# 编译器构造实验

## Lab I - 词法分析与语法分析

本次实验是在给出的示例代码的基础上进行修改。

### 实现的功能

```
(0x|0X)[0-9a-fA-F]*[g-zG-Z][0-9a-fA-F]* { printf("Error type A at Line %d: Illegal
hexadecimal number \'%s\'\n", yylineno, yytext);
              lexError++; }
{HEX}       { yylval = createNode("INT", ENUM_LEX_INT, yylineno, 0, NULL);
              yylval->intVal = hexstrToi(yytext);
              return INT; }
0[0-7]*[89][0-7]*     { printf("Error type A at Line %d: Illegal octal number
\'%s\'\n", yylineno, yytext);
              lexError++; }
{OCT}       { yylval = createNode("INT", ENUM_LEX_INT, yylineno, 0, NULL);
              yylval->intVal = octstrToi(yytext);
              return INT; }
{DEC}       { yylval = createNode("INT", ENUM_LEX_INT, yylineno, 0, NULL);
              yylval->intVal = atoi(yytext);
              return INT; }
({digit}*\.{digit}+|{digit}+\.)[eE] {
    printf("Error type A at Line %d: Illegal floating point number \"%s\"\n",
yylineno, yytext);
    lexError++;
}
```

其中添加了三个pattern：

- 第一个pattern

  ```
  (0x|0X)[0-9a-fA-F]*[g-zG-Z][0-9a-fA-F]*
  ```

  这个pattern可以识别非法的十六进制输入，同时输出错误

  ```
  Error type A at Line 4: Illegal hexadecimal number '0x3G'
  ```

- 第二个pattern

```
0[0-7]*[89][0-7]*
```

这个pattern可以识别非法的八进制输入，同时输出错误

```
Error type A at Line 3: Illegal octal number '09'
```

- 第三个pattern

```
({digit}*\.{digit}+|{digit}+\.)[eE]
```

该pattern可以识别非法的float输入，同时输出错误

```
Error type A at Line 3: Illegal floating point number "1.05e"
```

实验过程中发现错误类型A和B可能会同时出现，尝试在yyrestart(f)和yyparse()两个函数之间根据变量lexError的个数决定是否进行语法分析。但是发现词法分析和语法分析应该是同时进行的？lexError的值在yyparse()执行之前不会发现任何改变。因此，暂未能实现：发现词法错误时不再进行语法分析。

## 编译

Makefile未做更改。

也可逐条输入命令进行编译。

Code文件夹下

```
flex lexical.l
bison -d syntax.y
```

根目录下

```
gcc Code/main.c Code/syntax.tab.c Code/Tree.c -lfl -ly -o parser
```