编译原理课程Lab1

班级	队伍	组长	组员	任务号	联系邮箱
普通班	2人队	181870290 周心瑜	181860127 姚逸斐	16	181870290@smail.nju.edu.cn

实现功能

本程序能够对使用C--语言书写的源代码进行词法分析和语法分析，并打印分析结果，具体实现了以下这些功能：

- **识别词法错误并按要求输出错误信息**

  借助flex实现了词法分析。在lexical.l中通过正则表达式识别出相应的词法单元，并返回相应的token。重点在于识别INT,FLOAT和ID，对应的正则表达式如下：

  ```
  int_form    0|([1-9]{digital}*)
  commonfloat {int_form}\.{digital}+
  float_form  {commonfloat}
  id_form     {letter_}({letter_}|{digital})*
  ```

  对于那些无法匹配任何正确的词法单元的内容，便认定是词法错误，在所有规则最后加上对于词法错误的处理：设置的lexical_error（记录词法错误的标志位）置为1，打印错误信息。

- **识别语法错误并按要求输出错误信息**

  借助bison完成了语法分析。根据C--语法定义相应的词法单元，指定一些终结符的优先级与结合性，书写相应产生式，重新定义yyerror()函数使之能按照要求输出错误信息。通过编写一些包含error的产生式实现了错误恢复，通过传入不同的msg给yyerror()，完成了一些特殊的语法错误信息打印，例如括号丢失、分号丢失、非法id等。

- **识别八进制数和十六进制数并按要求输出错误信息**

  编写相应的的正则表达式来识别正确和错误格式的八进制数和十六进制数，正确的返回对应词法单元，错误的打印错误信息。

- **若输入文件无任何词法或语法错误，按要求打印语法树**

  语法树部分的代码实现在SyntaxTree.h与SyntaxTree.c中，自定义`TreeNode`数据结构来表示语法树节点，数据结构如下：

  ```
  typedef struct TreeNode{
      char name[32];//名称
      int node_type;//语法单元或词法单元
      int line;//位置
  ```

```
    int value_type;//节点值的类型（INT/FLOAT/TYPE/ID/OCT/HEX）
    union {//相应的值
        unsigned int int_val;
        float float_val;
        char type_val[32];
        char id_val[32];
    };

    struct TreeNode *first_child;//子节点头
    struct TreeNode *last_child;//子节点尾
    struct TreeNode *next;//下一兄弟节点
}TreeNode;
```

create_TreeNode函数负责节点的创建，connect函数负责连接父子节点，show负责按先序遍历打印语法树节点。

编译方法

实验编写环境为：

Ubuntu	GCC	Flex	Bison
16.04	5.4.0	2.6.0	3.0.4

使用makefile进行编译，在命令行输入`make`编译；输入`make test`对Test目录中的文件进行测试，也可以使用`./parser ../Test/test.cmm`对某一指定文件进行测试。

其他

- 实验中遇到的问题与解决方法
    - Exp -> Exp LB Exp RB. 为了避免冲突，引入新的状态 *error_Exp*.
    - Stmt -> Exp error vs. Stmt -> IF LP Exp RP Exp ELSE. 通过-d同时对照syntax.output，发现接受ELSE时率先发生error，而导致前者具体错误会被同一行忽略。故在判断IF中Stmt是否缺少';'时加入后者。
    - 负数的处理出现错误。在语法分析树中，需要将整型值的输出设为 %u，类型应该为unsigned int
- 感谢https://github.com/massimodong/compilers-tests提供的测试数据