON 格式： /YatCC/build/test/task1/score.json
[driver] Build completed: 00:00:06.038
[build] Build finished with exit code 0
```

实验过程

一、文法设计（SYsULexer.g4）

```
lexer grammar SYsULexer;

// 关键字
Int : 'int';
Return : 'return';
If : 'if';
Else : 'else';
// ...其他关键字...

// 运算符
Plus : '+';
EqualEqual : '==';
// ...其他运算符...

// 标识符和常量
Identifier : [a-zA-Z_][a-zA-Z0-9_]*;
Constant : [0-9]+;

// 预处理指令
LineAfterPreprocessing : '#' ~[\r\n]* -> skip;

// 空白字符
Whitespace : [ \t]+ -> skip;
Newline : ('\r' '\n'? | '\n') -> skip;
```

二、核心实现（lexer.cpp）

```
void print_token(const antlr4::Token* token,
                 const antlr4::CommonTokenStream& tokens,
                 std::ofstream& outFile,
                 antlr4::Lexer& lexer,
                 SourcePosition& pos,
                 int lastLine) {
    auto& vocabulary = lexer.getVocabulary();
    std::string tokenTypeName = (token->getType() == antlr4::Token::EOF) ?
                                "EOF" :
                                std::string(vocabulary.getSymbolicName(token-
>getType()));

    if (tokenTypeName.empty()) tokenTypeName = "<UNKNOWN>";
    if (tokenTypeMapping.find(tokenTypeName) != tokenTypeMapping.end()) {
        tokenTypeName = tokenTypeMapping[tokenTypeName];
    }

    // 计算实际行号（考虑行号偏移）
    pos.line = token->getLine() - pos.lineOffset - 1;
    pos.column = token->getCharPositionInLine();

    bool startOfLine = (pos.line != lastLine);

    bool leadingSpace = false;
    if (pos.column > 0) {
        // 获取token前的文本
        size_t tokenStart = token->getStartIndex();
        if (tokenStart > 0) {
            char prevChar = lexer.getInputStream()-
>getText(antlr4::misc::Interval(tokenStart-1, tokenStart-1))[0];
            leadingSpace = (prevChar == ' ' || prevChar == '\t');
        }
    }

    if (token->getText() != "<EOF>") {
        outFile << tokenTypeName << " '" << token->getText() << "'";
    } else {
        outFile << tokenTypeName << " ''";
    }

    if (startOfLine) outFile << "\t [StartOfLine]";
    if (leadingSpace) outFile << "\t [LeadingSpace]";

    // 使用原始文件路径而不是预处理后的路径
    std::string outputPath = pos.filename;
    size_t build_pos = outputPath.find("/build/");
    if (build_pos != std::string::npos) {
        size_t test_pos = outputPath.find("/test/", build_pos);
        if (test_pos != std::string::npos) {
            outputPath = outputPath.substr(test_pos + 1);
        }
    }
```

```
    outFile << "\tLoc=<" << outputPath << ":" << pos.line << ":" << pos.column +
1 << ">" << std::endl;
}
```

三、问题解决

1. 预处理行号处理

   在测试文件里，#开头的预处理行影响行号计数，且文件位置不同于源文件所在位置，而是井号行里的位置/因此写了个代码对行号偏移进行计算，忽略＃行，并提取＃行的位置作为输出

```
void handleLineDirective(const antlr4::Token* token, const std::string&
directive, SourcePosition& pos) {
    size_t lineNumEnd = directive.find('"');
    if (lineNumEnd == std::string::npos) return;

    // 提取行号
    size_t lineNumStart = directive.find_first_not_of(" \t", 1); // 跳过#和空
格
    if (lineNumStart == std::string::npos || lineNumStart >= lineNumEnd)
return;

    int newLineNum = std::stoi(directive.substr(lineNumStart, lineNumEnd -
lineNumStart));

    // 提取文件名
    size_t firstQuote = directive.find('"', lineNumEnd);
    size_t lastQuote = directive.rfind('"');
    if (firstQuote == std::string::npos || lastQuote == std::string::npos ||
firstQuote >= lastQuote)
        return;

    std::string newFileName = directive.substr(firstQuote + 1, lastQuote -
firstQuote - 1);

    // 更新位置信息
    if (newFileName != "<built-in>" && newFileName != "<command line>") {
        pos.filename = newFileName;
    }

    // 关键修改：根据预处理指令中的行号重置行号偏移
    pos.lineOffset = token->getLine() - newLineNum;
}
```

2. leadingspace和startofline判断

   在我修改代码过程中，发现会出现leadingspace的判断失误，因此修改了其判断逻辑：

   - 只有当 token 前面确实有空格或制表符时才标记为 [LeadingSpace]
   - 通过检查 token 前一个字符来确定是否有前导空格

   修改代码如上。

实验心得

这些实验感觉都特别难，AI并不好用，常常不知道自己要做什么，要修改哪些文件。且在调试方面也比较麻烦，在不熟悉系统的前期常常需要一个一个进行比对查看错误，花了很长时间。在对于一些难点也是花了很长时间去突破，比如预处理指令处理：通过lineOffset动态调整行号、位置信息精确到列：需考虑制表符和空格等。不过通过本次很难的实验，给我留下了很深的印象，我系统掌握了现代词法分析器开发的全流程，特别是理解了工业级编译器前端的位置信息处理机制，为后续语法分析实验打下了坚实基础。