

编译技术实验课

主讲教师：陈安龙

课程实验内容简介

实验一：词法分析（4学时）

用C语言编写程序实现：1) 识别给定程序中所有的整数、浮点数及字符串常量；2) 识别给定程序中所有的符号、标识符及关键字；用 flex 完成上述任务；

实验二：递归下降语法分析（4学时）

实验三：LR语法分析（4学时）

实验四：语义分析及 LLVM 代码生成 (4 学时)

第一次实验内容：

1. 以 test_cases 中的文件作为输入，编写程序（c/c++）
 - (1)识别程序中所有的常数、运算符、界符、标识符及关键字
 - (2)将所编写程序命名为“**man_lex.c**”
2. 学习所提供资料中的简单 flex 源程序
试着修改其中的规则及代码，再运行之
3. 用 flex 完成 任务1，并将所编写flex源文件命名为 **auto_lex.l**

实验安排要求：

1. 每位同学必须参与

2. 所有文件都以 **utf-8** 进行统一编码保存！！

(检查环境为Linux，如出现乱码等错误，将扣分！)

3. 提交文件：

1) 手动识别程序 **man_lex.c**

2) 自动识别程序 **auto_lex.l**

3) 实验报告：**点名册序号-词法分析_姓名.docx**

实验提交方式：

1. 手动识别程序 `man_lex.c` 提交到icoding系统
2. 自动识别程序 `auto_lex.l` 提交到icoding系统
3. 实验报告：最后需要提交综合四次的实验报告

实验用软件

<https://sourceforge.net/projects/tdm-gcc/files/>

<https://sourceforge.net/projects/gnuwin32/files/flex/2.5.4a-1/>

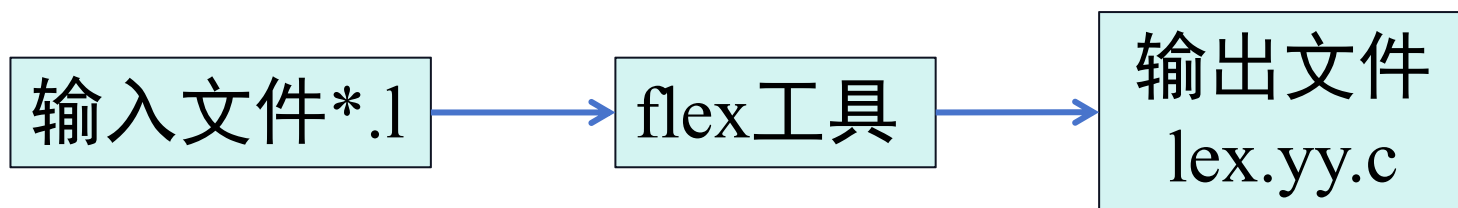
<https://sourceforge.net/projects/gnuwin32/files/bison/2.4.1/>

<https://sourceforge.net/projects/winflexbison/>

在QQ群里已经有flex2.6.4 和bison3.7，大家在群里自行下载

1. FLEX简介

单词的描述称为模式(Lexical Pattern)，模式一般用正规表达式进行精确描述。**FLEX通过读取一个有规定格式的文本文件，输出一个如下所示的C语言源程序。**



FLEX的输入文件称为LEX源文件，**它内含正规表达式和对相应模式处理的C语言代码**。LEX源文件的扩展名习惯上用.l表示。FLEX通过对源文件的**扫描自动生成相应的词法分析函数 `int yylex()`**，并将之输出到名规定为lex.yy.c的文件中。实用时，可将其改名为lexyy.c。**该文件即为LEX的输出文件或输出的词法分析器**。也可将 `int yylex()` 加入自己的工程文件中使用。

2. FLEX运行环境

在QQ群里下载FlexBison.zip文件，这里有flex2.6.4和bison3.7



解压FlexBison.zip文件

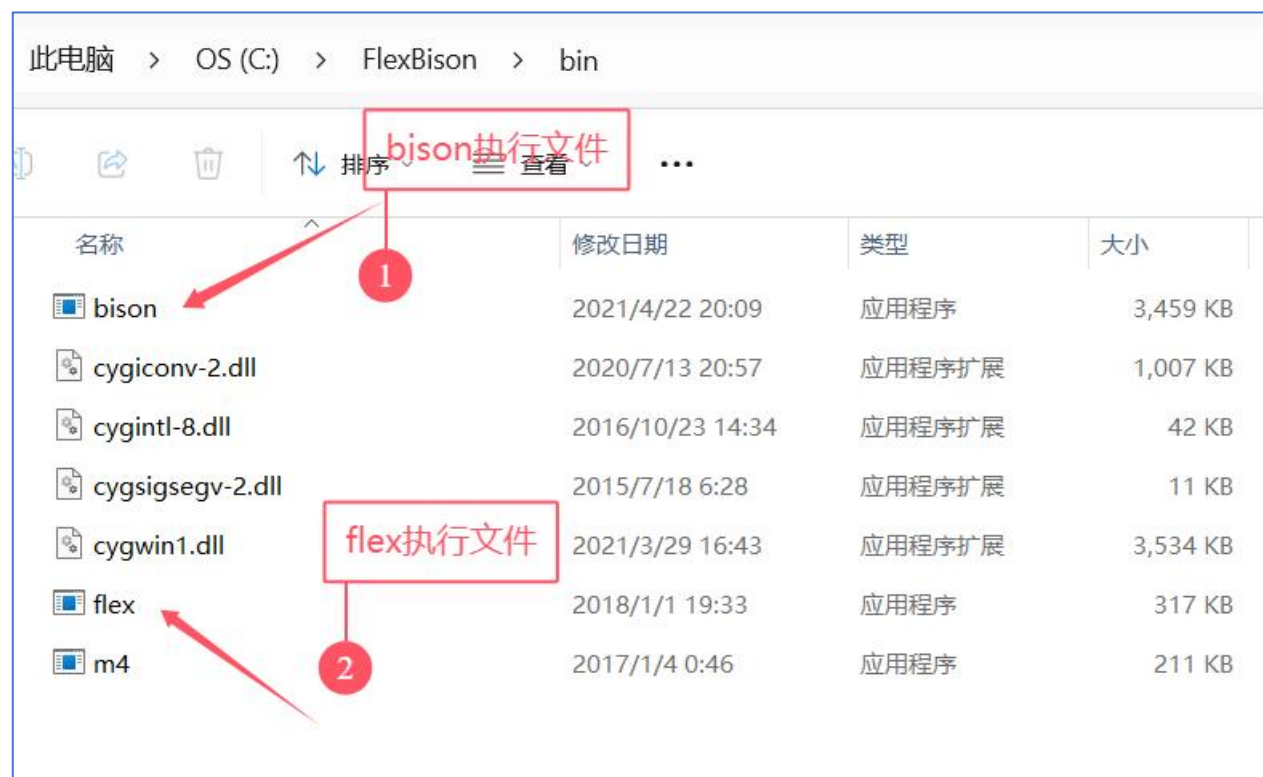
将FlexBison.zip文件解压到C:盘根目录，生成文件目录为FlexBison

名称	修改日期	类型	大小
Apps	2022/7/11 1:05	文件夹	
Dell	2022/7/28 23:23	文件夹	
Download	2023/5/6 6:36	文件夹	
Drivers	2022/7/11 16:49	文件夹	
edb	2022/7/29 7:34	文件夹	
e-logo	2022/7/11 16:47	文件夹	
FlexBison	2024/7/5 15:14	文件夹	
FIFile	2023/5/27 8:15	文件夹	
GnuWin32	2024/10/5 12:34	文件夹	
KDubaSoftDownloads	2024/6/19 9:25	文件夹	
Microsoft Shared	2024/9/1 8:35	文件夹	
OfFile	2023/5/27 8:15	文件夹	
OneDriveTemp	2022/7/28 11:54	文件夹	
PerfLogs	2022/5/7 13:24	文件夹	
Program Files	2024/10/6 14:09	文件夹	
Program Files (x86)	2024/10/2 7:52	文件夹	
ProgramData	2024/10/1 10:47	文件夹	

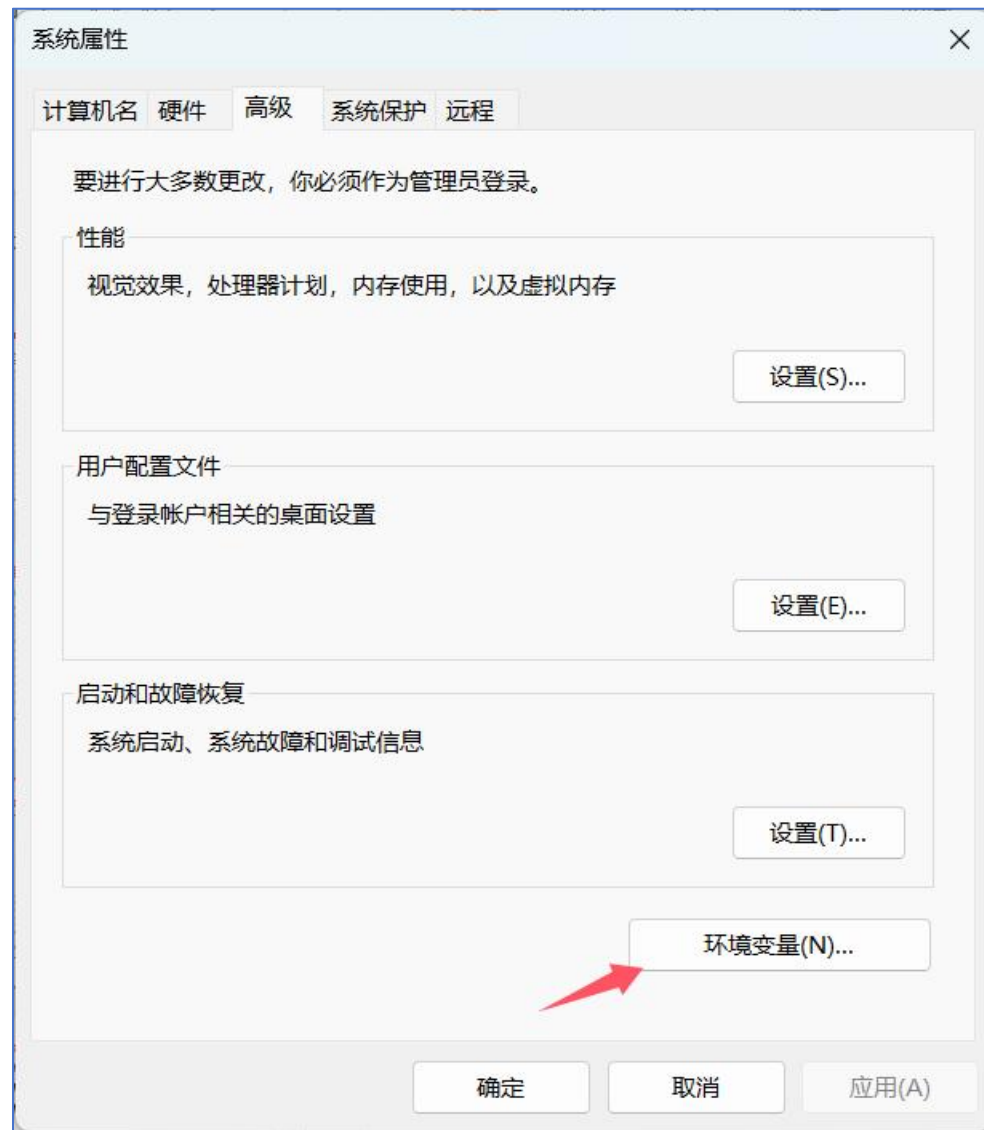
此电脑 > OS (C:) > FlexBison >		
名称	修改日期	类型
bin	2024/7/5 15:14	文件夹
data	2024/7/5 15:14	文件夹
readme	2021/4/22 20:27	文本文档

Flex命令文件目录

执行文件所在目录C:\FlexBison\bin

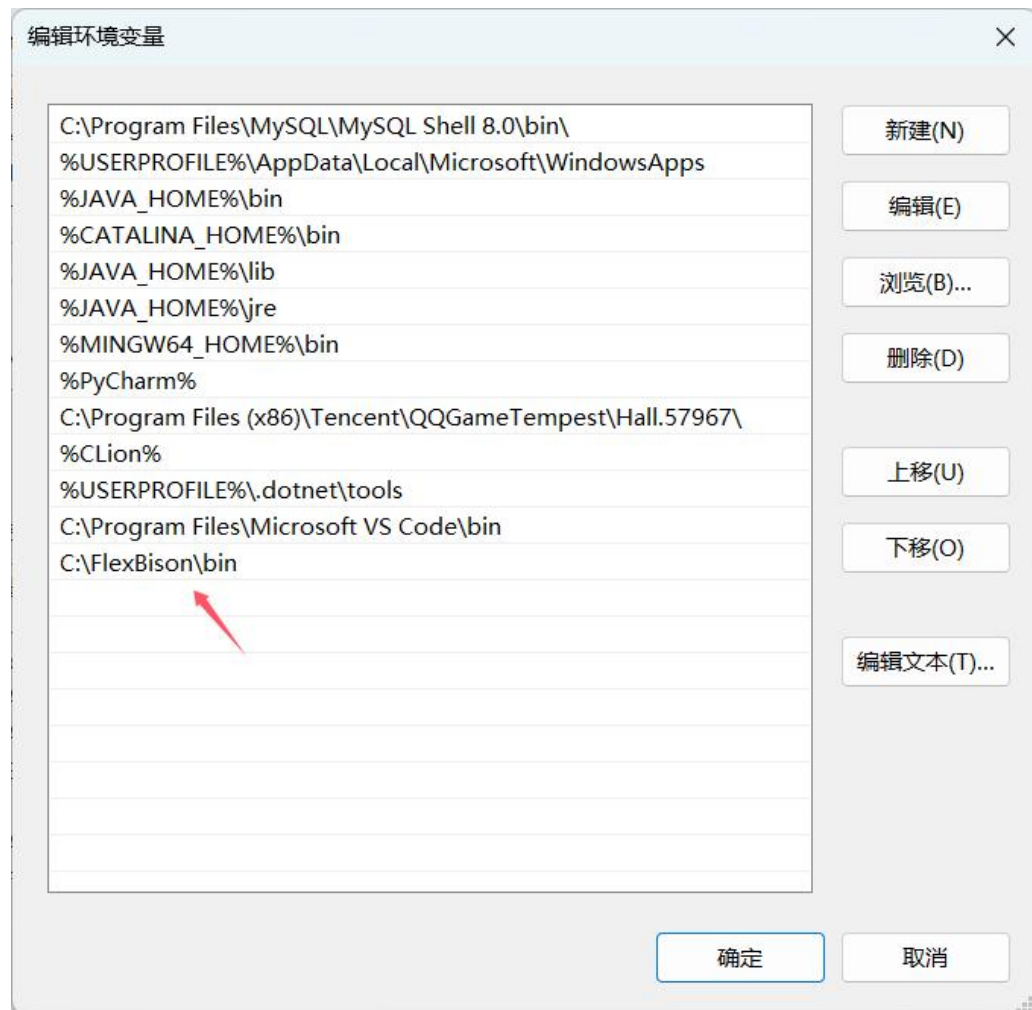
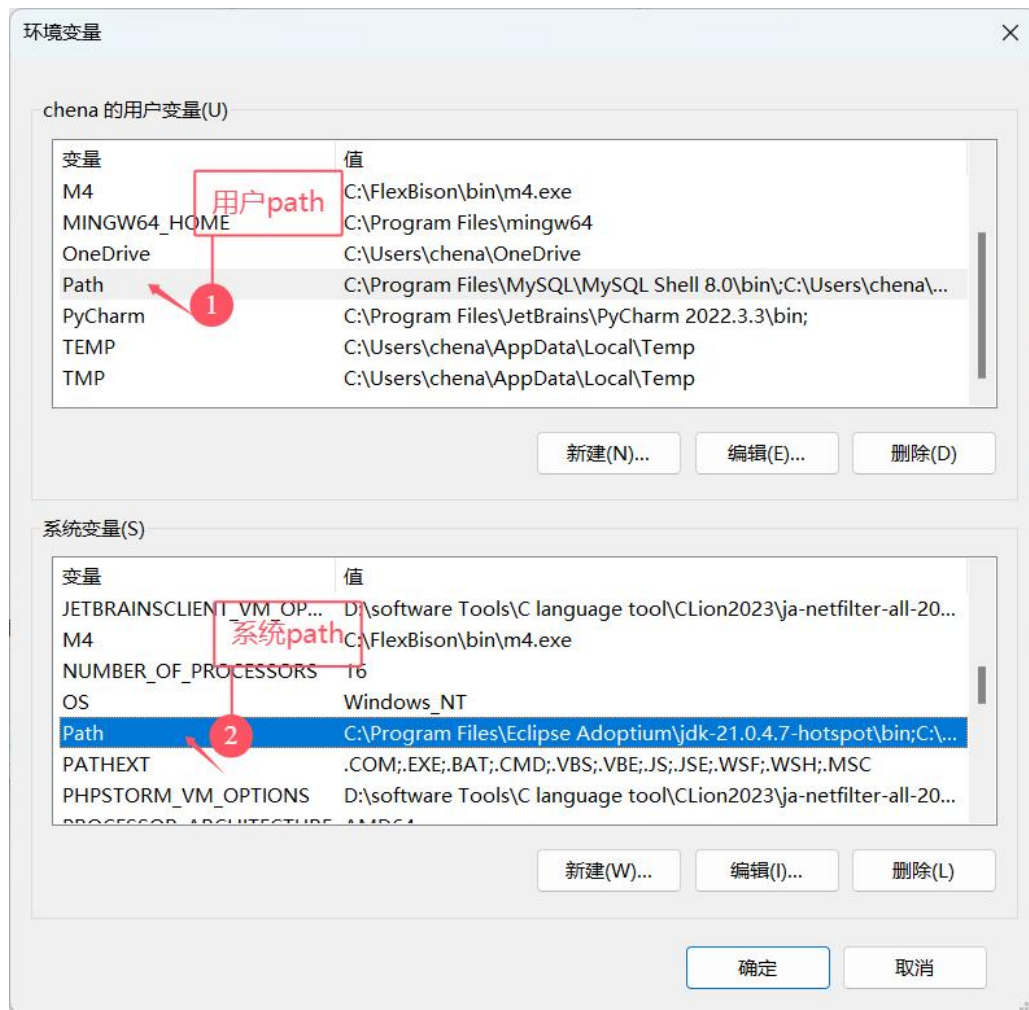


配置 Flex & bison 环境变量



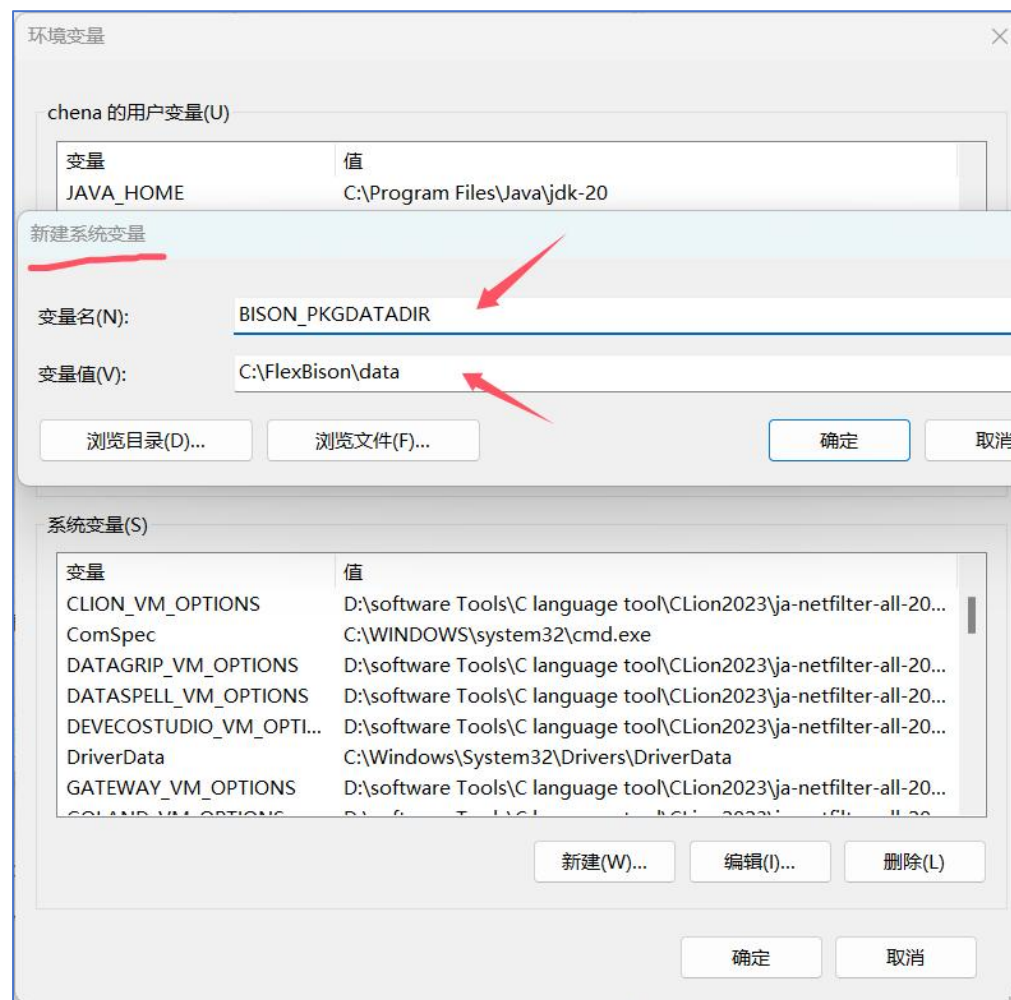
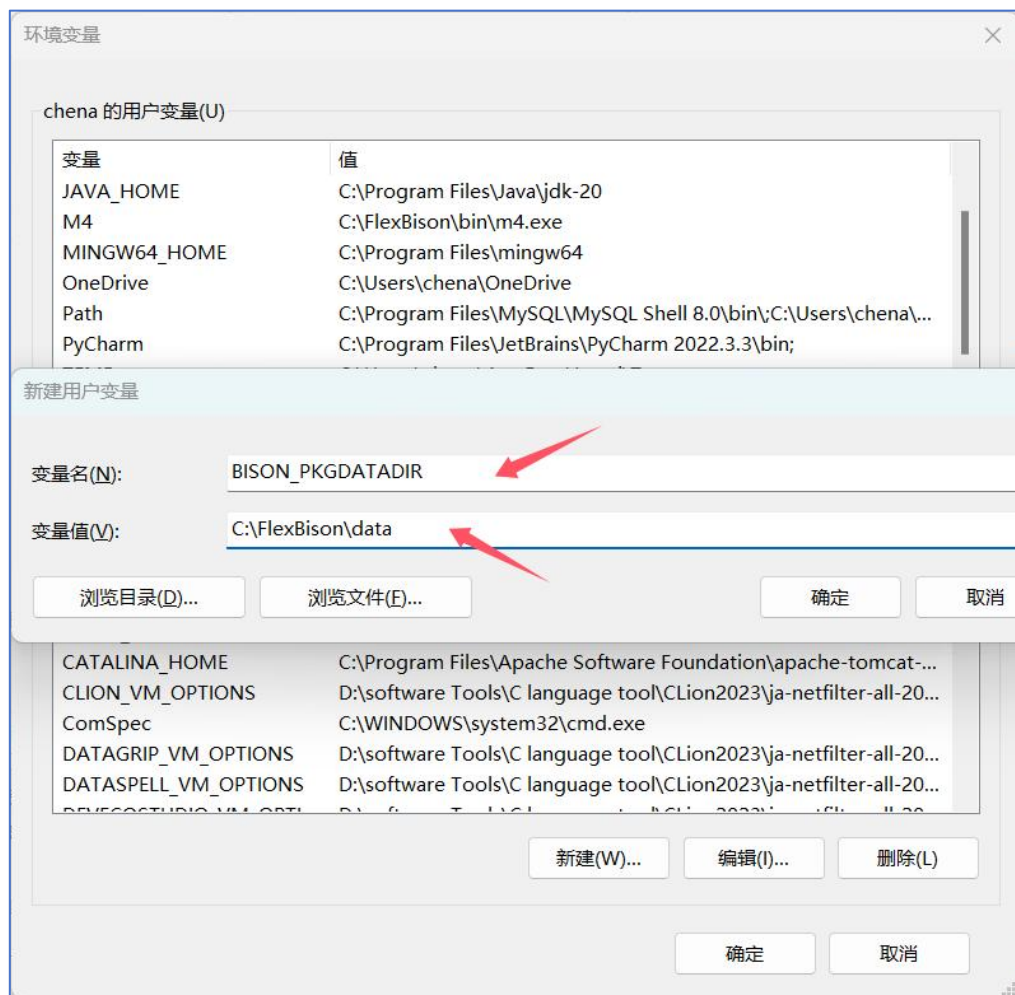
配置Flex&bison环境变量

1.修改用户/系统的环境变量，将C:\FlexBison\bin加到PATH中



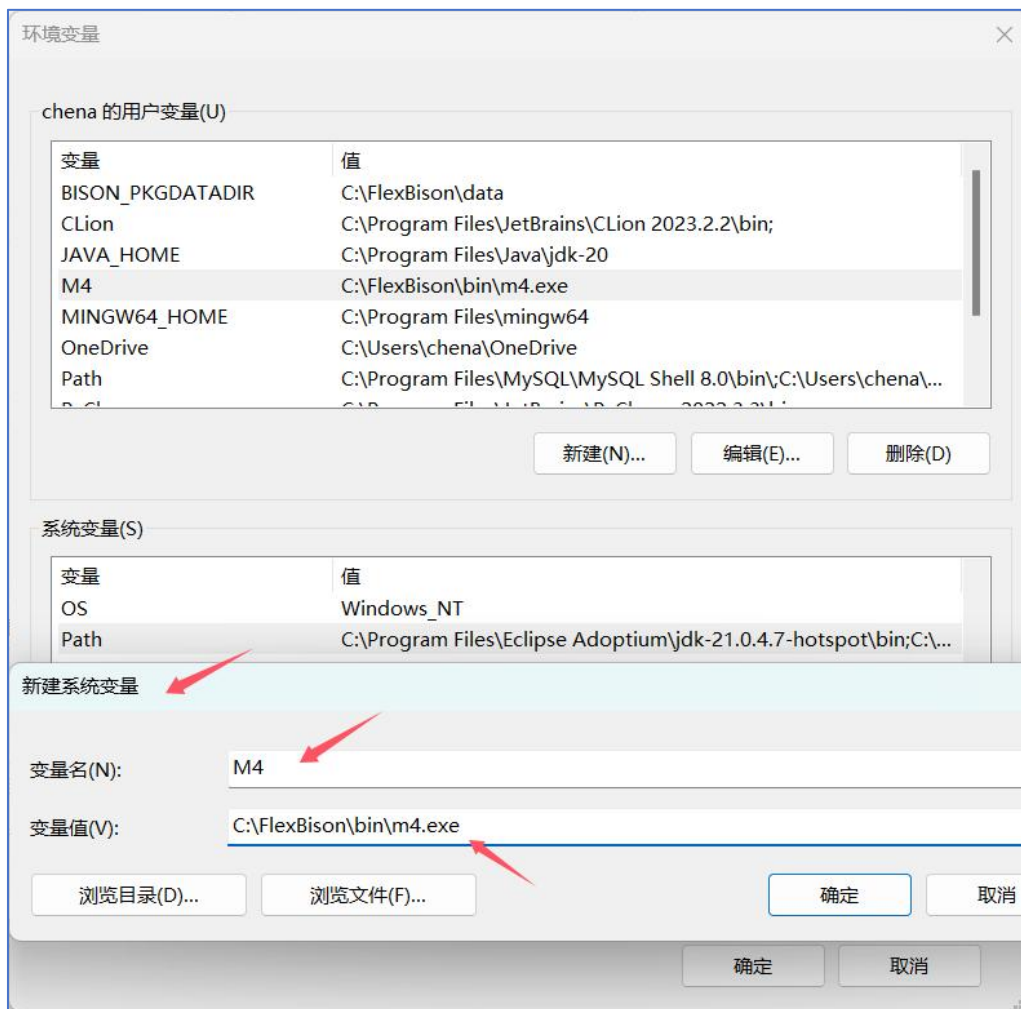
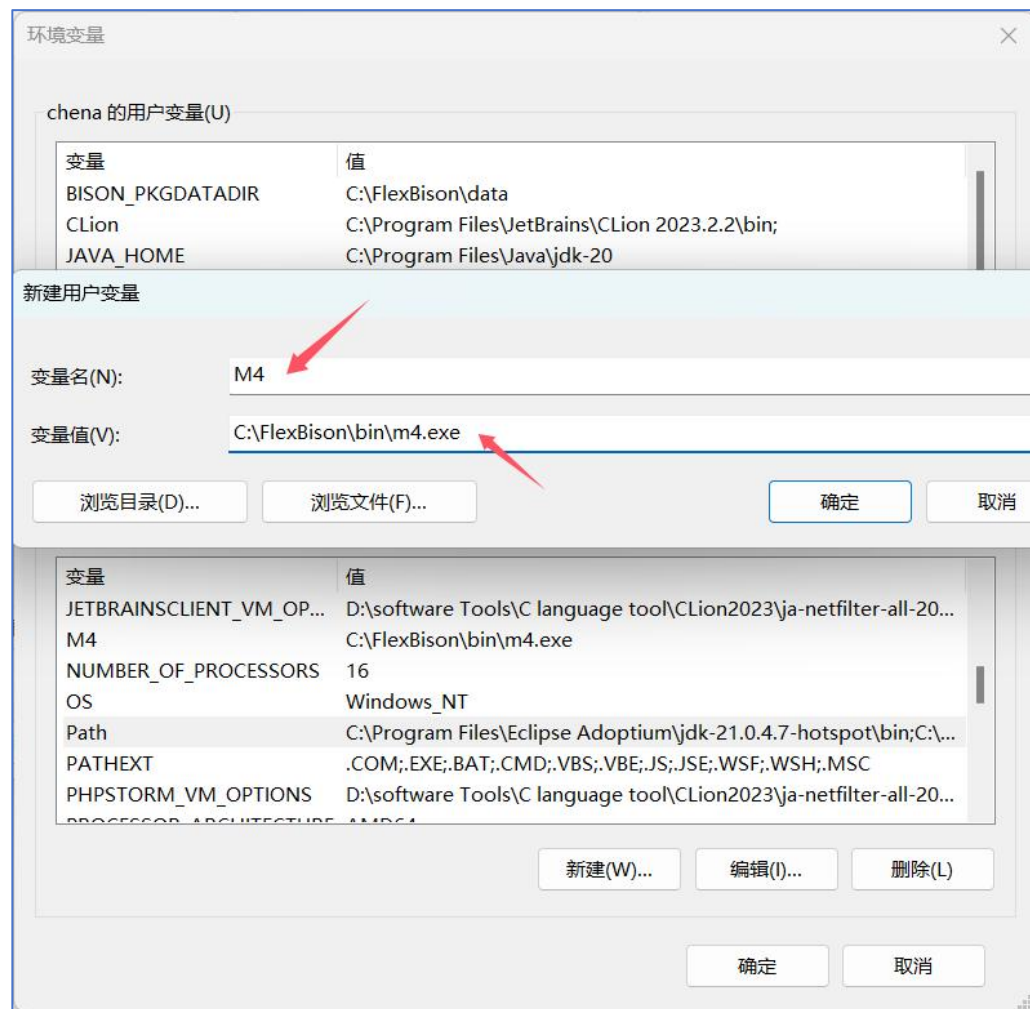
配置Flex&bison环境变量

2.增加用户/系统的环境变量BISON_PKGDATADIR, 将值设置为C:\FlexBison\data



配置Flex&bison环境变量

3.增加用户/系统的环境变量M4，将值设置为C:\FlexBison\bin\m4.exe



测试环境

C:>flex -V

```
命令提示符
Microsoft Windows [版本 10.0.22631.4249]
(c) Microsoft Corporation。保留所有权利。

C:\Users\chena>flex -V
flex 2.6.4

C:\Users\chena>
```

C:>bison -V

```
命令提示符
Microsoft Windows [版本 10.0.22631.4249]
(c) Microsoft Corporation。保留所有权利。

C:\Users\chena>flex -V
flex 2.6.4

C:\Users\chena>bison -V
bison (GNU Bison) 3.7
Written by Robert Corbett and Richard Stallman.

Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying
conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS
FOR A PARTICULAR PURPOSE.

C:\Users\chena>
```

定义部份

定义部份由C语言代码、模式的宏定义、条件模式的开始条件说明三部份组成。其中，C代码部份由顶行的%{和}%引入，LEX扫描源文件时将%{和}%之间的部分原封不动的拷贝到输出文件*.c中。

```
1
2 %{
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 %}
7
```

C语言的头文件这里引用

1

定义部份

模式的宏定义部份如同C语言中的宏定义，通过宏名定义一个模式，这样，可以简化在源文件中多次出现的正规表达式的书写。格式为：

宏名1 宏定义1

宏名2 宏定义2

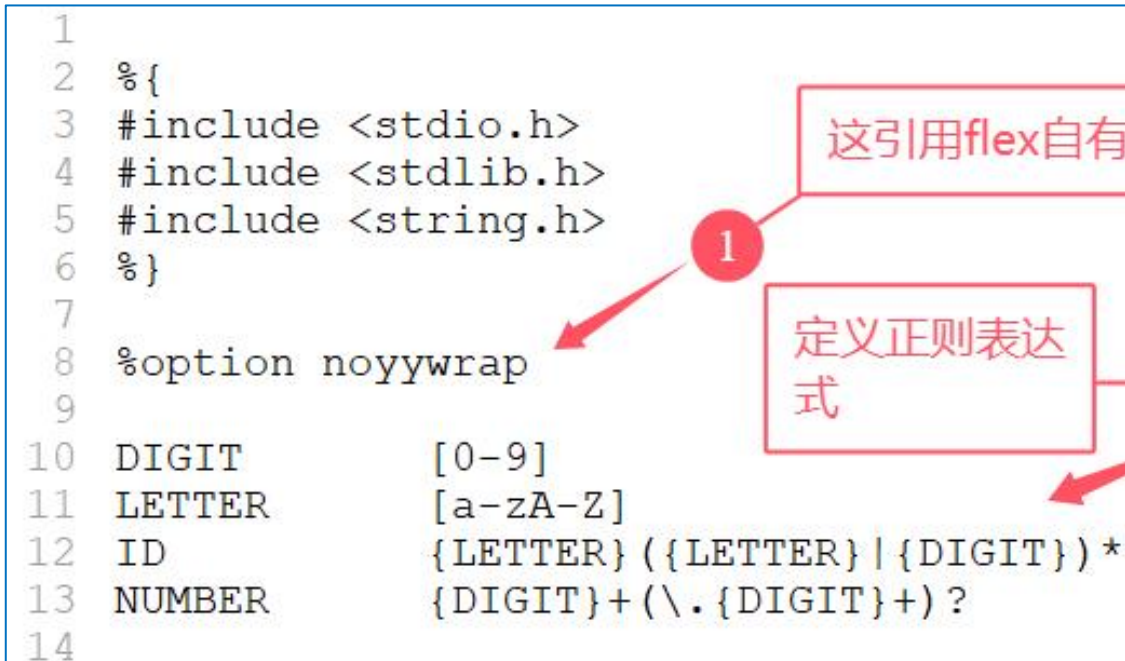
.....

例如：

DIGIT [0-9]

ID [A-Za-z][A-Za-z0-9_]*

```
1
2 %{
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 %}
7
8 %option noyywrap
9
10 DIGIT      [0-9]
11 LETTER     [a-zA-Z]
12 ID         {LETTER} ({LETTER} | {DIGIT}) *
13 NUMBER     {DIGIT}+ (\.{DIGIT}+)?
14
```



宏名是以字母和下划线“_”开始，以字母、数字和下划线组成的字符串，且大小写敏感。
宏名必须顶行写，宏名和宏定义必须写在同一行上。宏名和宏定义之间以空格符或TAB符隔开。

FLEX定义正则表达式

模 式	解 释
X	匹配单个字符x。(也可将模式写为" x")
.	匹配除换行符' \n' 之外的任意字符
[xyz]	匹配x、 y或z
[abj-oZ]	匹配字符集:a、 b、 Z以及j到o之间的字母(包括j和o)
[^A-Z]	匹配字符集A到Z之间字符集的补集。即除大写字母的其他字符
[^A-Z\n]	匹配除大写字母和换行符之外的其它字符
r*	R是正规表达式, r*匹配0个或多个r
r+	R是正规表达式, r+匹配1个或多个r
r?	R是正规表达式, r?匹配0个或1个r
r{2, 5}	R是正规表达式, r{2, 5}匹配2个到5个r
r{2, }	R是正规表达式, r{2, }匹配2个或以上r
r{4}	R是正规表达式, r{4}匹配4个r

{name}	name是在定义部份出现的模式宏名, 在规则部份将之替换为模式
"[xyz]\ " foo"	匹配字符串[xyz]" foo
\x	如x是' a' 、' b' 、' f' 、' n' 、' r' 或' t' , \x为转义字符, 定义同ANSI C, 否则, 匹配字符x. 此方法用于匹配正规表达式的运算符
\123	匹配八进制ASCII码为123的字符
\x2a	匹配十六进制ASCII码为2a的字符
(r)	匹配r, 优先运算正规式r
Rs	匹配正规式r和s的连接
r s	匹配正规式r或s
r/s	匹配正规式r, 但是, r之后一定要出现正规式s。称s为r的尾部条件
^r	匹配正规式r, 但是, r一定要出现在行首
r\$	匹配正规式r, 但是, r一定要出现在行尾
<s>r	匹配正规表达式r, 但是一定要在开始条件s激活之后
<<EOF>>	匹配文件结束标志

定义程序的token标记符示例

```
15  %{  
16      enum yytokentype {  
17          INT=256,  
18          IF=257,  
19          ELSE=258,  
20          WHILE=259,  
21          RETURN=260,  
22          BREAK=261,  
23          CONTINUE=262,  
24          ASSIGN=263,  
25          EQ=264,  
26          NEQ=265,  
27          LT=266,  
28          LE=267,  
29          GT=268,  
30          GE=269,  
31          ADD=270,  
32          SUB=271,  
33          MUL=272,  
34          DIV=273,  
35          MOD=274,  
36          LPAREN=275,  
37          RPAREN=276,  
38          LBRACE=277,  
39          RBRACE=278,  
40          SEMICOLON=279,  
41          NUMBER=280,  
42          ID=281,  
43      };  
44      int yylval;  
45      FILE *yyin; // 文件输入流  
46  %}
```

1
定义词法分析标记符

2
定义局部变量

规则定义部份

规则部份是FLEX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

C语言代码

模式1 动作1

模式2 |

模式3 动作3

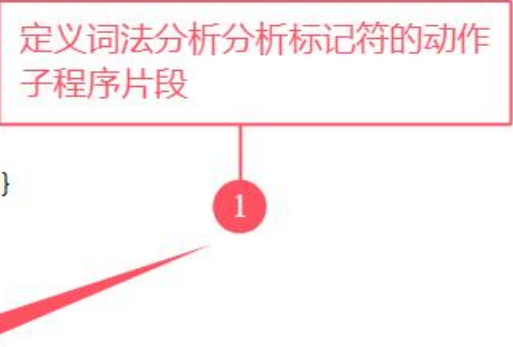
.....

同定义部分一样，C语言代码必须出现在第一个模式之前，包括在%{和}%之中，且%{必须顶行书写。%{和}%之间的代码部份可用来定义yylex()用到的局部变量。

模式必须顶行书写。模式可为正规式或用{}括起且在定义部份定义过的宏名。动作为用{}括起的C代码。且开始括号{与模式之间用白字符隔开，且须和模式在同一行上。注意，在模式后加一|表示模式2和3采用同一动作3.|和模式2以白字符隔开。

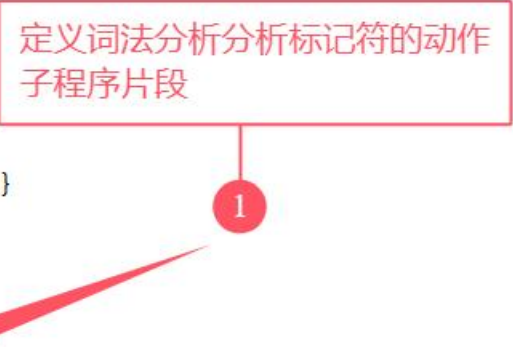
在%% ...%%之间定义标记符分析的子程序片段

```
47
48 %%
49 "int"      { return INT; }
50 "if"       { return IF; }
51 "else"     { return ELSE; }
52 "while"    { return WHILE; }
53 "return"   { return RETURN; }
54 "break"    { return BREAK; }
55 "continue" { return CONTINUE; }
56 "="        { return ASSIGN; }
57 "=="       { return EQ; }
58 "!="       { return NEQ; }
59 "<"        { return LT; }
60 "<="       { return LE; }
61 ">"        { return GT; }
62 ">="       { return GE; }
63 "+"        { return ADD; }
64 "-"        { return SUB; }
65 "*"        { return MUL; }
66 "/"        { return DIV; }
67 "%"        { return MOD; }
68 "("        { return LPAREN; }
69 ")"        { return RPAREN; }
70 "{"        { return LBRACE; }
71 "}"        { return RBRACE; }
72 ";"        { return SEMICOLON; }
73 {ID}       { return ID; }
74 {NUMBER}   { yylval = atof(yytext); return NUMBER; }
75 [ \t\n]    ; // 忽略空格、制表符和换行符
76 .         { printf("ERROR: Invalid character %s\n", yytext); }
77 %%
78
```



在%% ...%%之间定义标记符分析的子程序片段

```
47
48 %%
49 "int"      { return INT; }
50 "if"       { return IF; }
51 "else"     { return ELSE; }
52 "while"    { return WHILE; }
53 "return"   { return RETURN; }
54 "break"    { return BREAK; }
55 "continue" { return CONTINUE; }
56 "="        { return ASSIGN; }
57 "=="       { return EQ; }
58 "!="       { return NEQ; }
59 "<"        { return LT; }
60 "<="       { return LE; }
61 ">"        { return GT; }
62 ">="       { return GE; }
63 "+"        { return ADD; }
64 "-"        { return SUB; }
65 "*"        { return MUL; }
66 "/"        { return DIV; }
67 "%"        { return MOD; }
68 "("        { return LPAREN; }
69 ")"        { return RPAREN; }
70 "{"        { return LBRACE; }
71 "}"        { return RBRACE; }
72 ";"        { return SEMICOLON; }
73 {ID}       { return ID; }
74 {NUMBER}   { yylval = atof(yytext); return NUMBER; }
75 [ \t\n]    ; // 忽略空格、制表符和换行符
76 .         { printf("ERROR: Invalid character %s\n", yytext); }
77 %%
78
```



用户附加C语言部份

FLEX对此部份不作任何处理，仅仅将之直接拷贝到输出文件*.c的尾部。在这些部份，可定义对模式进行处理的C语言函数、主函数和yylex要调用的函数yywrap()等。如果用户在其他C模块中提供这些函数，用户代码部份可以省略。

编写C语言的main程序

```
78
79 int main(int argc, char *argv[]) {
80     int token;
81     if (argc != 2) {
82         fprintf(stderr, "Usage: %s <input_file>\n", argv[0]);
83         return 1;
84     }
85
86     yyin = fopen(argv[1], "r");
87     if (!yyin) {
88         fprintf(stderr, "Error opening input file\n");
89         return 1;
90     }
91
92     while((token = yylex()) != 0) {
93         printf("%s      %d", yytext, token);
94         if(token == NUMBER) printf(" =%d\n", yylval);
95         else printf("\n");
96     }
97     fclose(yyin);
98     return 0;
99 }
100
```

1 检查程序运行输入参数，是否有需词法分析的文件名及路径

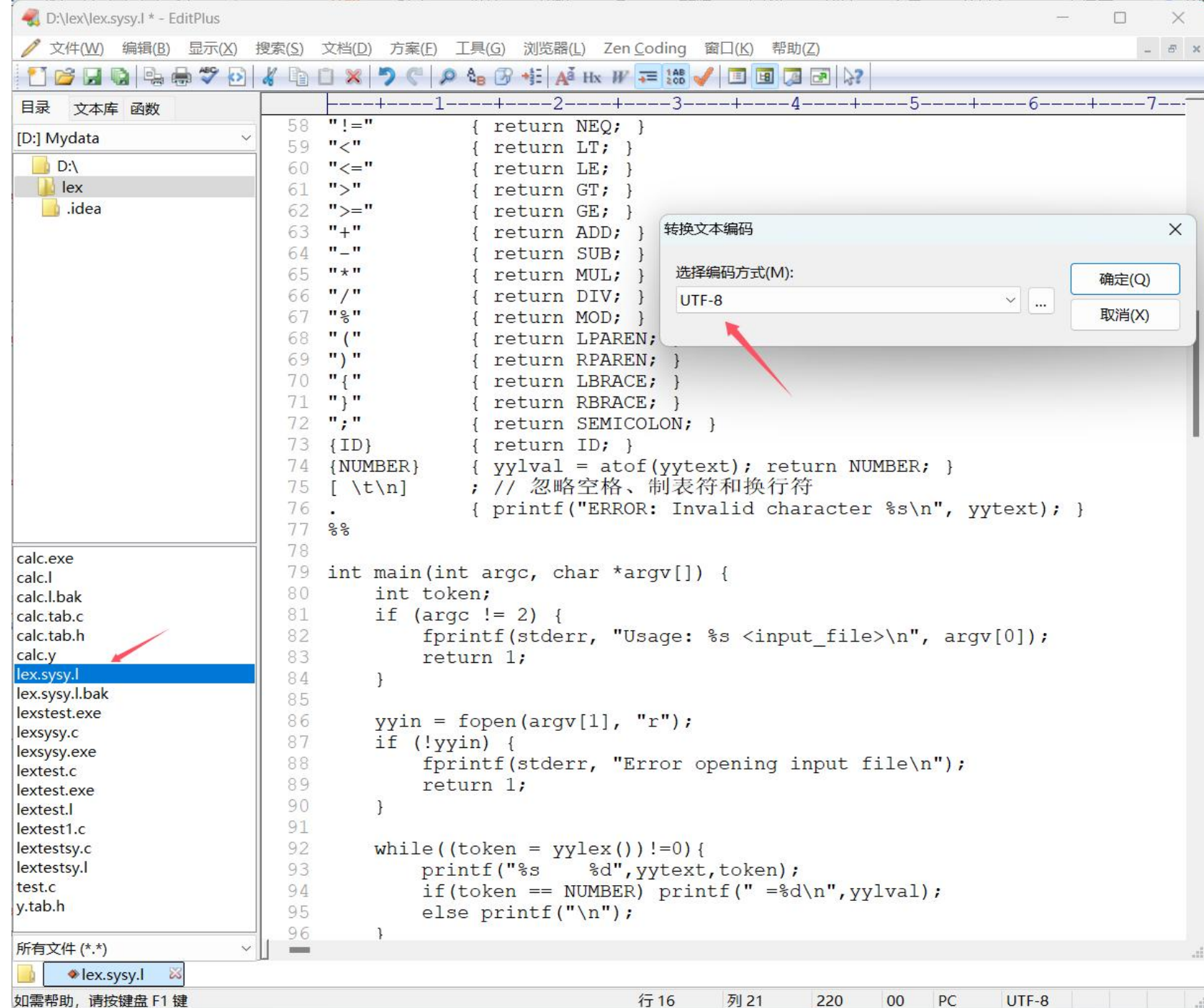
2 打开需词法分析的文件

3 显示词法分析的结果

模式匹配说明

`yylex()`函数被调用之后，它首先检查全局文件指针变量`yyin`是否有定义，如有，则将之设置为将要扫描的文件指针。如无，则设置为标准输入文件`stdin`。同理，如全局文件指针变量`yyout`无定义，则将之设置为标准输出文件`stdout`。

保存词法分析程序
为lex.sysy.l,字符
编码必须选UTF-8



使用 `flex -o lexsy.c lex.sys.l` 转化纯C语言程序

```
C:\Users\chena>D:
D:\>
D:\>cd lex
D:\lex>dir
驱动器 D 中的卷是 Mydata
卷的序列号是 F27A-34EB

D:\lex 的目录

2024/10/07 06:49 <DIR> .
2024/10/05 15:08 <DIR> .idea
2024/10/06 09:19      109,277 calc.exe
2024/10/06 21:09       1,201 calc.l
2024/10/06 09:07       1,652 calc.l.bak
2024/10/06 09:19     44,340 calc.tab.c
2024/10/06 06:54       2,945 calc.tab.h
2024/10/06 09:17       2,157 calc.y
2024/10/07 06:50       5,566 lex.sysy.l
2024/10/07 06:46       5,589 lex.sysy.l.bak
2024/10/06 12:42    105,844 lexstest.exe
2024/10/07 06:49     53,366 lexsy.c
2024/10/07 06:49    106,366 lexsy.exe
2024/10/06 13:41     48,629 lextest.c
2024/10/06 13:42    105,826 lextest.exe
2024/10/06 13:39       1,397 lextest.l
2024/10/06 13:21     48,595 lextest1.c
2024/10/06 13:18     48,663 lextestsy.c
2024/10/06 13:11       1,397 lextestsy.l
2024/10/04 21:56        109 test.c
2024/10/05 18:53       2,519 y.tab.h
                19 个文件      695,438 字节
                2 个目录 318,599,954,432 可用字节

D:\lex>
```

```
命令提示符
D:\lex 的目录

2024/10/07 06:49 <DIR> .
2024/10/05 15:08 <DIR> .idea
2024/10/06 09:19      109,277 calc.exe
2024/10/06 21:09       1,201 calc.l
2024/10/06 09:07       1,652 calc.l.bak
2024/10/06 09:19     44,340 calc.tab.c
2024/10/06 06:54       2,945 calc.tab.h
2024/10/06 09:17       2,157 calc.y
2024/10/07 06:50       5,566 lex.sysy.l
2024/10/07 06:46       5,589 lex.sysy.l.bak
2024/10/06 12:42    105,844 lexstest.exe
2024/10/07 06:49     53,366 lexsy.c
2024/10/07 06:49    106,366 lexsy.exe
2024/10/06 13:41     48,629 lextest.c
2024/10/06 13:42    105,826 lextest.exe
2024/10/06 13:39       1,397 lextest.l
2024/10/06 13:21     48,595 lextest1.c
2024/10/06 13:18     48,663 lextestsy.c
2024/10/06 13:11       1,397 lextestsy.l
2024/10/04 21:56        109 test.c
2024/10/05 18:53       2,519 y.tab.h
                19 个文件      695,438 字节
                2 个目录 318,599,954,432 可用字节

D:\lex>flex -lexsy.c lex.sys.l
flex: Unrecognized option 'e'
Try 'flex --help' for more information.

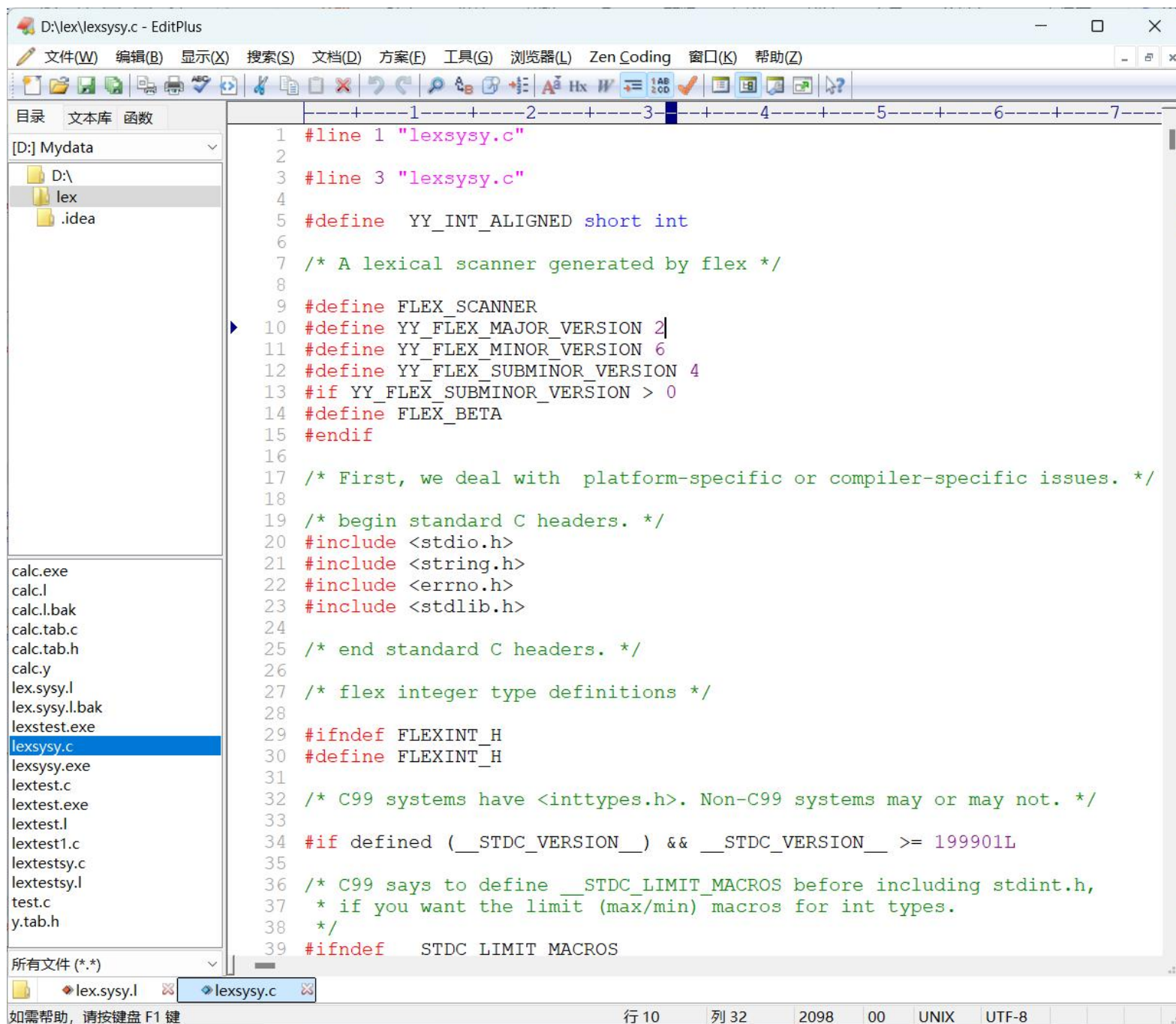
D:\lex>flex -o lexsy.c lex.sys.l

D:\lex>
```

注意: -o参数后面指出输出文件名

打开lexsysy.c

C语言程序



```
D:\lex\lexsysy.c - EditPlus
文件(W) 编辑(B) 显示(X) 搜索(S) 文档(D) 方案(E) 工具(G) 浏览器(L) Zen_Coding 窗口(K) 帮助(Z)

目录 文本库 函数
[D:] Mydata
D:\
lex
.idea

calc.exe
calc.l
calc.l.bak
calc.tab.c
calc.tab.h
calc.y
lex.sysy.l
lex.sysy.l.bak
lextest.exe
lexsysy.c
lexsysy.exe
lextest.c
lextest.exe
lextest.l
lextest1.c
lextestsy.c
lextestsy.l
test.c
y.tab.h

所有文件 (*.*)
lex.sysy.l lexsysy.c

1 #line 1 "lexsysy.c"
2
3 #line 3 "lexsysy.c"
4
5 #define YY_INT_ALIGNED short int
6
7 /* A lexical scanner generated by flex */
8
9 #define FLEX_SCANNER
10 #define YY_FLEX_MAJOR_VERSION 2
11 #define YY_FLEX_MINOR_VERSION 6
12 #define YY_FLEX_SUBMINOR_VERSION 4
13 #if YY_FLEX_SUBMINOR_VERSION > 0
14 #define FLEX_BETA
15 #endif
16
17 /* First, we deal with platform-specific or compiler-specific issues. */
18
19 /* begin standard C headers. */
20 #include <stdio.h>
21 #include <string.h>
22 #include <errno.h>
23 #include <stdlib.h>
24
25 /* end standard C headers. */
26
27 /* flex integer type definitions */
28
29 #ifndef FLEXINT_H
30 #define FLEXINT_H
31
32 /* C99 systems have <inttypes.h>. Non-C99 systems may or may not. */
33
34 #if defined (__STDC_VERSION__) && __STDC_VERSION__ >= 199901L
35
36 /* C99 says to define __STDC_LIMIT_MACROS before including stdint.h,
37  * if you want the limit (max/min) macros for int types.
38  */
39 #ifndef STDC_LIMIT_MACROS
```

编译lexsysy.c C语言程序

gcc lexsysy.c -o lexsysy -lm

```
命令提示符
2024/10/06 12:42 105,844 lexstest.exe
2024/10/07 06:49 53,366 lexsysy.c
2024/10/07 06:49 106,366 lexsysy.exe
2024/10/06 13:41 48,629 lextest.c
2024/10/06 13:42 105,826 lextest.exe
2024/10/06 13:39 1,397 lextest.l
2024/10/06 13:21 48,595 lextest1.c
2024/10/06 13:18 48,663 lextestsy.c
2024/10/06 13:11 1,397 lextestsy.l
2024/10/04 21:56 109 test.c
2024/10/05 18:53 2,519 y.tab.h
19 个文件 695,438 字节
2 个目录 318,599,954,432 可用字节

D:\lex>flex -lexsysy.c lex.sysy.l
flex: Unrecognized option 'e'
Try 'flex --help' for more information.

D:\lex>flex -o lexsysy.c lex.sysy.l

D:\lex>gcc lexsysy.c -o lexsysy -lm

D:\lex>
```

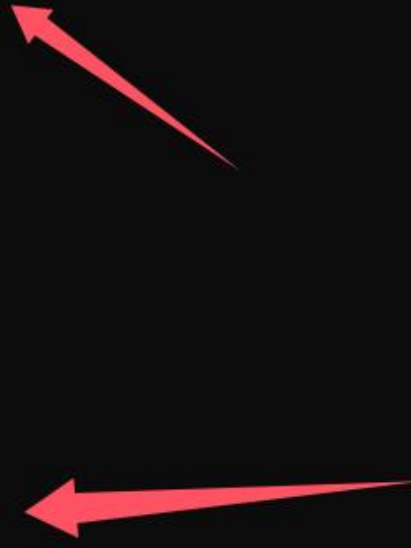
运行 `lexsysy test.c` 分析 `test.c` 程序

```
1 int main() {  
2     int a = 10;  
3     if (a > 5) {  
4         return a;  
5     } else {  
6         return 0;  
7     }  
8 }
```

命令提示符

```
D:\lex>lexsysy test.c  
int      256  
main     281  
(        275  
)        276  
{        277  
int      256  
a        281  
=        263  
10       280 =10  
;        279  
if       257  
(        275  
a        281  
>       268  
5        280 =5  
)        276  
{        277  
return   260  
a        281  
:        279  
;        278  
}        278  
else     258  
{        277  
return   260  
0        280 =0  
:        279  
;        278  
}        278  
}
```

D:\lex>



Flex高级部分：（系统变量）

yyin	FILE* 类型。它指向 lexer 正在解析的当前文件。
yyout	FILE* 类型。它指向记录 lexer 输出的位置。缺省情况下，yyin 和 yyout 都指向标准输入和输出。
yytext	匹配模式的文本存储在这一变量中（char*）。
yylen	给出匹配模式的长度。
yylineno	提供当前的行数信息。（lexer不一定支持。）

Flex高级部分：（系统函数）

<code>yylex()</code>	这一函数开始分析。它由 Lex 自动生成。
<code>yywrap()</code>	这一函数在文件（或输入）的末尾调用。如果函数的返回值是1，就停止解析。因此它可以用来解析多个文件。代码可以写在第三段，这就能够解析多个文件。方法是使用 <code>yyin</code> 文件指针（见上表）指向不同的文件，直到所有的文件都被解析。最后， <code>yywrap()</code> 可以返回 1 来表示解析的结束。
<code>yyless(int n)</code>	这一函数可以用来送回除了前n个字符外的所有读出标记。
<code>yyomore()</code>	这一函数告诉 Lexer 将下一个标记附加到当前标记后。