编译原理实验报告

task2——语法分析器（bison）

实验评分截图

```
[build] mini-performance/integer-divide-optimization.sysu.c ........... 100.00/100.00
[build] mini-performance/mm.sysu.c ..................................... 100.00/100.00
[build]
[build] task2
[build] 总分（加权）：100.00/100.00
[build] ================================================================================
[build] 成绩单已保存： /YatCC/build/test/task2/score.txt
[build] JSON 格式： /YatCC/build/test/task2/score.json
[driver] Build completed: 00:00:15.863
[build] Build finished with exit code 0
```

实验过程

一、词法分析（lex.cpp）

参考 task1 的标准输出 build/test/task1///answer.txt 补充lex.cpp文件的kTokenId，保证可能出现的 token 类型，均在kTokenId中被定义。补充如下：

```
static const std::unordered_map<std::string, int> kTokenId = {
    { "identifier", IDENTIFIER },
    { "numeric_constant", CONSTANT },
    { "hexadecimal_constant", CONSTANT },
    { "int", INT },
    { "void", VOID },
    { "return", RETURN },
    { "const", CONST },
    { "if", IF },
    { "else", ELSE },
    { "while", WHILE },
    { "do", DO },
    { "break", BREAK },
    { "continue", CONTINUE },
    { "l_paren", '(' },
    { "r_paren", ')' },
    { "l_brace", '{' },
    { "r_brace", '}' },
    { "semi", ';' },
    { "equal", '=' },
    { "l_square", '[' },
    { "r_square", ']' },
    { "comma", ',' },
    { "minus", '-' },
    { "plus", '+' },
    { "star", '*' },
    { "slash", '/' },
```

```
    { "percent", '%' },
    { "less", '<' },
    { "greater", '>' },
    { "exclaim", '!' },
    { "eof", YYEOF },
    { "equalequal", EQ_EQ },     // ==
    { "exclaimequal", NE },      // !=
    { "lessequal", LE },         // <=
    { "greaterequal", GE },      // >=
    { "ampamp", AND_AND },       // &&
    { "pipepipe", OR_OR },       // ||
};
```

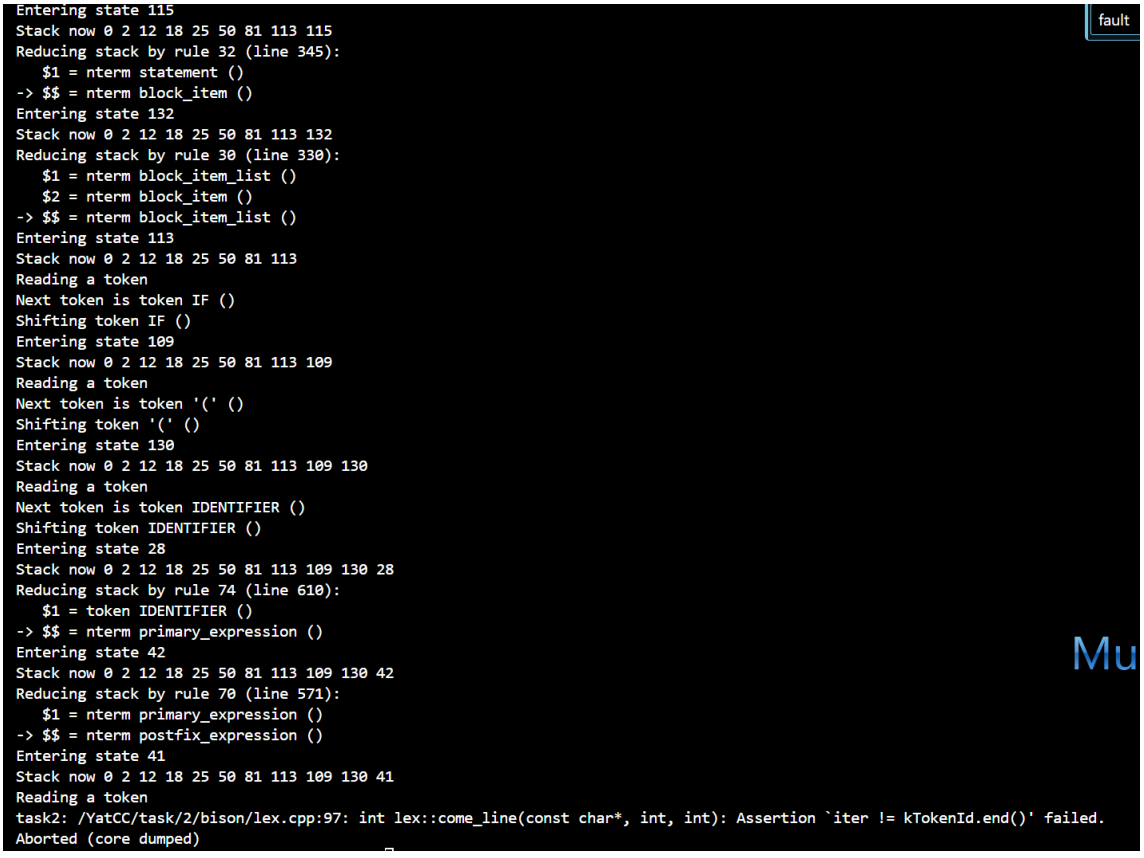
二、语法分析

类型检查和 ASG 生成 JSON 文件这两部分代码已经实现。阅读common/asg.hpp以及文法参考，了解不同类型 ASG 节点（结构体）的成员以及含义，并在par.y中补充缺少的语义规则。因代码太长这里就不多加展示。

三、问题解决

1. debug

   首先我选择了一个与实验一不同的方法进行代码编写（因为听说简单），所以我对bison和flex还不了解。即使看了教程也不知道哪个文件该写些什么。且发现实验输出若有错误会直接显示输出文件损坏，无法精确查看是哪里出了问题。于是只能参考教程中的yydebug发现问题。的确会简单很多



实验心得

实验太难了！！！！！

不过通过实现语法分析器，我深刻理解了如何将词法分析后的Token序列转换为有意义的语法结构，并构建AST。这让我对编译器的前端处理流程有了更清晰的认识。在实验中常常会遇到小bug比如：在解析复杂表达式（如+-!!!a）时，运算符的优先级和结合性需要仔细处理；还有边界情况的处理：空语句（仅含;）和短路求值（如&&、||）是容易忽略的细节。实验中，我意识到编译器必须严格处理这些特殊情况，否则会导致语义错误。

本次实验让我掌握了bison语法分析的核心技术，也让我体会到编译器设计的复杂性与严谨性。