实验一

171860588 史文泰 171860588@smail.nju.edu.cn
171860572 侯策 171860572@smail.nju.edu.cn

功能实现

- 词法分析
  - 错误的八进制数数通过词法分析检测，并打印$IllegalOctal$指示。
  - 错误的十六进制数通过词法分析检测，并打印$IllegalHex$指示。
  - 错误的 ID(即以数字开头)通过词法分析检测，并打印$IllegalId$指示。
- 语法分析
  - 对于如下语法错误进行指示与恢复：

语法错误	错误恢复产生式
全局变量定义缺少分号	$ExtDef \rightarrow Specifier\ ExtDecList\ error$
全局`struct`定义缺少分号	$ExtDef \rightarrow Specifier\ error$
全局变量定义错误	$ExtDef \rightarrow Specifier\ error\ SEMI$
数组定义缺少右方括号	$VarDec \rightarrow VarDec\ LB\ INT\ error$
数组定义内部非整数	$VarDec \rightarrow VarDec\ LB\ error\ RB$
函数缺少右括号	$FunDec \rightarrow ID\ LP\ VarList\ error$
函数参数列表错误	$FunDec \rightarrow ID\ LP\ error\ RP$
$Stmt$以分号为划分普遍错误	$Stmt \rightarrow error\ SEMI$
$return$语句缺少分号	$Stmt \rightarrow RETURN\ Exp\ error$
if语句条件错误	$Stmt \rightarrow IFLP\ error\ RP\ Stmt$ $\%prec\ LOWERTHANELSE$
if语句条件错误	$Stmt \rightarrow IF\ LP\ error\ RP\ Stmt\ ELSE\ Stmt$
$while$语句条件错误	$Stmt \rightarrow WHILE\ LP\ error\ RP\ Stmt$
局部变量定义错误	$Def \rightarrow Specifier\ error\ SEMI$
局部变量定义缺少分号	$Def \rightarrow Specifier\ DecList\ error$
表达式缺少右括号	$Exp \rightarrow LP\ Exp\ error$
函数调用缺少右括号	$Exp \rightarrow ID\ LP\ Args\ error$

函数参数错误	$Exp \rightarrow ID\ LP\ error\ RP$
[ ]使用内部错误	$Exp \rightarrow Exp\ LB\ error\ RB$
[ ]使用内部缺少右方括号	$Exp \rightarrow Exp\ LB\ Exp\ error$
结构体访问错误	$Exp \rightarrow Exp\ DOT\ error$

- 结构体定义
  - 实验中使用

    $$\%union\{struct\ TreeNode*\ treeNode\};$$

    作为所有结点的类型，用于语法树建立与遍历操作。语法树采用多叉树构建。
  - 实验中使用到的函数如下：

    $$treeNode*\ initLexical(char*\ name,\ char*\ text,\ int\ lineno);$$
    $$treeNode*\ initSyntax(char*\ name);$$
    $$void\ buildTree(treeNode*\ parent,\ int\ childNum,...);$$
    $$void\ printNode(treeNode*\ root,\ int\ lineno);$$

    $initLexical$函数和$initSyntax$函数用于初始化词法结点和语法结点。

    $buildTree$函数使用可变参数，在规约时将产生式右部结点(孩子)插入到产生式左部结点(父亲)中。

    $printNode$函数被递归调用，用于语法正确时语法树的打印。

编译方法

- 实验严格按照提供的$makefile$文件进行编译与执行。命令如下：

  $$make\ clean$$
  $$make\ parser$$
  $$make\ test$$

  若需要添加新的测试文件，则在$Code$文件夹中添加文件，在$makefile$中修改$test$命令中的测试文件即可。

实验感悟

- 对于可以产生ε的产生式，在填写对应操作时应填为$\$\$ = NULL$;这样能够在它作为右部分传递给父亲时，能够被正确判断。或者，直接略去可以产生ε的产生式。
- 对于数组，需要分**定义**与**使用**两部分，定义部分必须为整数，而使用部分则可以为参数。
- 错误恢复机制理解如下：当读入终结符a，发生了错误，$yyerror$函数立即报出错误，在

a前面插入$error$，开始弹栈，直到栈顶中存在$A \to \alpha \bullet error\beta$，此时期待$\beta$，开始读取输入，直至能够规约出$\beta$，其间输入均被忽略。由此完成错误恢复机制。

由上述原理，可以认为：$error$尽量不要写在右部产生式的最右端，这会使得当发生错误后根据恢复产生式立即规约，从而忽略后续的错误标记。