编译原理实验一：词法分析

一、实验要求：

1.输出基本的词法分析结果

2.输出未定义的标识符

3.识别单行注释

4.识别八进制数和十六进制数

5.识别指数形式浮点数

6.识别多行注释

二、实验分工：

实验由我们小组三个人共同完成。其中，代码的基础框架编写由杨尚谕完成，主函数的读文件功能由欧铭伟完成，正则表达式的编写由宋昊完成。

三、实验环境：

MacOS(Dev) + Linux(Remote)：

首先尝试在本地(Mac OS环境)下使用VScode开发，在Mac环境下使用flex编译的指令应该是

```
gcc lex.yy.c -ll -o scanner  -> gcc lex.yy.c -lfl -o scanner
```

使用ssh连接远程开发主机失败，最后改为本地代码push到github，在云主机上pull代码

Windows(Dev) + Linux(Remote):

四、开发：

4.1开发流程概述

首先我们需要写出cf.l文件，然后使用flex将其编译为一个可执行的c文件。这个c文件的用处就是将字符序列转化为token的过程。

开发完成之后

4.2.l源文件文件开发

flex的结构定义如下：

定义部分:这一部分由C语言代码,模式的宏定义等组成

模式的宏定义是一个正则表达式定义，正则表达匹配如下

模式	解 释
x	配置单个字母x
.	匹配除换行符'/n'之外的任意字符
[xyz]	匹配x、y或z
[abj-oz]	匹配a、b、z及j至o之间的字母
[^A-Z]	除大写字母A-Z之外的其它字符
[^A-Z/n]	除大写字母A-Z和换行符之外的其它字符
r*	匹配0个或多个r
r+	匹配1个或多个r
r?	匹配0个或1个r

```
//**c语言代码**
%top{
#include<math.h>
#include<string.h>
int col = 1;
} // **这些可以全部转化为c语言代码 所以可以在这里进行全局变量的命名 引入头文件等等**
%option yylineno //**表示符号所在行号**
//模式宏定义是一个正则表达式的定义 书中的token也是在这一步进行定义
SEMI ;
COMMA ,
ASSIGNOP =
RELOP >|<|>=|<=|==|!=
PLUS \+
MINUS -
STAR \*
DIV "/"
AND &&
OR \|\|
DOT \.
NOT !
TYPE (int)|(float)
LP "("
RP ")"
LB "["
RB "]"
LC "{"
RC "}"
STRUCT (struct)
RETURN (return)
IF (if)
ELSE (else)
```

```
WHILE (while)
INT (0|[1-9][0-9]*)
INT8 0[0-7]*
INT16 0[xX][0-9a-fA-F]+
INT8_ERROR 0[0-9]+
INT16_ERROR 0[xX][0-9a-zA-Z]+
NORMAL_FLOAT ([0-9]*[\.][0-9]+|[0-9]+[\.][0-9]*)
SCI_FLOAT {NORMAL_FLOAT}[eE][+-]?{INT}
FLOAT ({NORMAL_FLOAT}|{SCI_FLOAT})
FLOAT_ERROR ({NORMAL_FLOAT}|{INT})[eE]([0-9a-zA-Z]+|{NORMAL_FLOAT})?
ID [A-Za-z_][A-Za-z0-9_]*
```

规则部分

 规则部份是LEX源文件的核心部份，它包括一组模式和在生成分析器识别相应模式后对相应模式进行处理的C语言动作(Action),格式如下。

```
//略 见代码
```

用户附加C语言部分

将文件读入,进行分析。

```
int main(int argc, char* argv[]) //接受文件输入的main函数
{
  if (argc > 1){
    if(!(yyin = fopen(argv[1], "r"))){
      perror(argv[1]);
      return 1;
    }
    while (yylex() != 0);
  }
  return 0;
}
```

4.3Makefile文件编写

考虑到每次书写指令比较麻烦，且语义不清晰，统一使用Makefile进行编译。

```
run:  运行所有的测试文件
   ./scanner test1.cmm
   ./scanner test2.cmm
build: //编译为c文件 生成scanner (MacOS环境下为 -ll Linux环境下为 --lfl)
  gcc lex.yy.c -ll -o scanner
flex: //将flex文件编译为c文件
  flex cf.l
```

4.4协同开发

考虑到三个人同期开发，我们选择将代码存储到gitHub，并且每次写作业时都拉取自己的分支，比如feature/ysy。然后由先完成的组员将自己的代码上传到master，后续完成的同学自行解合并的冲突，code review后merge到master。

链接：https://github.com/faker0817/Compilation

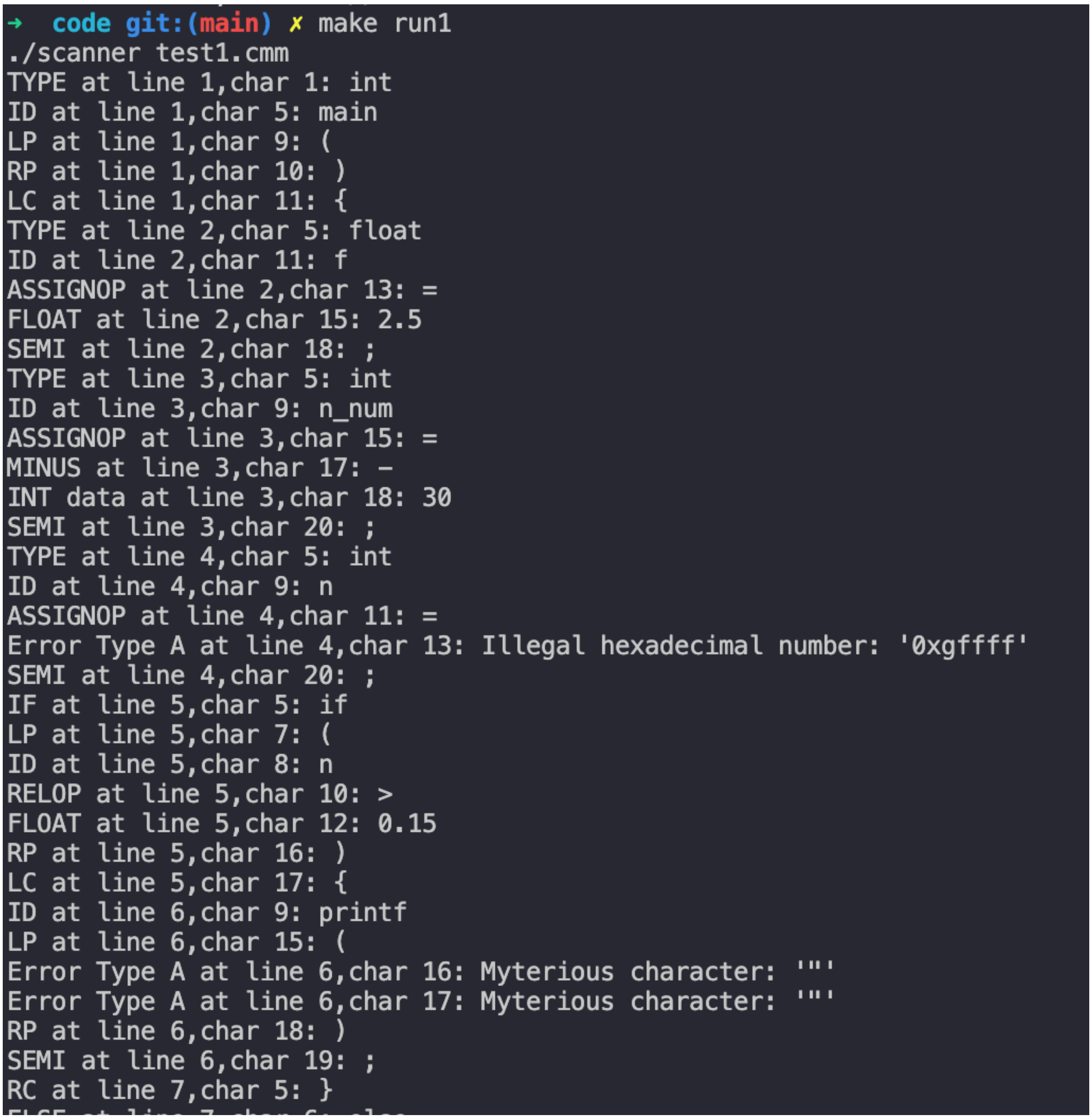
五、实验结果：

***编译*：**

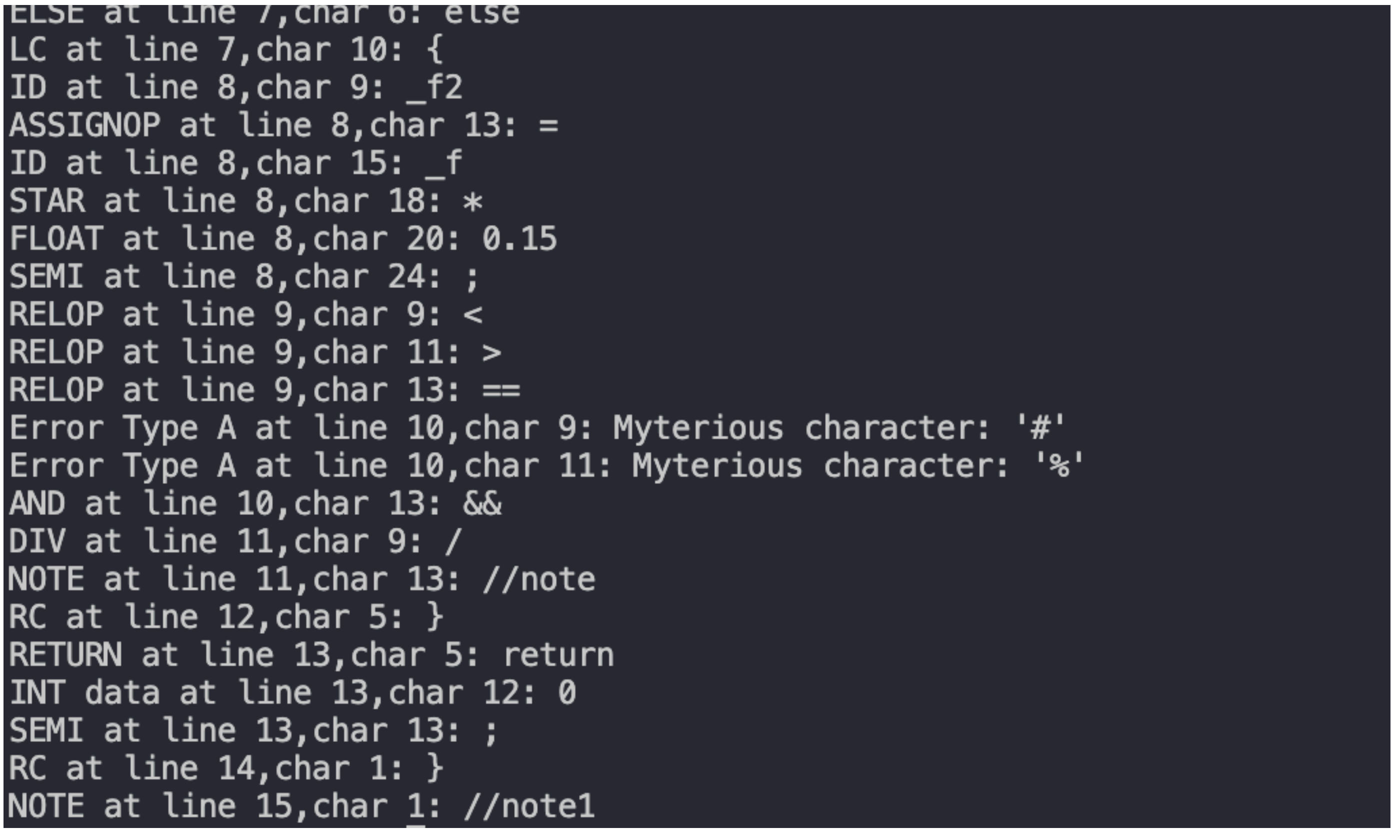
make flex

make build

***测试一运行*：**

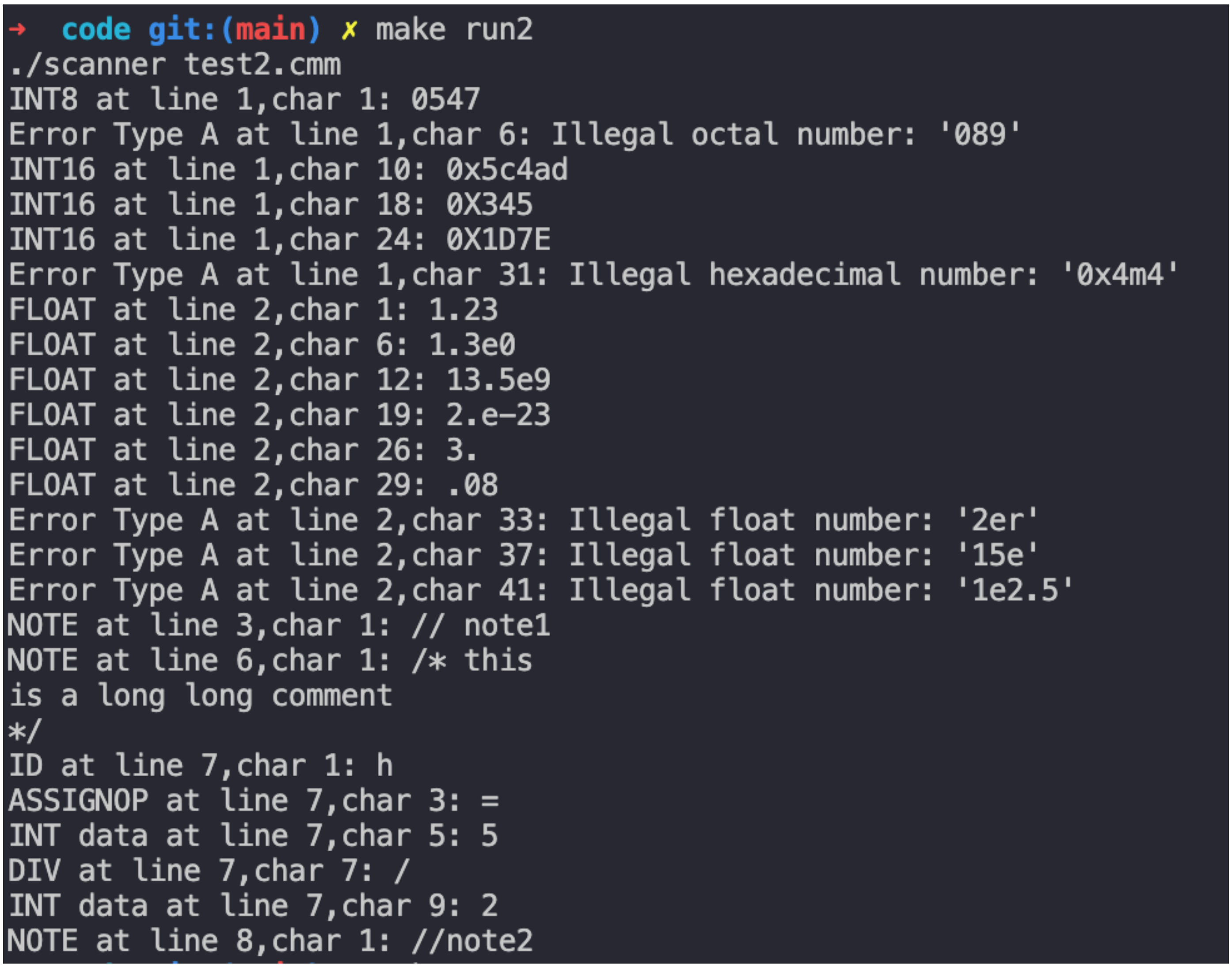


```
➜  code git:(main) ✗ make run1
./scanner test1.cmm
TYPE at line 1,char 1: int
ID at line 1,char 5: main
LP at line 1,char 9: (
RP at line 1,char 10: )
LC at line 1,char 11: {
TYPE at line 2,char 5: float
ID at line 2,char 11: f
ASSIGNOP at line 2,char 13: =
FLOAT at line 2,char 15: 2.5
SEMI at line 2,char 18: ;
TYPE at line 3,char 5: int
ID at line 3,char 9: n_num
ASSIGNOP at line 3,char 15: =
MINUS at line 3,char 17: -
INT data at line 3,char 18: 30
SEMI at line 3,char 20: ;
TYPE at line 4,char 5: int
ID at line 4,char 9: n
ASSIGNOP at line 4,char 11: =
Error Type A at line 4,char 13: Illegal hexadecimal number: '0xgffff'
SEMI at line 4,char 20: ;
IF at line 5,char 5: if
LP at line 5,char 7: (
ID at line 5,char 8: n
RELOP at line 5,char 10: >
FLOAT at line 5,char 12: 0.15
RP at line 5,char 16: )
LC at line 5,char 17: {
ID at line 6,char 9: printf
LP at line 6,char 15: (
Error Type A at line 6,char 16: Myterious character: '"'
Error Type A at line 6,char 17: Myterious character: '"'
RP at line 6,char 18: )
SEMI at line 6,char 19: ;
RC at line 7,char 5: }
ELSE at line 7,char 6: else
```



```
ELSE at line 7,char 6: else
LC at line 7,char 10: {
ID at line 8,char 9: _f2
ASSIGNOP at line 8,char 13: =
ID at line 8,char 15: _f
STAR at line 8,char 18: *
FLOAT at line 8,char 20: 0.15
SEMI at line 8,char 24: ;
RELOP at line 9,char 9: <
RELOP at line 9,char 11: >
RELOP at line 9,char 13: ==
Error Type A at line 10,char 9: Myterious character: '#'
Error Type A at line 10,char 11: Myterious character: '%'
AND at line 10,char 13: &&
DIV at line 11,char 9: /
NOTE at line 11,char 13: //note
RC at line 12,char 5: }
RETURN at line 13,char 5: return
INT data at line 13,char 12: 0
SEMI at line 13,char 13: ;
RC at line 14,char 1: }
NOTE at line 15,char 1: //note1
```

测试二运行:



```
➜  code git:(main) ✗ make run2
./scanner test2.cmm
INT8 at line 1,char 1: 0547
Error Type A at line 1,char 6: Illegal octal number: '089'
INT16 at line 1,char 10: 0x5c4ad
INT16 at line 1,char 18: 0X345
INT16 at line 1,char 24: 0X1D7E
Error Type A at line 1,char 31: Illegal hexadecimal number: '0x4m4'
FLOAT at line 2,char 1: 1.23
FLOAT at line 2,char 6: 1.3e0
FLOAT at line 2,char 12: 13.5e9
FLOAT at line 2,char 19: 2.e-23
FLOAT at line 2,char 26: 3.
FLOAT at line 2,char 29: .08
Error Type A at line 2,char 33: Illegal float number: '2er'
Error Type A at line 2,char 37: Illegal float number: '15e'
Error Type A at line 2,char 41: Illegal float number: '1e2.5'
NOTE at line 3,char 1: // note1
NOTE at line 6,char 1: /* this
is a long long comment
*/
ID at line 7,char 1: h
ASSIGNOP at line 7,char 3: =
INT data at line 7,char 5: 5
DIV at line 7,char 7: /
INT data at line 7,char 9: 2
NOTE at line 8,char 1: //note2
```

六、实验反思：

有时间想尝试一下不使用flex进行词法分析:-)