# **一**编译原理实验

ECNU 后端及编译器具体实现 - 第六小组

# **②实现语言**

JAVA

# **②**已完成

- ② 词法分析 + 分词符号表
- 溪 LL(1) 语法分析 + 恐慌模式错误恢复
- 羚 四元式中间代码 + 符号表生成
- 📤 语义分析解释器

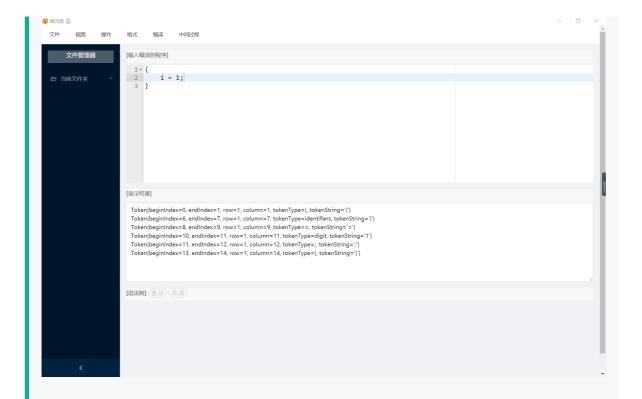
# 2 效果图

a) 基本的输入、输出界面;

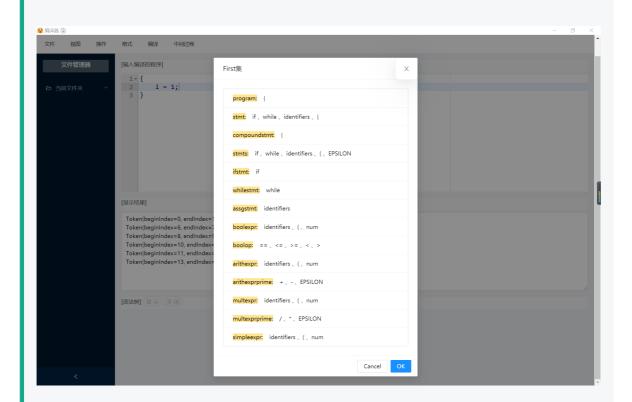
1. 输入

```
| i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i = 1 ; | i =
```

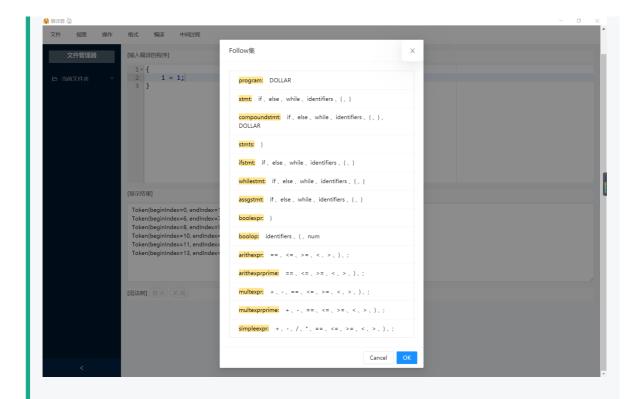
2. 词法分析结果



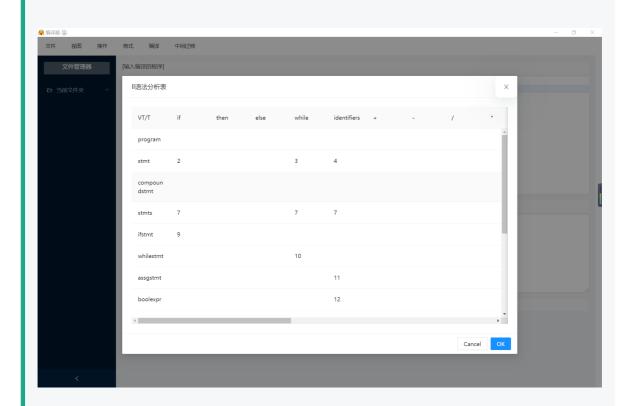
### 3. LL(1) FIRST 集



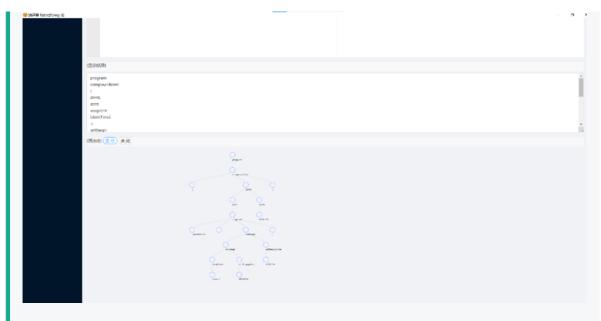
### 4. LL(1)Follow集

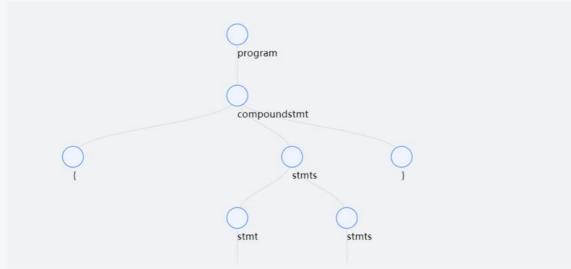


### 5. LL (1)解析表



### 6. LL(1) 生成的语法树

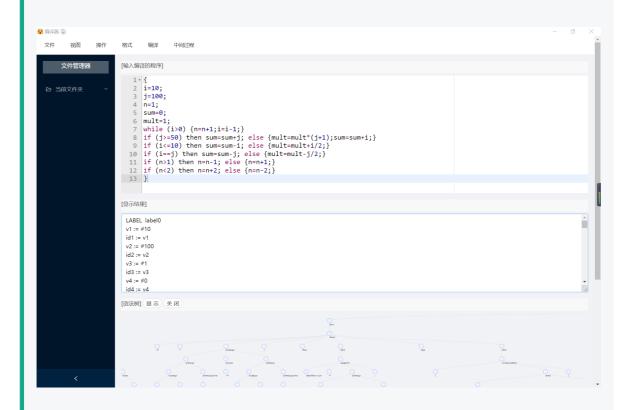




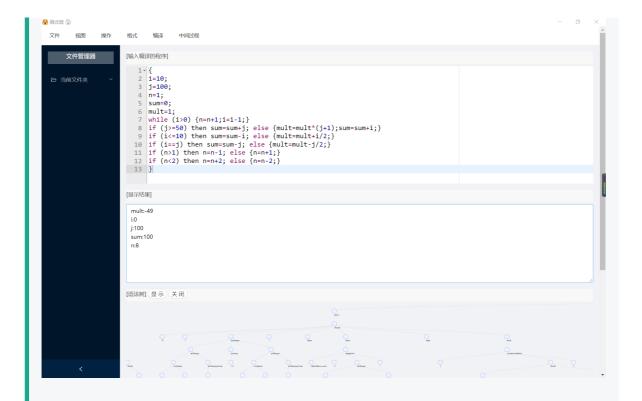
## 7. 错误提示 (错误高亮 和 错误日志显示)

```
🥯 编译器 hzr/czh/wyj 🗟
  文件
           视图
                    操作
                                    编译
                                                中间过程
                             「輸入编译的程序」
                                 2 i = * 10;
3 j = / 100;
   🗅 当前文件夹
                                 4 n=1;
                                 5 sum=0;
                                 6 mult=1;
                                 7 while (i>0) {n=n+1;i=i-1;}
                                while (190) {n=n+1;1=1-1;}
8 if (j>=50) then sum=sum+j; else {mult=mult*(j+1);sum=sum+i;}
9 if (i<=10) then sum=sum-i; else {mult=mult+i/2;}
10 if (i arr ==j) then sum=sum-j; else {mult=mult-j/2;}
11 if (n>1) then n=n-1; else {n=n+1;}
12 if (n<2) then n=n+2; else {n=n-2;}</pre>
                              13 }
                             [显示结果]
                              日志格式[行=列=错误信息]
                              行: 10, 列: 7, 此处的字符串"arr" 附近或许存在错误,提示: 出现了额外的标识符,请确认此处是否应该存在该变量
                              行: 3, 列: 5, 此处的字符串"/" 附近或许存在错误,提示: 不合理的操作符,请确认此处是否存在不合理的输入
                               行: 2, 列: 5, 此处的字符串"*" 附近或许存在错误,提示: 不合理的操作符,请确认此处是否存在不合理的输入
                             [语法树] 显示 关闭
```

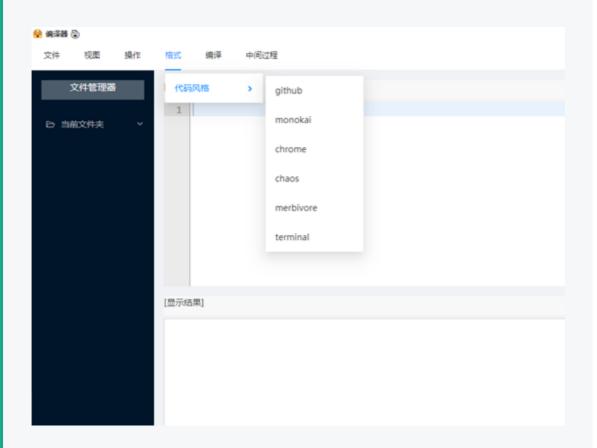
#### 8. 中间代码生成



#### 9. 语义分析解释结果



### 10. 自定义代码风格切换



```
1 a = 7;

2 b = 10;

3 if (a < 10) then a--; else {a = b;}
```

```
1 a = 7;
2 b = 10;
3 if (a < 10) then a--; else {a = b;}
```

```
1 a = 7;
2 b = 10;
3 if (a < 10) then a--; else {a = b;}
```

```
a = 7;
      b = 10;
      if (a < 10) then a--; else {{a = b;}}
b) 界面布局与设计
警 编译器 hzr/czh/wyj ②
文件 视图 操作
              格式
                      中间过程 操作栏
                 编译
               est1.txt 输入编译的First集
               1 · {
2 i=1;
3 }
                        Follow集
                        解析表
              (显示结果)
                                 输出或错误日志
```

# ◎运行说明

- 1. git clone https://github.com/Wind-Gone/Toy-compiler-GUI 克隆本项目的前端库
- 2. yarn 安装相关依赖包
- 3. yarn start 启动 Electron

(语法树) 🗎 🗆

4. git clone https://github.com/Wind-Gone/Toy-compiler 克隆项目后端编译器

语法分析的语法树

5. 启动 Springboot 即可运行

# **Ø**项目目录

```
-src
 |-main
    ├-java
      ∟<sub>com</sub>
          L-example
             L-compiler
                CompilerApplication.java
                                            // Springboot启动入口
                                            // 用于处理前后端请求的Controller层目录
                -controller
                                             // 具体的执行文件Controller.java
                     controller.java
                                             // 实体目录
                -entity
                | |-gui
                                             // 用于与前端交互的特殊类目录
                                             // 与树形组件配套的语法树类
                        GuiNode.java
                        Text.java
                                             // 前端传递的数据的实体类
                | token
                                            // Token相关目录
                       Token.java
                                             // 词法分析解析的Token类
                                             // 词法分析表的主体类
                        TokenTable.java
                        TokenType.java
                                             // Token类型的枚举类
                | |-tree
                                             // 语法树相关目录
                                             // 语法树的实体类
                        SyntaxTree.java
                        TreeNode.java
                                             // 语法树节点的实体类
                  Lwrong
                                             // 错误信息相关目录
                                             // 错误代码和代码指示的信息的枚举类
                         ErrorCode.java
                         WrongMessage.java
                                            // 定义详细错误信息的实体类
                                           // 中间代码生成
                ├intermediateCodeGeneration
                     Dag.java
                     Leaf.java
                      Node.java
                |-lexer
                                              // 词法分析器目录
                      Lexer.java
                                             // 词法分析器(ex1)
                                             // 词法分析器主体,实现了对token判别类型
                      Main.java
                |-|11Parser
                                             // 语法分析目录
                                             // 存储所有语法规则的产生式类
                     Grammer.java
                     LLParser.java
                                             // 语法分析器(ex2)
                     LLUtil.java
                                             // 语法分析器处理first/follow集合等函数的工具类
                      NonTerminalType.java
                                             // 非终结符枚举类
                                             // 语法解析表实体类
                     ParsingTable.java
                     Production.java
                                             // 语法产生式实体类
                                             // 语义分析目录
                -semantic
                                             // 语义分析器(ex4),调用中间代码生成解释器
                     Main.java
                                             // 语义分析其具体执行类
                     SemanticAnalyzer.java
                                             // 语义分析符号表类
                     SymbolTable.java
                                             // 三地址码实体类
                      ThreeCodeTable.java
                                             // 项目所需要使用的工具类目录
                L-utils
                      FileUtils.java
                                             // 文件工具类,负责文件10
    Lresources
          application.properties
  L_test
     ∟java
        L_{-com}
           L_{-example}
               └-compiler
                     CompilerApplicationTests.iava // 常规测试
```

	1.1 .1
TestForGrammer.java	// 语法分析测试
TestForLexer.java	// 词法分析测试
TestForSemantic.java	// 语义分析测试

# 為特別鸣谢

@@caizhenghai

@@Ling-WYJ

@@Wind-Gone