编译技术实验二报告

1.语法制导定义

语法制导定义是一种将语言的语法结构和语义处理相结合的方法。在本程序中，通过为每个语法产生式关联语义动作来实现语法制导定义。例如，对于表达式的产生式，语义动作包括生成三地址代码指令，为表达式计算结果分配临时变量等。

对于条件表达式，如 `if (expr) stmt` 这种结构，其语法制导定义为：当解析到 `if` 关键字后，先处理条件表达式 `expr`，生成条件表达式对应的三地址代码，并标记条件为真和为假时的跳转标签；然后处理语句 `stmt`，根据条件表达式的结果决定语句的执行逻辑，生成相应的跳转和标签指令。

对于赋值语句 `id := expr`，语法制导定义为：解析到标识符 `id` 后，匹配赋值符号 `:=`，接着处理表达式 `expr`，为表达式生成三地址代码，最后生成将表达式结果赋值给标识符的三地址代码指令。

2.改写后的产生式集合

2.1 语句（S）

- `S -> ID ASSIGN E SEMICOLON`：表示赋值语句，先识别标识符，然后是赋值符号，接着解析表达式，最后以分号结束。
- `S -> IF C THEN S`：条件语句，`if` 后跟随条件表达式，再是 `then` 关键字，最后是语句。
- `S -> IF C THEN S ELSE S`：带 `else` 分支的条件语句。
- `S -> WHILE C DO S`：循环语句，`while` 后是条件表达式，然后是 `do` 关键字，最后是语句。

2.2 条件表达式（C）

`C -> E COMP_OP E`：条件表达式由两个表达式通过比较运算符连接而成，比较运算符包括 `>`、`<`、`==` 等。

2.3 表达式（E）

- `E -> T`：表达式可以是一个项。
- `E -> E PLUS T`：表达式可以是一个表达式加上一个项。
- `E -> E MINUS T`：表达式可以是一个表达式减去一个项。

2.4 项（T）

- `T -> F`：项可以是一个因子。
- `T -> T MULTIPLY F`：项可以是一个项乘以一个因子。
- `T -> T DIVIDE F`：项可以是一个项除以一个因子。

2.5 因子（F）

- `F -> ID`：因子可以是标识符。
- `F -> INT8`：因子可以是八进制整数。
- `F -> INT10`：因子可以是十进制整数。
- `F -> INT16`：因子可以是十六进制整数。

- `F -> LPAREN E RPAREN`：因子可以是括号括起来的表达式。

3. 递归子程序的算法

3.1 语句处理函数（S）

首先检查当前词法单元 `currentToken`：

- 如果是 `TOK_ID`，表示是赋值语句。记录标识符，匹配赋值符号 `TOK_ASSIGN`，调用表达式处理函数 `E()` 获取表达式结果，再匹配分号 `TOK_SEMICOLON`，最后生成将表达式结果赋值给标识符的三地址代码指令。
- 如果是 `TOK_IF`，表示是条件语句。前进到下一个词法单元，调用条件表达式处理函数 `C()` 获取条件表达式的代码和跳转标签，匹配 `TOK_THEN`，再调用 `S()` 处理 `then` 后的语句。如果当前词法单元是 `TOK_ELSE`，则前进并再次调用 `S()` 处理 `else` 后的语句，根据条件表达式的跳转标签生成相应的标签和跳转指令。
- 如果是 `TOK_WHILE`，表示是循环语句。生成循环开始的标签，调用 `C()` 获取条件表达式的代码和跳转标签，匹配 `TOK_DO`，调用 `S()` 处理循环体语句，再生成回到循环开始的跳转指令和循环结束的标签。
- 其他情况，抛出语法错误。

3.2 条件表达式处理函数（C）

- 先调用表达式处理函数 `E()` 获取第一个表达式的代码和结果变量。
- 检查当前词法单元是否为比较运算符（`TOK_GREATER`、`TOK_LESS`、`TOK_EQUAL` 等），如果是，记录运算符并前进到下一个词法单元；否则抛出语法错误。
- 再次调用 `E()` 获取第二个表达式的代码和结果变量。
- 生成条件判断的三地址代码指令，根据条件是否成立跳转到相应的标签，返回条件表达式的代码和跳转标签。

3.3 表达式处理函数（E）

- 调用项处理函数 `T()` 获取第一个项的代码和结果变量。
- 进入循环，检查当前词法单元是否为加法（TOK_PLUS）或减法（TOK_MINUS）运算符：
  - 如果是，记录运算符并前进到下一个词法单元，再次调用 `T()` 获取下一个项的代码和结果变量。
  - 生成相应的加法或减法三地址代码指令，使用新的临时变量存储结果，更新结果变量和代码。
- 循环结束后，返回表达式的代码和结果变量。

3.4 项处理函数（T）

- 调用因子处理函数 `F()` 获取第一个因子的代码和结果变量。
- 进入循环，检查当前词法单元是否为乘法（TOK_MULTIPLY）或除法（TOK_DIVIDE）运算符：
  - 如果是，记录运算符并前进到下一个词法单元，再次调用 `F()` 获取下一个因子的代码和结果变量。
  - 生成相应的乘法或除法三地址代码指令，使用新的临时变量存储结果，更新结果变量和代码。
- 循环结束后，返回项的代码和结果变量。

3.5 因子处理函数（F）

- 检查当前词法单元：
  - 如果是 `TOK_ID`，表示因子是标识符，直接将标识符作为结果变量，前进到下一个词法单元。
  - 如果是 `TOK_INT8`、`TOK_INT10` 或 `TOK_INT16`，将整数转换为十进制表示（如果是八进制或十六进制），作为结果变量，前进到下一个词法单元。
  - 如果是 `TOK_LPAREN`，前进到下一个词法单元，调用表达式处理函数 `E()` 获取表达式结果，再匹配右括号 `TOK_RPAREN`。
  - 其他情况，抛出语法错误。
- 返回因子的代码和结果变量。

4.三地址代码生成器的数据结构

4.1 指令结构

使用结构体 `Instruction` 表示三地址代码指令，包含四个成员变量：

- `op`：操作符，如 `":="`、`"+"`、`"if"` 等。
- `arg1`：第一个操作数。
- `arg2`：第二个操作数（对于二元运算等情况）。
- `result`：结果变量或标签（对于 `label`、`goto` 等指令）。
  通过 `toString()` 方法将指令转换为字符串形式，方便输出和查看。

4.2 四元式表

使用 `vector<Instruction>` 类型的 `quadTable` 存储生成的三地址代码指令序列，即四元式表。程序在生成三地址代码时，通过 `gen()` 函数将指令添加到该表中。

4.3 临时变量计数器和标签计数器

- `tempCount` 用于生成新的临时变量名，每次调用 `newtemp()` 函数时，`tempCount` 自增，并以 `"t"` 加上计数器值的形式生成临时变量名，如 `t1`、`t2` 等。
- `labelCount` 用于生成新的标签名，每次调用 `newlabel()` 函数时，`labelCount` 自增，并以 `"L"` 加上计数器值的形式生成标签名，如 `L1`、`L2` 等。

4.4 符号表

使用 `map<string, Symbol>` 类型的 `symbolTable` 作为符号表，`Symbol` 结构体包含标识符的名称、类型和值。符号表用于存储程序中出现的标识符及其相关信息，便于在生成代码时进行查找和使用。

4.5 表达式、条件表达式和语句结果结构

- `Expr` 结构体表示表达式结果，包含 `place`（存储表达式结果的变量）和 `code`（生成的代码）。
- `Cond` 结构体表示条件表达式结果，包含 `code`（生成的代码）、`trueLabel`（条件为真时跳转的标签）和 `falseLabel`（条件为假时跳转的标签）。
- `Stmt` 结构体表示语句结果，包含 `code`（生成的代码）和 `nextLabel`（语句执行后的下一个标签）。

5.程序结构的说明

5.1 词法分析部分

词法分析器由 `yylex()` 函数实现，通过遍历输入字符串 `testInput`，根据字符类型识别词法单元。当遇到字母或下划线开头的字符序列时，识别为标识符或关键字；遇到数字时，根据数字开头的特征识别为八进制、十进制或十六进制整数；遇到特定符号时，识别为相应的运算符或标点符号。词法分析器通过更新全局变量 `inputPos` 来跟踪当前处理位置，并将识别出的词法单元通过返回值返回，同时设置全局变量 `yytext` 和 `yyleng` 记录词法单元的文本内容和长度。

5.2 语法分析和代码生成部分

语法分析采用递归下降分析法，通过多个递归子程序实现：

- `S()` 函数处理语句，根据当前词法单元判断语句类型并进行相应处理，调用其他函数处理子结构，并生成对应的三地址代码。
- `C()` 函数处理条件表达式，先处理条件表达式的左右子表达式，根据比较运算符生成条件判断的三地址代码和跳转标签。
- `E()`、`T()`、`F()` 函数分别处理表达式、项和因子，按照语法规则递归处理子结构，生成相应的三地址代码，并返回结果信息。

5.3 辅助函数和数据结构管理部分

- `advance()` 函数用于读取下一个词法单元，更新 `currentToken`。
- `match()` 函数用于匹配期望的词法单元，如果不匹配则抛出语法错误。
- `gen()` 函数用于将三地址代码指令添加到四元式表 `quadTable` 中。
- `newtemp()` 和 `newlabel()` 函数分别用于生成新的临时变量名和标签名。

5.4 错误处理部分

`yyerror()` 函数用于处理语法错误，当程序检测到语法错误时，通过该函数输出错误信息并终止程序执行，错误信息包括错误类型和相关词法单元信息，便于定位和排查问题。