COOL 语言词法分析器开发报告

Compiler Principle Assignment

姓名: 在此处填写你的姓名
学号: 在此处填写你的学号
班级: 在此处填写你的班级

2025 年 11 月 14 日

摘要

(自己根据情况改改) 本文档详细记录了 COOL (Classroom Object-Oriented Language) 词法分析器的设计与实现过程。报告首先深入阐述 Flex 词法分析器的工作原理，包括有限状态自动机理论、模式匹配机制和状态转换过程；然后详细说明词法规则的实现方法，涵盖关键字、标识符、字符串、注释及错误处理；最后通过完整的测试验证，包括集成测试，展示词法分析器与编译器其他组件的协作以及最终程序的正确运行。

1 评分细则与报告要求

本报告的最终得分将基于以下几个方面。请确保您的报告和代码完整覆盖了所有评分项。

特别说明：

- **Flex 原理阐述**（20 分）和**集成测试**（15 分）是本次作业的重点，请务必认真完成
- 代码查重将使用专业查重工具检测，查重率达到 100% 及以上将导致整个报告得 0 分
- 鼓励独立思考和原创性工作，可以参考资料但必须用自己的话表述
- 教师主观评分（10 分）将根据报告整体质量、理解深度、创新性等综合评定

2 项目概述与环境

2.1 项目目标

(自己根据情况改改) 本次作业的目标是使用词法分析器生成工具 Flex 为 COOL 语言设计并实现一个完整的词法分析器。该分析器需要能够将 COOL 源代码文本文件转换为一个词法单元（Token）序列，并能正确处理各种边界情况和词法错误。

2.2 开发环境

2.2.1 硬件配置

- **CPU**: 例如：Intel Core i7-10700K @ 3.80GHz
- **内存**: 例如：16GB DDR4
- **硬盘**: 例如：512GB SSD

2.2.2 软件环境

- **操作系统**: 例如：Ubuntu 22.04.3 LTS / Windows 11 Pro 22H2 / macOS Ventura 13.4
- **内核版本**: 运行 uname -r 或 ver 查看
- **Flex 版本**: 运行 flex –version，例如：flex 2.6.4
- **G++ 版本**: 运行 g++ –version，例如：g++ (Ubuntu 11.4.0-1ubuntu1) 11.4.0
- **Make 版本**: 运行 make –version
- **其他工具**: SPIM 版本、编辑器等

表 1: COOL 词法分析器评分标准

类别	评分项说明	分数
理论理解		
Flex 工作原理	详细阐述 Flex 的工作流程、有限状态自动机（DFA/NFA）原理、模式匹配机制、状态转换过程。要求有清晰的图示或流程说明。	20
基础功能实现		
关键字识别	正确识别所有 COOL 关键字，处理大小写不敏感。布尔常量首字母必须小写。	8
标识符与常量	正确区分 TYPE_ID 和 OBJECT_ID，正确解析整数和布尔常量。	8
操作符与注释	完整处理多字符操作符、单行注释、嵌套多行注释。	7
核心难点		
字符串处理	使用状态机正确处理字符串，支持转义字符，处理各种字符串错误。	10
错误处理	检测并报告所有类型的词法错误（未闭合字符串/注释、字符串过长、非法字符等）。	7
测试验证		
集成测试	完整的编译流程展示（词法 → 语法 → 语义 → 代码生成 →SPIM 运行），测试用例设计合理，输出结果完整正确。	15
报告与代码		
报告质量	报告结构清晰，实现说明详细，测试结果完整。	5
代码质量	cool.flex 代码风格良好，逻辑清晰，注释适当。	5
其他		
教师主观评分	整体完成度、创新性、深度理解等综合评价。	10
代码查重	查重率 <50% 得 5 分；60% 得 4 分；70% 得 3 分；80% 得 2 分；90% 得 1 分；100% 及以上整个报告 0 分。	5
	总分	**100**

2.2.3 项目目录结构

```
/usr/class/assignments/PA2/(根据你情况)
|-- cool.flex
|-- test.cl
|-- lexer
|-- cool-lex.cc
|-- Makefile
|-- parser
|-- semant
`-- cgen
```

2.2.4 环境配置过程

3 Flex 词法分析器原理

本节要求详细阐述 Flex 的工作原理和词法分析的理论基础。（**本节占 20 分，是评分重点！**）

3.1 Flex 工作流程

3.2 有限状态自动机（FSA）原理

3.3 模式匹配机制

3.4 状态与状态转换

4 实现细节

本节简要说明词法规则的实现思路。完整代码见附录。

4.1 关键字与标识符

基本要求：

- 关键字（如 class、if、while 等）是大小写不敏感的
- 布尔常量 true 和 false 的首字母必须小写
- TYPE_ID 以大写字母开头，OBJECT_ID 以小写字母开头
- 整数常量为数字序列

4.2 字符串处理

基本要求：

- 字符串以双引号开始和结束，不能跨行
- 支持转义字符：`\n`, `\t`, `\b`, `\f`, `\"`, `\\`
- 最大长度 1024 字符，不能包含空字符
- 需要使用 Flex 状态机（`%x STRING`）处理

4.3 操作符与注释

基本要求：

- 多字符操作符：`<-`、`<=`、`=>`
- 单行注释：`--`到行尾
- 多行注释：`(* *)`，支持嵌套
- 忽略空白字符，换行时更新行号

4.4 错误处理

需要检测的错误：

- 未闭合的字符串、字符串过长、字符串中的空字符
- EOF 在字符串或注释中
- 未匹配的注释结束符 `*)`
- 源代码中的非法字符

5 测试与验证

为了验证词法分析器的正确性，我设计了多个测试用例，并使用了项目提供的测试工具。

5.1 基础功能测试

测试目标：验证关键字、标识符、常量、操作符的正确识别。

测试用例（test_basic.cl）：

```
class Main {
    x : Int <- 42;
    flag : Bool <- true;
    main() : Int { x };
};
```

Listing 1: 基础功能测试

测试命令：

```
$ ./lexer test_basic.cl
```

实际输出结果：

```
#name "test_basic.cl"
#1 CLASS
#1 TYPEID Main
#1 '{'
#2 OBJECTID x
#2 ':'
#2 TYPEID Int
#2 ASSIGN
#2 INT_CONST 42
#2 ';'
#3 OBJECTID flag
#3 ':'
#3 TYPEID Bool
#3 ASSIGN
#3 BOOL_CONST true
#3 ';'
#4 OBJECTID main
...
```

5.2 字符串与注释测试

测试目标：验证字符串转义字符、嵌套注释、各种错误检测。

测试用例（test_string.cl）：

```
(* 测试注释 (* 嵌套注释 *) *)
class Test {
    str1 : String <- "Hello\nWorld";  -- 转义字符
    str2 : String <- "Quote\"Test\"";
};
```

Listing 2: 字符串测试

测试命令与输出：

```
$ ./lexer test_string.cl
```

```
#name "test_string.cl"
#2 CLASS
#2 TYPEID Test
#2 '{'
#3 OBJECTID str1
#3 ':'
#3 TYPEID String
#3 ASSIGN
#3 STR_CONST "Hello\nWorld"
...
```

5.3 错误处理测试

测试目标：验证各种错误情况的检测和报告。

测试 1：未闭合字符串

测试代码：`class Main { str : String <- "unclosed`
输出：`#1 ERROR "Unterminated string constant"`

测试 2：未匹配的注释结束符

测试代码：`class Main { x : Int; *) }`
输出：`#1 ERROR "Unmatched *)"`

测试 3：EOF 在注释中

测试代码：`(* comment without closing`
输出：`#1 ERROR "EOF in comment"`

5.4 集成测试

本部分占 15 分，是评分重点！

测试目标：验证词法分析器能与编译器其他阶段（语法分析、语义分析、代码生成）正确协作，最终生成可运行的 MIPS 汇编代码。

测试程序（hello.cl）：

```
class Main inherits IO {
    main() : Object {
        out_string("Hello, COOL!\n")
    };
};
```

Listing 3: 集成测试程序

编译过程：

```
$ ./mycoolc hello.cl
```

运行结果：

```
$ spim hello.s
SPIM Version 8.0 of January 8, 2010
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
Loaded: /usr/class/lib/trap.handler
Hello, COOL!
COOL program successfully executed
```

测试结论：

词法分析器成功识别了所有 Token，编译器顺利完成了语法分析、语义分析和代码生成，生成的 MIPS 汇编代码在 SPIM 模拟器上正确执行，输出了预期结果。这证明词法分析器的实现是正确和完整的。

6 遇到的问题与解决方案

7 总结

通过本次实验，我深入理解了词法分析的理论基础和 Flex 工具的工作原理。从有限状态自动机的理论出发，理解了 Flex 如何将正则表达式转换为高效的 DFA，如何进行模式匹配和状态转换。在实践中，我成功实现了一个功能完整且健壮的 COOL 语言词法分析器，特别是掌握了状态管理机制在处理复杂词法结构时的应用。通过完整的集成测试，我验证了词法分析器能够与编译器其他组件正确协作，最终生成可执行的 MIPS 代码。这次实验让我对编译器前端有了全面而深刻的认识。

A 附录：cool.flex 完整源码

说明：将完成的 `cool.flex` 文件放在同一目录下，重新编译即可自动包含完整代码。

示例结构（3 段式）：定义段 – 规则段 – 用户代码段