

中国矿业大学计算机学院

## 系统软件开发实践报告

课程名称：系统软件开发实践

实验名称：实验二 利用 Flex\_Bison 构造编译器

学生姓名：陈柏翰

学生学号：02140385

专业班级：计算机科学与技术 2014-4 班

任课教师：张博老师

## 实验二 利用 Flex/Bison 构造编译器

### 一 实验要求

1. 编写 LEX 源文件，实现 C 语言子集的词法分析功能，最后上机调试。
2. 要求编写一个测试程序，以给定的测试文件作为输入，输出运行结果到输出文件中。

### 二 实验内容

给定 C 语言的一个子集，具体内容如下：

1. 下面是语言的关键字：

else if switch for int float return void while

所有的关键字都是保留字，并且必须是小写。

2. 下面是专用符号：

+ - \* / < <= > >= == != = ; , ( ) [ ] { } /\* \*/

3. 其他标记是标识符（ID）和数字（NU），通过下列正则表达式定义：

ID = letter letter\*

NUM = digit digit\*

letter = a|..|z|A|..|Z

digit = 0|..|9

注：小写和大写字母是有区别的。

4. 空格由空白、换行符和制表符组成。空格通常被忽略，除了它必须分开 ID、NUM 关键字。

5. 注释用通常的 C 语言符号/\* ... \*/围起来。注释可以放在任何空白出现的位置(即注释不能

放在标记内)上，且可以超过一行。注释不能嵌套。

### 三 LEX 程序功能描述

LEX 程序实现了 C 语言子集的词法分析功能

#### 四 LEX 程序结构描述（代码标红处是对原代码的更正）

letter [a-zA-Z\_] 定义字母 letter

digit [0-9] 定义数字 digit

ID {letter}({letter})\* 定义单词 ID 由若干个字母组成

NUM {digit}({digit})\* 定义数字串 NUM 由若干个数字组成

B {letter}({digit}|{letter})\* 定义标识符 B 由数字或字母组成

%{

int nchar, nword, nline; nchar 字符数 nword 单词数 nline 行数

int line=1; line 为当前行数 初始化为 1

%}

%%

"else"|"if"|"else if"|"switch"|"for"|"int"|"float"|"return"|"void"|"while"  
{nword++;nchar+=yyleng;printf("第 %d 行:\t",line);printf("关键字 : %s\n",yytext);}

若匹配上 else int float 等上述的关键字：单词数+1；字符数增加相应的个数；

输出 “第 line 行 yytext \n” ；

{B} {nword++;nchar+=yyleng;printf("第 %d 行:\t",line);printf("标识符 : %s\n",yytext);}

若匹配上 B 标识符：单词数+1；字符数增加相应的个数；

输出 “第 line 行 标识符 : yytext \n” ；

{NUM} {nword++;nchar+=yyleng;printf("第 %d 行:\t",line);printf("数字 : %s\n",yytext);}

若匹配上 NUM 数字：单词数+1；字符数增加相应的个数；

输出 “第 line 行 数字 : yytext \n” ；

```
\+|\-|\*|\/|\<|\>|\=|\;|\,|\(|\)|\[\|\]\{\|\} {nchar+=yyleng;printf("第 %d 行:\t",line);printf("专用符号： %s\n",yytext);}
```

若匹配上 + - \* / 等专用符号：字符数增加相应的个数；

输出 “第 line 行 专用符号：yytext \n” ；

```
\<|\=|\>|\=|\=|\!=|\^|\*|\*\^ {nword++;nchar+=yyleng;printf("第 %d 行:\t",line);printf("专用符号： %s\n",yytext);}
```

若匹配上 < = > 等专用符号：字符数增加相应的个数；

输出 “第 line 行 专用符号：yytext \n” ；

```
[\t]+ {nchar++;}
```

若匹配上制表符：字符数+1；无输出；

```
\n { nline++;line++;nchar++; }
```

若匹配上回车\n：字符数+1；行数+1；当前行数 line+1；无输出；

```
[^\t\n]+ { nword++;nchar+=yyleng;printf("第 %d 行:\t",line);printf("其他符号： %s\n",yytext);}
```

若匹配上其他符号：字符数增加相应的个数；

输出 “第 line 行 其他符号：yytext \n” ；

```
%%
```

```
void main()
```

```
{
```

```
yylex();          调用 yylex 函数
```

```
printf("字符数： %d\t 单词数： %d\t 行数： %d\n", nchar, nword,nline);
```

```
}                最后输出字符数、单词数、行数
```

```
int yywrap()
```

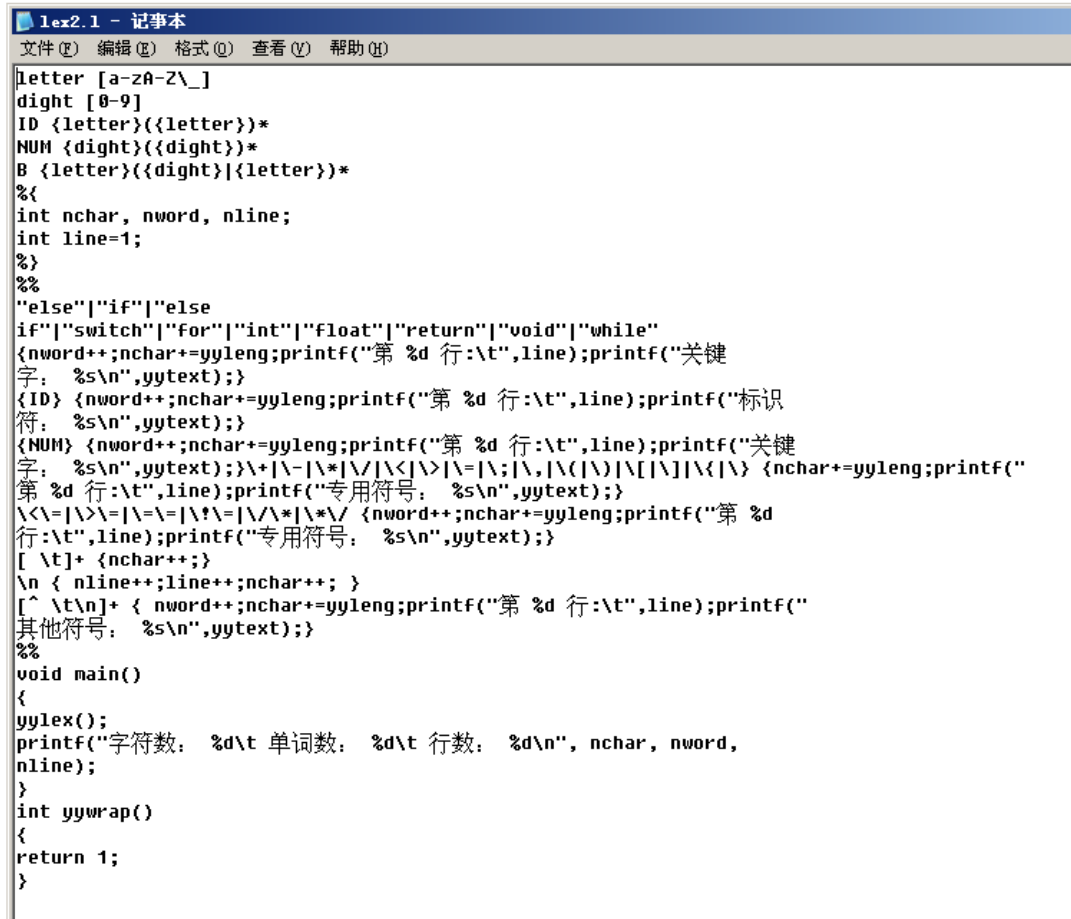
```
{
```

```
return 1;
```

```
}
```

## 五 实验步骤

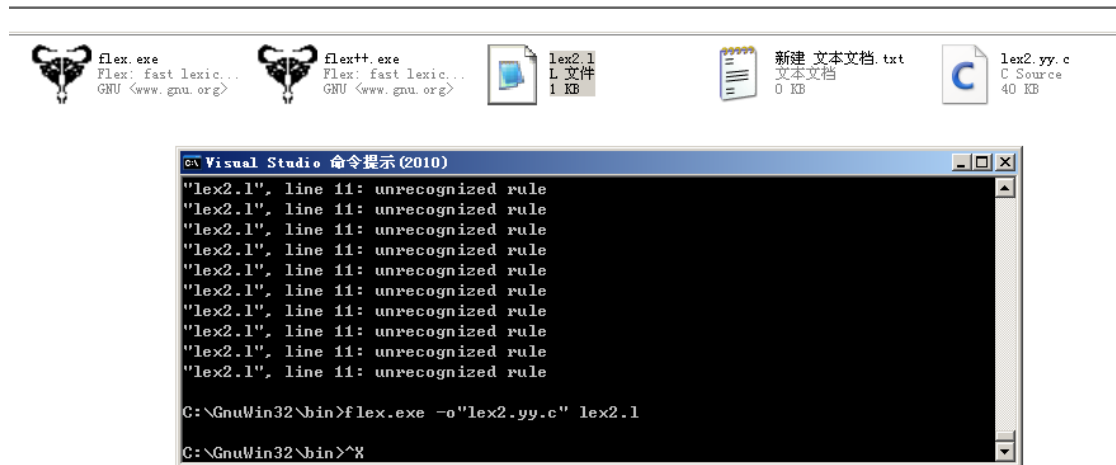
### 1. Lex2 源代码



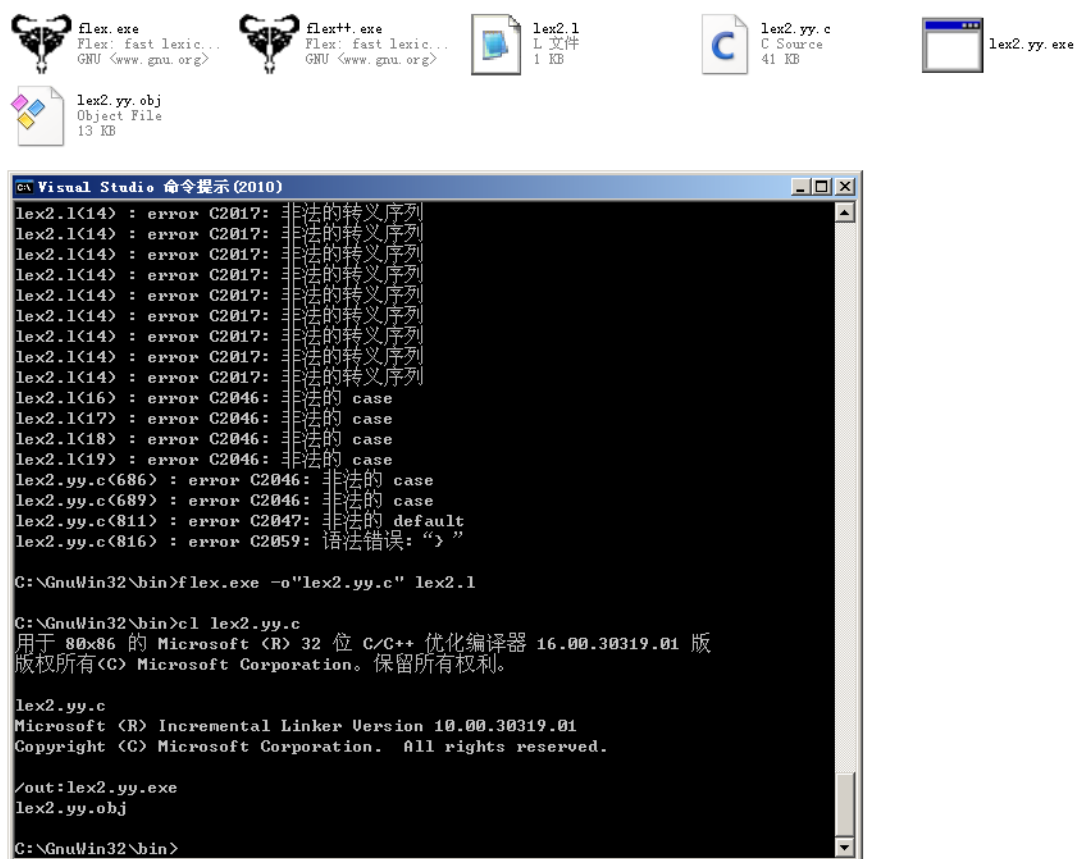
```
lex2.1 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

letter [a-zA-Z\_]\_
dight [0-9]
ID {letter}({letter})*
NUM {dight}({dight})*
B {letter}({dight}|{letter})*
%{
int nchar, nword, nline;
int line=1;
}%
%%
"else"|"if"|"else
if"|"switch"|"for"|"int"|"float"|"return"|"void"|"while"
{nword++;nchar+=yytext;printf("第 %d 行:\t",line);printf("关键
字: %s\n",yytext);}
{ID} {nword++;nchar+=yytext;printf("第 %d 行:\t",line);printf("标识
符: %s\n",yytext);}
{NUM} {nword++;nchar+=yytext;printf("第 %d 行:\t",line);printf("关键
字: %s\n",yytext);}
{nchar+=yytext;printf("第 %d 行:\t",line);printf("专用符号: %s\n",yytext);}
{B} {nchar+=yytext;printf("第 %d
行:\t",line);printf("专用符号: %s\n",yytext);}
[ \t]+ {nchar++;}
\n { nline++;line++;nchar++; }
[^\t\n]+ { nword++;nchar+=yytext;printf("第 %d 行:\t",line);printf("
其他符号: %s\n",yytext);}
%%
void main()
{
yytext();
printf("字符数: %d\t 单词数: %d\t 行数: %d\n", nchar, nword,
nline);
}
int yywrap()
{
return 1;
}
```

2. 输入>flex.exe -o"lex2.yy.c" lex2.l,生成 lex2.yy.c 文件



3. 输入>cl lex2.yy.c,生成 lex2.yy.exe 和 lex2.yy.obj 两个文件



4. 输入>lex2.yy.exe < 1-1.cpp 得到输出结果

```
C:\GnuWin32\bin>lex2.yy.exe < 1-1.cpp
第 1 行: 其他符号: #include
第 1 行: 标识符: iostream
第 2 行: 标识符: using
第 2 行: 标识符: namespace
第 2 行: 标识符: std
第 3 行: 关键字: int
第 3 行: 标识符: main
第 4 行: 标识符: cout
第 4 行: 其他符号: "Hello!"
第 4 行: 其他符号: "<<endl"
第 5 行: 标识符: cout
第 5 行: 其他符号: "Welcome"
第 5 行: 标识符: to
第 5 行: 其他符号: "c++!"
第 5 行: 其他符号: "
第 5 行: 标识符: endl
第 6 行: 关键字: return
字符数: 103 单词数: 17 行数: 5
```

## 六 实验总结

1. 你在编程过程中遇到了哪些难题？你是怎么克服的？

在实验过程中，在使用 dos 命令时候，遇到了一些问题，以及 lex2 源代码的编程上，出现了错误。

我先是自己动手调试，尝试各种方法找出错误，后来通过老师的帮助，顺利解决了所有的困难，完成了实验。

2. 你对你的程序的评价？

本次实验完成的程序只是一个简单的语法分析器，甚至不具备匹配浮点数等基本类型的功能，但是比第一次实验的语法分析器显然进步了许多，我认为在语法分析器的编程上，还有很多东西要学习与实践。

3. 你的收获有哪些

通过本次实验，我先是编写 LEX 源文件，实现 C 语言子集的词法分析功能，然后编写一个测试程序，以给定的测试文件作为输入，输出运行结果到输出文件中。

我进一步掌握了 Yacc 与 Lex 基本使用方法，此外还学习了在 LINUX 系统下完成该实验的方法。并且在老师的答疑下解决了一些重点难点，受益匪浅。