Lab1实验报告

赵铭南 161271033

一、完成的功能

- 词法分析：
  - 能够查出C--源代码中词法未定义的字符以及任何不符合词法单元定义的字符；
  - 识别合法的八进制，如035、072；
  - 识别合法的十六进制数，如0x23、0X4a；
  - 识别合法的指数形式的浮点数，如1.2、.2、1.2e+4；
- 语法分析：
  - 能够查出C--源代码中的语法错误；
- 没有词法和语错误的情况，则打印语法树

二、实现方法

- 词法分析：

Flex每成功识别一个词法单元就会调用`create_newnode()`为该词法单元创建一个语法树中的节点，然后返回相应的标识符，节点数据结构为：

```
struct treenode{
    int type;                   \\节点类型：0-终结符号、1-非终结符号、2-空串
    char* name;                 \\节点名称：TYPE、Program等
    int lineno;                 \\行号
    char* value;                \\属性值：INT、FLOAT的值，ID的字符串等
    struct treenode* child;     \\指向子节点的指针
    struct treenode* sibling;   \\指向兄弟节点的指针
};
```

对于ID，直接将识别的字符串存放在`value`中；对于INT、FLOAT的值的处理，先将识别的字符串通过函数（八进制：`otoi()`，十六进制：`htoi()`，浮点数：`atof()`）转化成相应格式的数字，然后以字符串的形式存放在`value`中：

```
{id}            {yylval.node=create_newnode(0,"ID",yylineno,yytext);
return ID;}
{decinteger}    {yylval.node=create_newnode(0,"INT",yylineno,yytext);
return INT;}
{octinteger}    {yylval.node=create_newnode(0,"INT",yylineno,"");
```

```
sprintf(yylval.node->value,"%d",otoi(yytext));  return INT;}
{hexinteger}    {yylval.node=create_newnode(0,"INT",yylineno,"");
sprintf(yylval.node->value,"%d",htoi(yytext));  return INT;}
{float}         {yylval.node=create_newnode(0,"FLOAT",yylineno,"");
sprintf(yylval.node->value,"%f",atof(yytext));  return FLOAT;}
```

对于其他合法的字符，直接创建节点，比如：

```
"="             {yylval.node=create_newnode(0,"ASSIGNOP",yylineno,yytext);
return ASSIGNOP;}
```

对于词法未定义的字符以及任何不符合词法单元定义的字符，直接报错：

```
.               {allright=0;  printf("Error type A at line %d: Mysterious
character \'%s\'.\n",yylineno,yytext);}
```

- 语法分析：

每分析完一个产生式就调用`create_newnode()`给产生式左侧的起始符号创建一个语法树中的节点，并将产生式右侧的符号对于的节点组织在该新节点下（将第一个节点用`insert_child()`接在新节点下，之后的节点用`insert_sibling()`接在前一个节点后），如：

```
Def : Specifier DecList SEMI{
    $$=create_newnode(1,"Def",@$.first_line,"");
    insert_child($$,$1);
    insert_sibling($1,$2);
    insert_sibling($2,$3);
}
;
```

如果产生语法错误，直接调用`yyerror()`输出错误信息。

三、编译运行方法

- 编译：

进入目录`./Code`，终端下输入：`make`，或：

```
flex lexical.l
bison -d syntax.y
gcc tree.c syntax.tab.c main.c -lfl -ly -o parser
```

- 运行：

终端下输入：`./parser [filename]`