Lab1实验报告

151242060 银加

一、实验环境及编译方法

我使用的是MacOS系统，我在实验的根目录下创建了Makefile文件。在测试时可以在根目录下执行make all即可完成编译并且生成parser语法分析程序。需要注意，与实验指导上稍有区别的是，在mac上“-lfl”需要替换为“-ll”才能编译成功，所以在测试时可能需要根据相应操作系统对Makefile进行修改。

进行测试时，执行make test1…make test10 即可测试相应测试样例的输出，这十个测试样例是实验指导中按顺序出现的十个样例。如果需要添加新的测试样例，可以在根目录下test文件夹中添加新的样例，在Makefile中添加对应语句或直接执行即可。

二、实现功能说明

1.**识别八进制数和十六进制数**：我在实现词法时，增加了识别正确的八进制数和十六进制数的功能，也增加了识别错误的八进制数和十六进制数的功能，识别的正则表达式方式如下：

```
INT8  0[0-7]*
INT8_ERR 0[0-9]*
INT16 ("0x"|"0X")[0-9a-fA-F]*
INT16_ERR ("0x"|"0X")[0-9a-zA-Z]*
```

_ERR结尾的表示识别错误的数，但错误只局限于使用了对应进制中不被允许的符号。如果识别到正确的8或16进制数，会连同10进制作为INT，如果识别到错误的8或16进制数，则会报Error type A的错误。可以通过样例5和6进行测试。

2.**识别指数形式的浮点数：**正则表达式如下：

```
FLOAT_1 {INT10}"."[0-9]+
FLOAT_2 [0-9]*"."[0-9]*[Ee][+-]?[0-9]+
FLOAT_1_ERR ({INT10}".")|("."{INT10})
FLOAT_2_ERR [0-9]*"."[0-9]*[Ee][+-]?
```

其中FLOAT_1是正常形式的浮点数，FLOAT_2是指数形式的浮点数，_ERR后缀的是相应的错误形式，FLOAT_1_ERR能识别小数点前或后没有数字的，FLOAT_2_ERR能识别样例中出现的“e”后面缺失指数的形式，识别到错误后都会直接报Error type A的错误。可以通过样例7和8进行测试。

3.**识别行注释和块注释：**正则表达式如下：

```
COMMENT_1 "//"[^\n]*
COMMENT_2 "/*"(([^\*]*(\*[^\/])?)*)"*/"
COMMENT_ERR "/*"(([^\*]*(\*[^\/])?)*)({COMMENT_2}(([^\*]*(\*[^\/])?)*))+"*/"
```

COMMENT_1识别的是行注释，COMMENT_2识别的是块注释，需要说明的是这里COMMENT_2匹配“ /* ”和“ */ ”采用的是非贪婪匹配，因为flex好像不支持用“ ？ ”直接实现非贪婪匹配，所以我这里实现非贪婪匹配有些复杂，基本原理是在“ /* ”之后要么没有匹配到“ * ”要么匹配到 “ * ”但没有紧跟着匹配到“ / ”，直到匹配到“ */ ”。COMMENT_ERR则能够识别到嵌套的块注释，原理是正则表达式里面用了{COMMENT_2} 来识别嵌套。识别到错误后都会直接报Error type A的错误。可以通过样例9和10进行测试。

4.**记录行号：**我采用的是yylineno选项记录行号，只需在lexical.l文件开头处加上 %option yylineno即可

5.**错误恢复：**我在产生式中若干位置插入了“error”，插入原则与实验指导的相同，以SEMI, RC, RP，RB作为错误恢复的同步符号，全局变量errorCount用于记录错误的总数，包括词法错误和语法错误，当errorCount为零的情况下才会输出语法树。我重写了yyerror函数（syntax.y最后）用于输出符合要求的表达。

6.**语法树的实现**：这一部分主要在node.h 和 node.c 两个文件中，node.h中主要定义了Node这种结构类型，属性name用于记录语法符号，如SEMI、Specifier等等。属性value用于记录实际正则表达式匹配到的字符串（只在语法树的叶子节点即词法单元时用到），属性lineno用于记录行号，child指针指向子节点，bro指针指向兄弟节点。

在node.c中实现了三个函数：

initNode(char* name,char* value)用于生成语法树上的新节点。

addChild(Node* parent, Node* child)用于将child设为parent的子节点，注意：是当前parent存在的子节点的前面插入一个新的子节点（倒序插入），并且将插入的child节点的bro指针指向原来的子节点。将parent节点的行号更新为child节点行号能始终保证打印语法树时语法单元的行号是指该语法单元产生出的所有词素中的第一个在输入文件中出现的行号。

printTree用于按照顺序输出符合格式要求的语法树。注意在将字符串转换为浮点数时使用atof函数，将字符串转换为整数时使用strtol函数，它可同时识别10，8，16进制数。

initNode出现在词法单元或语法单元的action中，例如：
词法：

```
{SEMI}      {yylval.node=initNode("SEMI",yytext);return SEMI;}
```

语法：

```
Specifier : TYPE {$$=initNode("Specifier","");addChild($$,$1);}
```

addChild出现在语法单元的action中，并且注意addChild对产生式右边有多个语法单元的情况下的添加顺序。例如：

```
ExtDef : Specifier ExtDecList SEMI {$$=initNode("ExtDef","");addChild($$,$3);addChild($$,$2);addChild($$,$1);}
```

这种倒序插入是为了保证输出语法树时为正序输出。

printTree只要实现深度优先扫描即可。

三、补充说明

虽然我的语法分析程序能成功识别样例中的所有错误，但我仔细分析了一下，我的程序在语法分析时有一些局限性，因为error插入位置的局限性，使得有些错误无法恢复，或者出现异常，比如说我构建一个新的测试样例，样例中连续两句话没有加分号，此时语法分析器只会报第一处的错误，我利用动态调试进行跟踪，发现第一处发生错误进入错误恢复过程后，因为无法匹配到同步符号，所以不能进行错误恢复，导致第二个语句会被“丢弃”掉，当然也就不会报错。这个问题我没多少头绪，因为有些情况确实找不到合适的同步符号。