词法分析器的设计与实现

姓名：黄昱衡 班级：2017211301 学号：2017210445

一.实验目的

理解词法分析器在编译器中的作用。

通过手动实现词法分析器，来加深对词法分析、正规表达式、正规文法、有限状态自动机的理解。

二.实验要求

要求设计并实现 C 语言的词法分析程序，并要求实现如下功能：

1. 可以识别出用 C 语言编写的源程序中的每个单词符号，并以记号的形式输出每个单词符号。
2. 可以识别并跳过源程序中的注释。
3. 可以统计源程序中的语句行数、各类单词的个数，以及字符总数，并输出统计结果。
4. 监察源程序中存在的词法错误，并报告错误所在位置。
5. 对源程序中出现的错误进行适当的恢复，使词法分析可以继续进行，对源程序进行一次扫描，即可监察并报告源程序中存在的所有词法错误。

三.实验环境

Operating system：macOS Mojave 10.14.5。

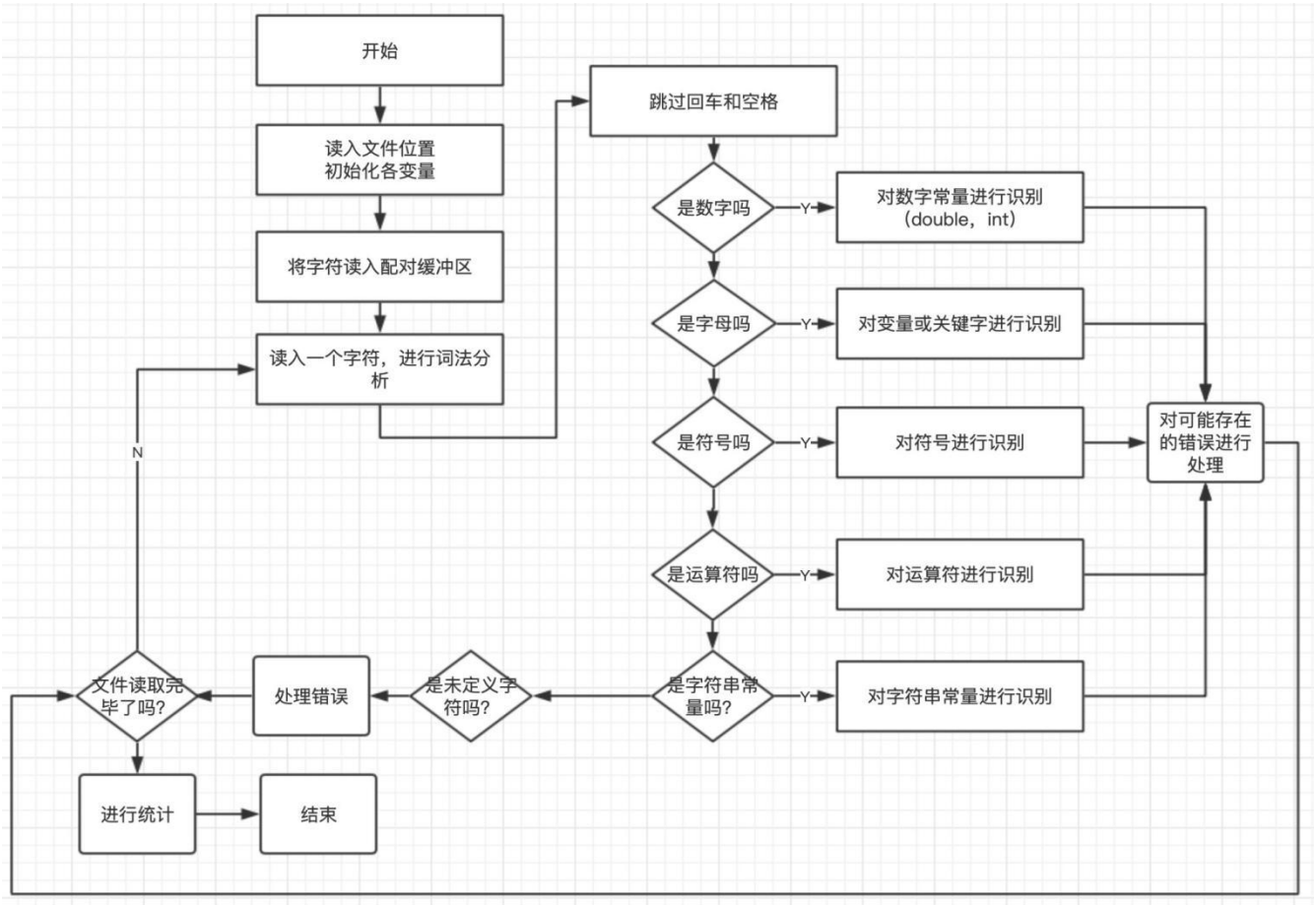
IDE：Clion.

Language standard：CMAKE_CXX_STANDARD 14

四.实验内容

1）主要内容概述

本次完成的词法分析器，其大致流程如下图所示：



其中识别的单词符号列于下表：

Operators

表达式(name)	记号(symbol)	属性(property)
*	op	multiply
/	op	divide
%	op	mod
+	op	add
-	op	subtract
<	op	less
<=	op	lessequal
>	op	greater
>=	op	greaterequal
==	op	equal
!=	op	notequal
!	op	not
=	op	assign
&&	op	and
&	op	bitAnd
||	op	or

|	op	bitOr
<<	op	shiftLeft
>>	op	shiftRight
--	op	selfSubtract
++	op	selfAdd
+=	op	addAssign
-=	op	subAssign
*=	op	mulAssign
/=	op	divAssign
%=	op	modAssign
&=	op	bitAndAssign
^=	op	bitXorAssign
^	op	bitXor
|=	op	bitOeAssign
<<=	op	shiftLAssign
>>=	op	shiftRAssign

Symbols

表达式	记号	属性
(	symbol	LeftParen
)	symbol	RightParen
{	symbol	LeftBrace
}	symbol	RightBrace
;	symbol	Semicolon
,	symbol	Comma

keywords

表达式	记号	属性
if	keyword	if
else	keyword	else
while	keyword	while
print	keyword	print
for	keyword	for
then	keyword	then
do	keyword	do
void	keyword	void
char	keyword	char
double	keyword	double
enum	keyword	enum
float	keyword	float
int	keyword	int

long	keyword	long
short	keyword	short
signed	keyword	signed
struct	keyword	struct
union	keyword	union
unsigned	keyword	unsigned
auto	keyword	auto
return	keyword	return
assert	keyword	NULL
break	keyword	break
case	keyword	case
const	keyword	const
continue	keyword	continue
default	keyword	default
extern	keyword	extern
goto	keyword	goto
register	keyword	register
sizeof	keyword	sizeof
static	keyword	static
switch	keyword	swithc
typdef	keyword	typedf
volatile	keyword	volatile

constant and variable

表达式	记号	属性
ID(自定义变量名称)	ID	入口指针
int	constant	int
double	constant	double
“string”（字符串）	constant	string
‘s’（字符）	constant	literal

另外，为了提高程序运行效率，本次实验尽量少使用 C++的相关内容。比如说全部使用 char 数组来存储字符串而不是 string 类。本次实验也未直接使用 C 与 C++的正则表达式。

2）细节阐释

1.使用缓冲区处理输入的字符串

为了得到一个单词符号的确切性质，需要超前扫描若干个字符。因此需要缓冲区存储输入的字符串流。然而由于缓冲区多大，都不能保证单词不被它的边界打断，因此需要使用配对缓冲区来处理输入的字符串流。配对缓冲区类似于循环队列，当向前指针指向前半区或者后半区的中止哨兵时,将接下来的字符串读入到另一个半区当中。该过程的伪代码表示如下:

```
向前指针前移一个位置;
If(向前指针指向 eof(即 ch==0)){
    If(向前指针在左半区的终点){
        读入字符串，填充右半区;
        向前指针前移一个位置;
    }
    else if(向前指针在右半区的终点){
        读入字符串，填充左半区;
        向前指针移到缓冲区的开始位置;
    }
    else 终止词法分析
}
```

在 C 语言中，文件流的部分读取可使用 stdio.h 中 fread 函数实现。

词法分析器的指针前移，指针回退与读入配对缓冲区过程分别由函数 forward_pointer 、retract_pointer 与 read_to_buff 三个函数实现。

为了配合词法分析器中指针退回的操作，在实际实现的时候每个半区另外增加了个一个哨兵，用以处理指针刚好越过半区时需要回退的情况。具体处理如下：

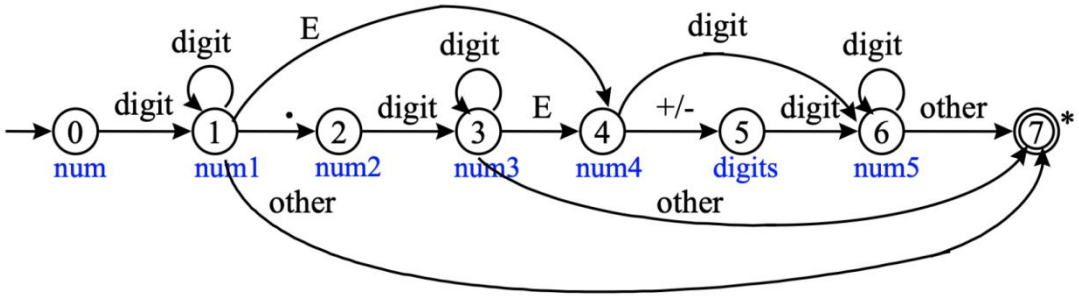
1）如果指针在左半区开头，那么将指针退回到 BUFFLEN + 2 的位置，并把右半区的最后一个字符存入 buff[BUFFLEN+2]. 2）如果指针在右半区开头，则将指针退回到 BUFFLEN/2 + 1 的位置，并把左半区最后一个字符存入 buff[BUFFLEN/2+1]。这样处理可以避免回退时可能会出现的重复读取的情况。

2. 词法分析的实现

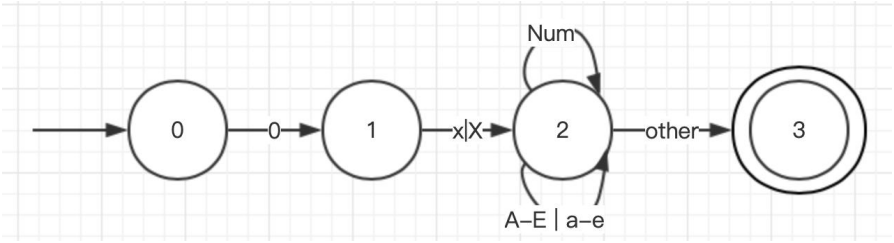
每个单词符号由三元组(name, symbol, property) 组成。

1. 对常量数字的处理

本次实验完成了常量数字处理的**两种**实现。一种借助了函数 `strtod` 与 `strtol` 来实现，通过这两个函数对引用指针(`Reference to an already allocated object of type char*`)的修改来判断输入的常量数字是否合法。该实现通过 `deal_with_num` 函数完成。另一种借助有限状态自动机来完成。该有限状态自动机如下图所示：



以上自动机没有实现对 16 进制数(0x)的识别，追加针对 16 进制数识别的自动机，如下图所示：



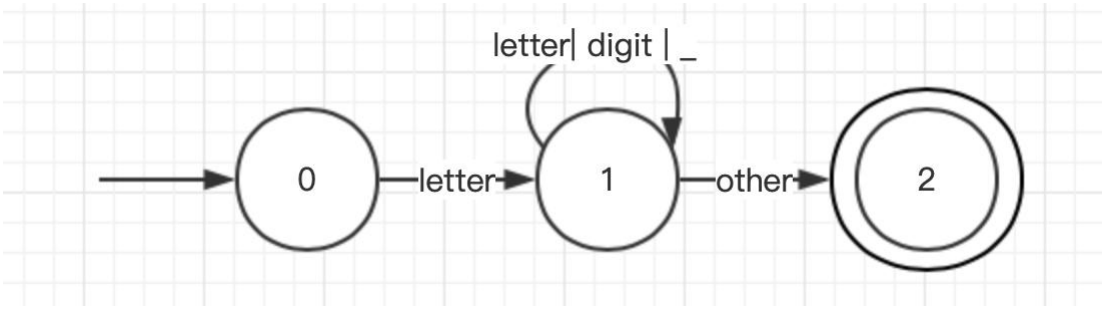
该种实现通过函数 fm_deal_with_num 完成。

常量数字中，double 的三元组为(num, constant, double)，int 的三元组为(num, constant, int)。其中 num 为识别出的数字。

2.对标识符的处理

当指针指向的字符为字母时(a-z or A-Z)，进入函数 deal_with_letter。该函数将处理标识符，并将拼接起的字符串与保留字进行比对。若是保留字，则返回三元组(name, keyword, property)。若是用户自定义标识符，则返回三元组(name, ID, sequence-pointer)。其中 sequence 表示该标识符是第几个标识符，pointer 表示该标识符的记号表入口指针。

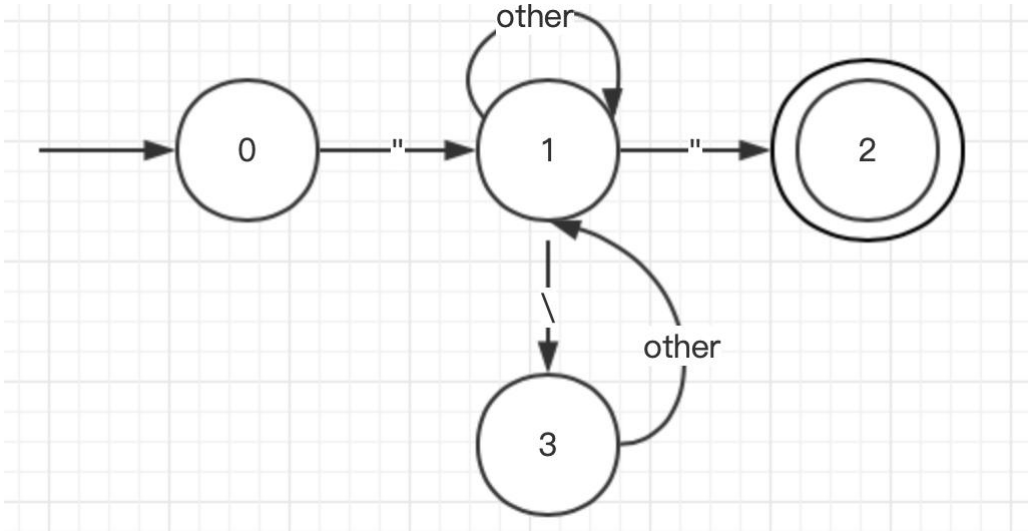
标识符的自动机识别如下图所示：



3.对字符与字符串的处理

用‘’包围起来的字符将被识别为字符常量，用“”包围起来的字符串将被识别为字符串常量类型。字符串常量中的转义字符\n 在行统计中不会被算作一行。但是注释中的\n 将被算作一行。这样的设定在词法分析器报错显示错误所在行数时会比较符合平常使用的习惯。

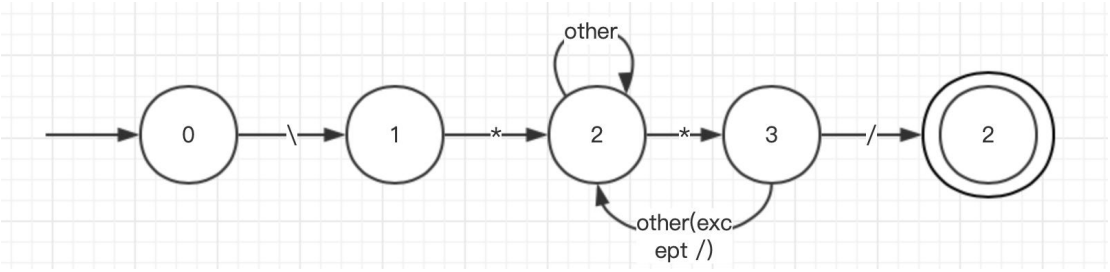
针对字符串处理的自动机如下图所示，其中状态 3 用以处理转义字符：



字符的处理类似

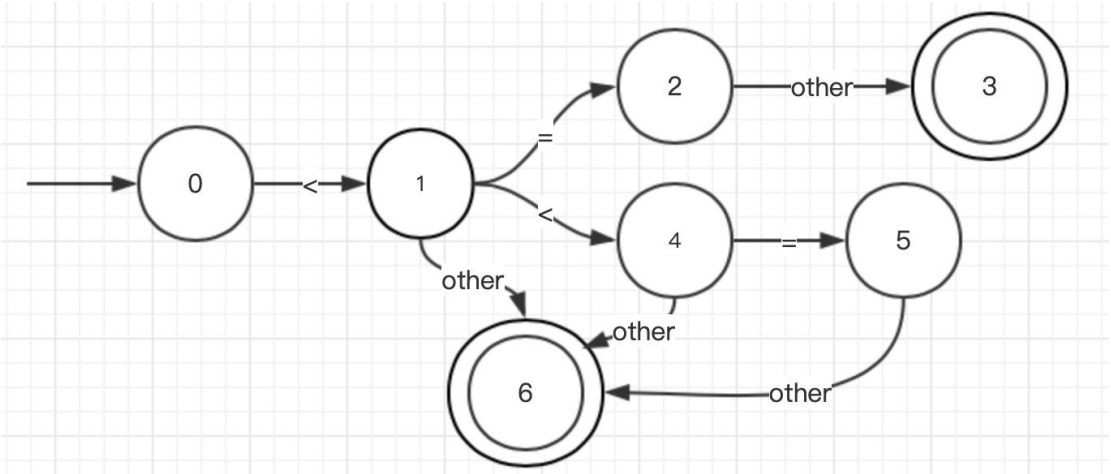
4.对注释的处理

处理注释的自动机如下图所示：



5. 对符号，运算符的处理

大致环节与下图类似，该图展示了 <, <=, <<= 三种符号的识别过程：



因为实际的符号与运算符较多，不再一一在图中列出。

若读入字符不在符号、运算符定义当中，又不是注释，字符串的内容，将进入错误处理，触发 `InvalidTok` 错误。详细细节见下文。

5.对错误的处理

1) `InvalidTok` 错误：读入字符未定义

2)`NumConvWro` 错误：数字转换错误。诸如 123ADE，或者是 1.EE1 将被定义为数字转换错误。换言之，在 `fm_deal_with_num` 函数中，所有不被自动机接收的字符串都会触发数字转换错误。

3) `Overflow` 错误： 输入数字超过上限。在 deal_with_num 该上限对于整形来说是 long 的范围，对于浮点数来说是 double 范围。在 `fm_deal_with_num` 中该上限是数字字符长度超过 `100.`（即缓存长度）

4) `StrTooLarge` 错误：字符串常量输入超过上限，该上限是 100 个字符（可在宏定义中修改）。

5) `InvalidCha` 错误：字符常量中有超过一个以上的字符。(转义字符\n 算一个字符)

6) `pairing error` 错误：注释部分只有/* 而没有*/，字符串常量只有单边引号“

当遇到以上错误时，错误处理函数 error()将记录错误所在行数与列数，并前移指针直到一个空格或一个换行符出现为止（即舍弃掉出错的单词）。

6.统计

程序将输出文件的字符个数，语句行数，各类单词个数，并输出统计结果。

五.测试样例

部分测试样例参照 https://www.rosettacode.org/wiki/Compiler/lexical_analyzer

Test case 1:

```
/*
  Hello world
 */
print("Hello, World!\n");
```

输出:

```
line num:4  chracter num:47
operator num:0  symbol num:3    constant num:1  keyword num:1   variable num:0
                 name        symbol    property
                print       keyword           -
                    (        symbol   LeftParen
   "Hello, World!\n"      constant      string
                    )        symbol  RightParen
                    ;        symbol   Semicolon
error Report:   0 error

Process finished with exit code 0
```

Test case 2:

```
/*
  Show Ident and Integers
 */
phoenix_number = 142857;
print(phoenix_number, "\n");
```

输出:

```
line num:5  chracter num:86
operator num:1  symbol num:5    constant num:2  keyword num:1   variable num:2
                name        symbol    property
      phoenix_number            ID        0--0
                   =            op      assign
              142857      constant         int
                   ;        symbol   Semicolon
               print       keyword           -
                   (        symbol   LeftParen
      phoenix_number            ID        0--0
                   ,        symbol       Comma
                "\n"      constant      string
                   )        symbol  RightParen
                   ;        symbol   Semicolon
error Report:   0 error

Process finished with exit code 0
```

Test case 3:

```
/*
  All lexical tokens - not syntactically correct, but that will
  have to wait until syntax analysis
 */
/* Print   */  print   /* Sub    */  -
/* Putc    */  putc    /* Lss    */  <
/* If      */  if      /* Gtr    */  >
/* Else    */  else    /* Leq    */  <=
/* While   */  while   /* Geq    */  >=
/* Lbrace  */  {       /* Eq     */  ==
/* Rbrace  */  }       /* Neq    */  !=
/* Lparen  */  (       /* And    */  &&
/* Rparen  */  )       /* Or     */  ||
/* Uminus  */  -       /* Semi   */  ;
/* Not     */  !       /* Comma  */  ,
/* Mul     */  *       /* Assign */  =
/* Div     */  /       /* Integer */  42
/* Mod     */  %       /* String  */  "String literal"
/* Add     */  +       /* Ident   */  variable_name
/* character literal */  '\n'
/* character literal */  '\\'
/* character literal */  ' '
```

输出:

```
line num:22 chracter num:845
operator num:16 symbol num:6    constant num:5  keyword num:4   variable num:2
                name        symbol    property
               print       keyword           -
                   -            op    subtract
                putc            ID        0--2
                   <            op        less
                  if       keyword           -
                   >            op     greater
                else       keyword           -
                  <=            op   lessequal
               while       keyword           -
                  >=            op  greaterequal
                   {        symbol   LeftBrace
                  ==            op       equal
                   }        symbol  RightBrace
                  !=            op    notequal
                   (        symbol   LeftParen
                  &&            op         and
                   )        symbol  RightParen
                  ||            op          or
                   -            op    subtract
                   ;        symbol   Semicolon
                   !            op         not
                   ,        symbol       Comma
                   *            op    multiply
                   =            op      assign
                   /            op      divide
                  42      constant         int
                   %            op         mod
    "String literal"      constant      string
                   +            op         add
       variable_name            ID       1--29
                '\n'      constant     literal
                '\\'      constant     literal
                 ' '      constant     literal
error Report:   0 error
```

Test case 4:

```
int a = 1;
int b = 1234;
int max =
12314567890123145678901231456789012314567890123145678901231456789012314567890123145678901231456789012314567890123145678901231456789012314567890123145678901231456789012314567890;
int wow = 123w213;
int $wdad = 123;
```

输出:

```
line num:5  chracter num:249
operator num:5  symbol num:3    constant num:3  keyword num:5   variable num:4
                name        symbol    property
                 int       keyword           -
                   a            ID        0--1
                   =            op      assign
                   1      constant         int
                   ;        symbol   Semicolon
                 int       keyword           -
                   b            ID        1--6
                   =            op      assign
                1234      constant         int
                   ;        symbol   Semicolon
                 int       keyword           -
                 max            ID       2--11
                   =            op      assign
                 int       keyword           -
                 wow            ID       3--14
                   =            op      assign
                 int       keyword           -
                   =            op      assign
                   1      constant         int
                   ;        symbol   Semicolon
error Report:   3 error
Wrong on line 3 col 101 Overflow
Wrong on line 4 col 14  NumConvWro
Wrong on line 5 col 5   InvalidTok

Process finished with exit code 0
```

Test case 5:

```
int a = 1;
a++;
int b = 0x10;
b++;
/*This is a test for pairing error
```

输出:

```
line num:6  chracter num:70
operator num:4  symbol num:4    constant num:2  keyword num:2   variable num:4
                name        symbol    property
                 int       keyword           -
                   a            ID        0--1
                   =            op      assign
                   1      constant         int
                   ;        symbol   Semicolon
                   a            ID        0--1
                  ++            op     selfAdd
                   ;        symbol   Semicolon
                 int       keyword           -
                   b            ID        1--9
                   =            op      assign
                0x10      constant         int
                   ;        symbol   Semicolon
                   b            ID        1--9
                  ++            op     selfAdd
                   ;        symbol   Semicolon
error Report:   1 error
Wrong on line 6 col 1   pairing error
```