实验一 词法分析器与语法分析器

姓名：肖丹妮　学号：201220199　联系邮箱：3165372798@qq.com

实验环境

1) GNU Linux Release: Ubuntu 20.04, kernel version 5.15.0-48-generic；

2) GCC version 7.5.0；

3) GNU Flex version 2.6.4；

4) GNU Bison version 3.5.1。

实现功能

必做内容

词法分析

- 能够进行词法分析，并识别词法错误（错误类型A）
  - C--词法中未定义的字符，与词法单元最后一条规则“ . ”匹配，以如下格式报错：
    Error type A at Line [行号]: Mysterious characters '[未定义字符]'.
  - 不符合C--词法单元定义的字符，以如下形式报错：
    - 匹配词法单元illegal_INT： Error type A at Line [行号]: Illegal INT '[不合法字符]'.
    - 匹配词法单元illegal_FLOAT： Error type A at Line [行号]: Illegal FLOAT '[不合法字符]'.

语法分析

- 能够进行语法分析，并且识别语法错误（错误类型B）
  - 若有错误，每行错误均以如下方式报错：
    - Error type B at Line [行号]: [说明文字].

语法树

- 若即无词法错误也无语法错误，按要求打印语法树

选做内容

八进制与十六进制

- 合法的八进制与十六进制：
  - 匹配词法单元OCTAL、HEXAL
- 不合法的八进制与十六进制：
  - 匹配词法单元illegal_OCTAL、illegal_HEXAL，并且输出类型A的错误信息
    Error type A at Line [行号]: Illegal OCTAL '[不合法字符]'.
    Error type A at Line [行号]: Illegal HEXAL '[不合法字符]'.

指数形式的浮点数

- 合法的浮点数：
  - 匹配词法单元expFLOAT

- 不合法的浮点数：
    - 匹配词法单元illegal_expFLOAT，并且输出类型A的错误信息
      Error type A at Line [行号]: Illegal expFLOAT '[不合法字符]'.

注释

- “//”：匹配词法单元ANNO
- “/*...*/”：匹配词法单元l_ANNO，若遇到非法的注释，以以下方式报错：
  Error type B at Line [行号]: No match annotator "/*".

编译命令

make、make clean、make test。

实现细节

语法树：

数据结构

- lexical.l文件中定义数据结构——多叉树如下：

```
typedef union Value{
    int ival;
    double dval;
}Value;//数值
typedef struct TreeNode{
    int type;//1-词法单元/0-语法单元
    int column;//行号
    char name[32];//名称
    char strval[32];//内容(词法单元对应为词素/语法单元的值为“null")
    union Value val;//
    struct TreeNode* firstchild, * nextsibling;
    struct TreeNode* parent;
}TreeNode;
```

- syntax.y文件中声明定义语法树的根节点root：

```
struct TreeNode* root = NULL;
```

- 使用%union{...}定义所有属性值类型以及变量 yylval的类型为TreeNode*

```
%union {
    struct TreeNode* nodep;
}
```

节点的初始化、子节点的插入、语法树的打印

在lexical.l和syntax.y中分别定义两个标志：lexicalerror与syntaxerro，这两者初始化都为0（含义是0/1代表没有/有语法错误）。若在完成词法分析与语法分析之后，lexicalerror和syntaxerro仍然都为0，则代表没有词法错误且没有语法错误，就打印语法树。

- 节点的初始化——TreeNodeSet()：
    - 实现：初始化结点的值，需要注意INT、OCTAL、HEXAL、FLOAT、expFLOAT的数值设置。
    - 使用：在词法分析的规则部分，添加动作将yyval.nodep初始化，返回词法单元，语法分析就可以通过yylex()获得终结符的属性值；在语法分析的规则部分，添加语义动作将终结符的属性值$$初始化。
- 子节点的插入——TreeNodeInsert()：

    - 实现：将根节点指针、子节点指针用作参数传给函数，不定参数使用va_list、va_start、va_arg、va_end获得。
    - 使用：在语法分析的规则部分，添加语义动作，将终结符的属性值$$初始化之后将子结点的属性值$1、$2、$3、……依次与$$建立联系。
- 语法树的打印——TreePrint()：
    - 实现：具有两个参数当前子树根节点root、需要缩进的层数retra_n。每次打印当前子树的根节点后，递归打印以子结点为根的子树。
    - 使用：在main.c中进行完词法分析与语法分析后，若没有错误就从全局变量root也就是Program开始递归打印语法树。

语法错误信息的打印

由syntax.y中的yyerror()完成：
重写yyerror()函数，将默认的打印信息syntax error忽略掉，使得函数根据设定的不定参数进行信息的打印。

遇到的问题与解决方法

- 合法词法单元与非法词法单元的表示：
  尤其是浮点数、指数形式的浮点数的正则表达式的书写，需要考虑全面。
- 在词法分析时的词法单元优先级问题：
  考虑各个规则的先后顺序，比如{ID}需要放在{STRUCT}{WHILE}等等的后面。
- 在语法分析中的错误恢复与二义性和冲突处理：
  在写错误恢复的过程中，很容易出现移入/归约冲突、归约/归约冲突，需要根据output文件找到冲突出现的位置，然后再改写错误恢复的式子。