2.1 ToyC 语言的文法

ToyC 语言的文法如下，其中 `CompUnit` 为开始符号：

编译单元	`CompUnit` → `FuncDef`$^+$
语句	`Stmt` → `Block` | “;” | `Expr` “;” | `ID` “=” `Expr` “;”
	| “int” `ID` “=” `Expr` “;”
	| “if” “(” `Expr` “)” `Stmt` (“else” `Stmt`)?
	| “while” “(” `Expr` “)” `Stmt`
	| “break” “;” | “continue” “;” | “return” “;”
语句块	`Block` → “{” `Stmt`* “}”
函数定义	`FuncDef` → (“int” | “void”) `ID` “(” (`Param` (“,” `Param`)*)? “)” `Block`
形参	`Param` → “int” `ID`
表达式	`Expr` → `LOrExpr`
逻辑或表达式	`LOrExpr` → `LAndExpr` | `LOrExpr` “||” `LAndExpr`
逻辑与表达式	`LAndExpr` → `RelExpr` | `LAndExpr` “&&” `RelExpr`
关系表达式	`RelExpr` → `AddExpr`
	| `RelExpr` (“<” | “>” | “<=” | “>=” |
	“==” | “!=”) `AddExpr`
加减表达式	`AddExpr` → `MulExpr`
	| `AddExpr` (“+” | “-”) `MulExpr`
乘除模表达式	`MulExpr` → `UnaryExpr`
	| `MulExpr` (“*” | “/” | “%”) `UnaryExpr`
一元表达式	`UnaryExpr` → `PrimaryExpr`
	| (“+” | “-” | “!”) `UnaryExpr`
基本表达式	`PrimaryExpr` → `ID` | `NUMBER` | “(” `Expr` “)”
	| `ID` “(” (`Expr` (“,” `Expr`)*)? “)”

词法分析

- **目标：**
  - 识别源码中的记号（token），包括标识符、关键字、数字、运算符等

- **关键点：**
  - 完成标识符和整数的正则匹配
  - 忽略空白和注释
  - 完整覆盖 ToyC 定义的运算符和关键字

- **设计：**
  - 构建 DFA
  - 字符匹配，构建token，识别类型
  - 检查非法token，给出错误提示

语法分析（AST输出为例）

- **目标：**
  - 构建抽象语法树（AST），确定源代码结构

- **关键点：**
  - 完整实现文法规则，处理所有产生式
  - 构建解析器（如自顶向下表驱动解析器）
  - 检查结构性错误（括号匹配、非法嵌套）

- **设计：**
  - 定义好 AST 结构
  - 简化和模块化设计每个产生式的解析器
  - 在解析阶段收集函数、参数、变量等定义，构建符号表框架

2.3 ToyC 语言的语义约束

程序结构 程序必须包含一个名为 `main`、参数列表为空、返回类型为 int 的函数作为入口点。

函数 ToyC 中的函数支持传入若干 int 类型的参数，返回值可以为 int 或 void 类型。函数只能声明在全局作用域中且名称唯一，不能声明在函数体内。函数不能作为值来储存、传递或运算。函数调用必须写在被调函数声明之后（支持函数内部调用自身）。返回类型为 int 的函数必须在每一条可能的执行路径上通过 return 语句返回一个 int 类型的值。返回类型为 void 的函数可以不包含 return 语句，如果包含，则不能有返回值。

语句 ToyC 中的语句包括语句块（由大括号包围的语句序列）、空语句（只有分号）、表达式语句（表达式加分号）、变量赋值、变量声明、if-else 条件分支、while 循环、break、continue 和 return，它们的语法和语义与 C 语言一致：没有返回值的函数调用表达式不能作为 if/while 条件或赋值语句的右值；break 和 continue 只能出现在循环中等。

变量声明 ToyC 支持声明 32 位有符号整数（int）类型的局部变量，每个变量声明语句只声明一个变量，且必须带有初始化表达式。变量的使用必须发生在声明之后。变量的作用域规则与 C 语言一致：变量的生命周期从声明点开始，到所在作用域结束时结束；函数形参在函数体内可见；语句块 `Block` 会创建新的作用域；内层作用域会屏蔽外层作用域的同名变量等。

表达式 ToyC 支持逻辑运算、关系运算、算术运算、括号表达式、变量引用、字面值和函数调用等 C 语言中的常见表达式。运算符的优先级、结合性和计算规则等与 C 语言一致：逻辑与、逻辑或运算符遵循“短路计算”规则；非零视为“真”、零视为“假”；除数不能为零等。

语义分析

可实现更多检查

- **目标：**
  - 检查类型和作用域、确定引用和定义是否匹配
- **关键点：**
  - 检查变量是否先声明后使用
  - 检查函数是否先定义后调用
  - 检查类型匹配和返回值类型
- **设计：**
  - 在 AST 遍历过程中建立和管理符号表
  - 检查控制流结构（break、continue、return）是否合法
  - 完成简单类型推导，检查参数和调用参数类型是否匹配

代码生成

- **目标：**
  - 为 AST 节点生成 RISC-V32 汇编代码

- **关键点：**
  - 函数调用栈结构管理
  - 寄存器和堆栈管理
  - 短路计算、条件和循环代码生成

- **设计：**
  - 定义清晰的代码生成接口，例如 emit()
  - 遍历IR成分，转成 RISC-V32代码
  - 完成简单优化（如公共子表达式、常量折叠、寄存器分配），以获取更好性能