南京大学本科生实验报告

课程名称： **编译原理**

学号 191220029 **姓名** 傅小龙

1.实验名称

词法分析和语法分析

2.实验内容

2.1词法错误

使用`GNU Flex`词法分析工具识别C--中定义的词法。对于未定义或不符合C--词法的字符使用如下规则匹配：

```
. {printf("Error type A at Line %d: Undefined identifier \'%s\'.\n", yylineno,
yytext); right = 0;}
```

这一规则是所有匹配规则中的最后一条规则，以匹配前面所有规则无法识别的字符。

2.2 语法错误

2.2.1 语法树构建

语法树结点数据结构如下：

```
typedef enum {
        Int, Float, Type, Id, Relop, Rkw, Null, Gu
}wType;

typedef struct Node{
    char* name;
    wType type;
    int lineNum;
    struct Node* child;
    struct Node* next;
    struct Node* parent;
    union {
        unsigned intVal;
        float floatVal;
        char strVal[40];
    }val;
}Node;
```

结点类型由属性`type`给出，包括整型、浮点型、类型关键字、ID、运算符、其余关键字、空串、语法单元。

在构建词法单元结点时，其`*child`，`*parent`，`*next`指针初始化为空值，将属性`val`赋为用户输入的对应内容。构建语法单元结点时`*child`指针指向其首个子节点，`*parent`指针指向其父节点，`*next`指针指向其下一个兄弟结点。具体实现详见`tree.h`和`tree.c`文件。

词法单元结点在词法分析中成功匹配合法字符的规则后生成，调用`tree.h`中定义的`createNode(...)`函数。例如对十进制数的词法单元结点的生成：

```
([1-9]{DIGIT}*)|0 {yylval.node = createNode("INT", Int, yytext); return INT;}
```

语法单元结点在语法分析时根据`syntax.y`文件中给出的语法规则生成。具体实现详见`lexcial.l`和`syntax.y`文件。

2.2.2 识别八进制数和十六进制数

八进制数和十六进制数的匹配规则如下：

```
0[0-7]+ {yylval.node = createNode("OCT", Int, yytext); return INT;}
0{DIGIT}+ {printf("Error type A at Line %d: Invalid octal number \'%s\'.\n",
yylineno, yytext); correct = 0; yylval.node = createNode("INT", Int, yytext);
return INT;}
0[xX][0-9a-fA-F]+ {yylval.node = createNode("HEX", Int, yytext); return INT;}
0[xX][0-9a-zA-Z]+ {printf("Error type A at Line %d: Invalid heximal number
\'%s\'.\n", yylineno, yytext); correct = 0; yylval.node = createNode("INT", Int,
yytext); return INT;}
```

2.2.3 识别指数形式的浮点数

指数形式和普通形式的浮点数匹配规则如下：

```
({DIGIT}*\.?{DIGIT}+|{DIGIT}+\.)[eE][+-]?{DIGIT}+ {yylval.node =
createNode("FLOAT", Float, yytext); return FLOAT;}
({DIGIT}*\.?{DIGIT}+|{DIGIT}+\.)[eE][+-]?({DIGIT}*\.{DIGIT}*)? {printf("Error
type A at Line %d: Invalid float number \'%s\'.\n", yylineno, yytext); correct =
0; yylval.node = createNode("FLOAT", Float, yytext); return FLOAT;}
{DIGIT}+\.{DIGIT}+ {yylval.node = createNode("FLOAT", Float, yytext); return
FLOAT;}
```

需注意的是非指数形式的浮点数小数点前后必须都有数字。指数形式浮点数的指数部分只能是整数。

2.2.4 识别注释

单行注释和段落注释匹配规则如下：

```
WHITESPACE [\t ]+
\/\/.* {}
\/\*.*\*\/.*\*\/ {printf("Error type A at Line %d: Invalid comment \'%s\'.\n",
yylineno, yytext); correct = 0;}
"/*"(([^\*]|\*+[^\*\/]|{WHITESPACE})*\n?)*\*+\/ {}
```

需要注意的是C--不支持嵌套段落注释。

2.2.5 重写yyerror

对于不同的语法错误需要给出对应的提示信息，故这里将`yyerror`改写为一个带有变长参数列表的函数`int yyerror(const char *s, ...)`.参数`const char *s`用于给出提示信息，变长参数列表为可选参数`int line_num`，用于给出错误位置的行数。如果没有指定错误位置的行数，那么`line_num`的值默认为0，将采用当前正在处理的行号`yylineno`作为错误位置输出。

3. 编译本程序

在提交文件的根目录下，`/base_file`文件夹内为基础版本的词法语法分析器，未实现：1.1识别八进制数和十六进制数/1.2识别指数形式的浮点数/1.3识别“//”和“/.../”形式的注释；`/extensive_file`文件夹内为实现实验手册中所有基础和选做要求的词法语法分析器。请使用这两个文件夹下的`Makefile`编译生成不同版本的可执行对象。

在`/base_file`或`/extensive_file`目录下的终端输入`make`指令即可生成可执行目标`parser`，使用形如`./parser <test file>`的指令即可对参数`<test file>`进行词法语法分析.