哈爾濱工業大學

编译原理实验报告

题　　目 词法分析与语法分析

专　　业 信息安全

学　　号 1190201012

学　　生 张中界

一、程序功能

1. 识别词法错误

在 lexical.l 文件中用正则表达式标识各种词法单元，其中包括八进制、二进制和十六进制的整型数、浮点数以及标识符等。

（1） 整型

正则表达式为：

INT 0|[1-9][0-9]{0,31}|0[0-7]{1,32}|0[xX][0-9a-fA-F]{1,32}|0[bB][01]{1,32}

识别整型数的错误：

INT_ERROR 0[0-7]*[89]+[0-7]*|0[xX][0-9a-fA-F]*[g-zG-Z]+[0-9a-fA-F]*|0[bB][01]*[2-9]+[01]*

（2） 指数形式的浮点数

正则表达式为：

FLOAT [0-9]+\.[0-9]+|([0-9]+\.[0-9]*|[0-9]*\.[0-9]+)[eE][\+\-]?[0-9]+

（3） 注释

行注释“//”要匹配一行的内容，块注释“/* */”会在遇到”/*“之后匹配第一个”*/”，中间的内容全部视为注释。正则表达式为：

COMMENT ("//".*\n?)|("/*"([*]*(([^*/])+([/])*)*)*"*/")

（4） 正则表达式的顺序

首先匹配注释、空白符和换行符，其次为关键字、整型、浮点型和各种符号，最后为标识符，若标识符仍无法匹配，则输出 A 类错误信息。关键字优先于标识符的匹配，将不合法的标识符视作语法错误。

2. 识别语法错误

在 syntax.y 文件中实现 C--的语法识别。

（1） 定义语法单元和词法单元

将语法单元和词法单元的类型都定义为树节点（下面介绍），词法单元的名称和 flex 文件返回的名称相对应

（2） 语法动作

成功规约一条产生式后，为产生式左边的语法单元创建一个树节点，产生式右边的语法单元为该节点的子节点。

（3） 处理二义性与冲突

通过指定运算符的结合性和优先级消除二义性和冲突。值得一提的是 if-else 匹配的问题，理想情况下，遇到 else 应该移入而不是规约，所以加入一条优先级更低的产生式：Stmt->IF LP Exp RP Stmt %prec LOWER_THAN_ELSE，而 ELSE 和 LOWER_THAN_ELSE 都设置为无结合性。

（4） 错误恢复

遇到一个错误后，我们期望的结果是跳过有错误的代码段继续向下扫描以发现所有错误，而不是直接退出。实现方法：为所有能推导出分号的产生式加入一条新的右部：error SEMI，当出现错误时，将栈顶没有处理完的规则出栈，直到语法分析器回到一个可以移入 error 的状态，然后移入 error，丢弃输入的语法单元直至遇到分号，之后进行正常的语法分析。

3. 语法分析树

语法分析树属于多叉树，使用“孩子-兄弟“的实现方式，每个树节点记录它的左子树 leftchild 和自己的兄弟节点的链表 next。newTree 方法用来创建树节点，若为叶节点需指定行号，若为非叶节点需指明子节点的数量并传入所有子节点的指针。

由于在语法分析的过程中，语法分析树是从下向上创建的，也就是叶节点先创建，需要遍历整棵树来寻找根节点；为了简化这一过程，使用一个数组 IsChild 标识每个节点是否是根节点，每当调用 newTree 创建一个非叶节点时，将参数中的所有子节点标识为非根节点，这样只需遍历 IsChild 数组就可以找到根节点了。

Preorder 方法用来先序遍历语法树，根据节点所在的深度进行相应长度的缩进，打印每个节点的类型和所在行号，对数值型（int float）和标识符额外打印它的内容。

词法分析是建立叶节点的过程，语法分析是建立非叶节点的过程。

二、编译命令

make

若无错误，应生成可执行程序 parser，运行命令为:

./parser text