

北京邮电大学

设计报告



课程名称: 编译原理与技术

实验名称: 词法分析

学 院: 计算机学院

班 级: 2022211312

学 号: 2022211404

姓 名: 唐梓楠

2024 年 10 月 20 日

目录

1	什么是 lex/flex?	2
1.1	程序设计说明	2
1.2	测试报告	2
2	用 flex 生成 PL 语言的词法分析器	2
2.1	程序设计说明	2
2.1.1	具体实现	3
2.2	测试报告	3
2.2.1	识别 CHARCON	3
3	词法分析程序的设计与实现 (C/C++)	5
3.1	程序设计说明	5
3.1.1	配对缓冲区实现	6
3.1.2	常量表数据结构	7
3.1.3	完整转换图	7
3.2	测试报告	9
3.2.1	对合法的不带后缀的八进制数与十六进制数的识别	9
3.2.2	对合法的带后缀的十进制数的识别	9
3.2.3	考虑完整的浮点型常量的识别	9
3.2.4	带前缀的字符串常量的识别	9
3.2.5	完整的字符常量的识别	10
3.2.6	字符串	10
4	线上测试例通过情况截图	10

1 什么是 lex/flex?

1.1 程序设计说明

使用 lex/flex 实现对以小写字母 ab 结尾的字符串 (只包含大小写字母) 的识别, 如 Helloab 和 Goab.

1.2 测试报告

该题较为简单, 使用正则表达式表达以 ab 结尾的纯字母字符串即可. 通过参考 [flex 手册的第七章](#), 可以得到如 Figure 1 所示的正则表达式规则.

符号	含义
	或
[]	括号中的字符取其一
-	a-z表示ascii码中介于a-z包括a.z的字符
\	转义 (flex不能识别除字母外的字符)
*	0或多个字符
?	0或1个字符
+	1或多个字符
^	除此之外的其余字符
.	除\n外的所有字符, 等价于^\n

图 1: flex patterns

根据规则, 可以构造出表达式即:

```
1  /* begin */
2  [a-zA-Z]*ab    {printf("%s: Hit!\n", yytext);}
3  /* end */
```

需要注意的是, 使用 flex 时正则表达式前面不能有空格或缩进, 否则会报错.

2 用 flex 生成 PL 语言的词法分析器

2.1 程序设计说明

利用 flex 工具生成 PL 语言的词法分析器, 要求输入一个 PL 语言源程序文件 demo.pl, 输出一个文件 tokens.txt, 该文件包括每一个单词及其种别枚举值, 每行一个单词.

此外还有一个类别 **ERROR**, 包括不是出现在字符串中的非法字符, 当词法分析器读到非法字符时, 应该输出 **ERROR** 作为种别值.

2.1.1 具体实现

PL 语言单词的符号及其种别值如 Figure 2 所示:

单词符号	种别枚举值	单词符号	种别枚举值
标识符	IDENT]	RBRACK
整常量	INTCON	(LPAREN
字符常量	CHARCON)	RPAREN
+	PLUS	,	COMMA
-	MINUS	;	SEMICOLON
*	TIMES	.	PERIOD
/	DIVSYM	:=	BECOME
=	EQL	:	COLON
<>	NEQ	begin	BEGINSYM
<	LSS	end	ENDSYM
<=	LEQ	if	IFSYM
>	GTR	then	THENSYM
>=	GEQ	else	ELSESYM
of	OFSYM	while	WHILESYM
array	ARRAYSYM	do	DOSYM
program	PROGRAMSYM	call	CALLSYM
mod	MODSYM	const	CONSTSYM
and	ANDSYM	type	TYPESYM
or	ORSYM	var	VARSYM
not	NOTSYM	procedure	PROCSYM
[LBRACK		

图 2: PL 语言单词符号及其种别值

可以构建 PL 单词的状态图如 Figure 3 所示.

2.2 测试报告

大部分状态按照状态图, 根据 flex 语言的正则表达式模式就可以轻松完成. 需要注意的是优先级顺序, 应该将优先级高的对应正则表达式子放在前面, 低的放在后面, 最后处理 **ERROR**.

比较困难的是对于字符常量 **CHARCON** 的识别.

2.2.1 识别 CHARCON

输入: 'shuebqbdhh_fye788893\da''

运行结果: CHARCON

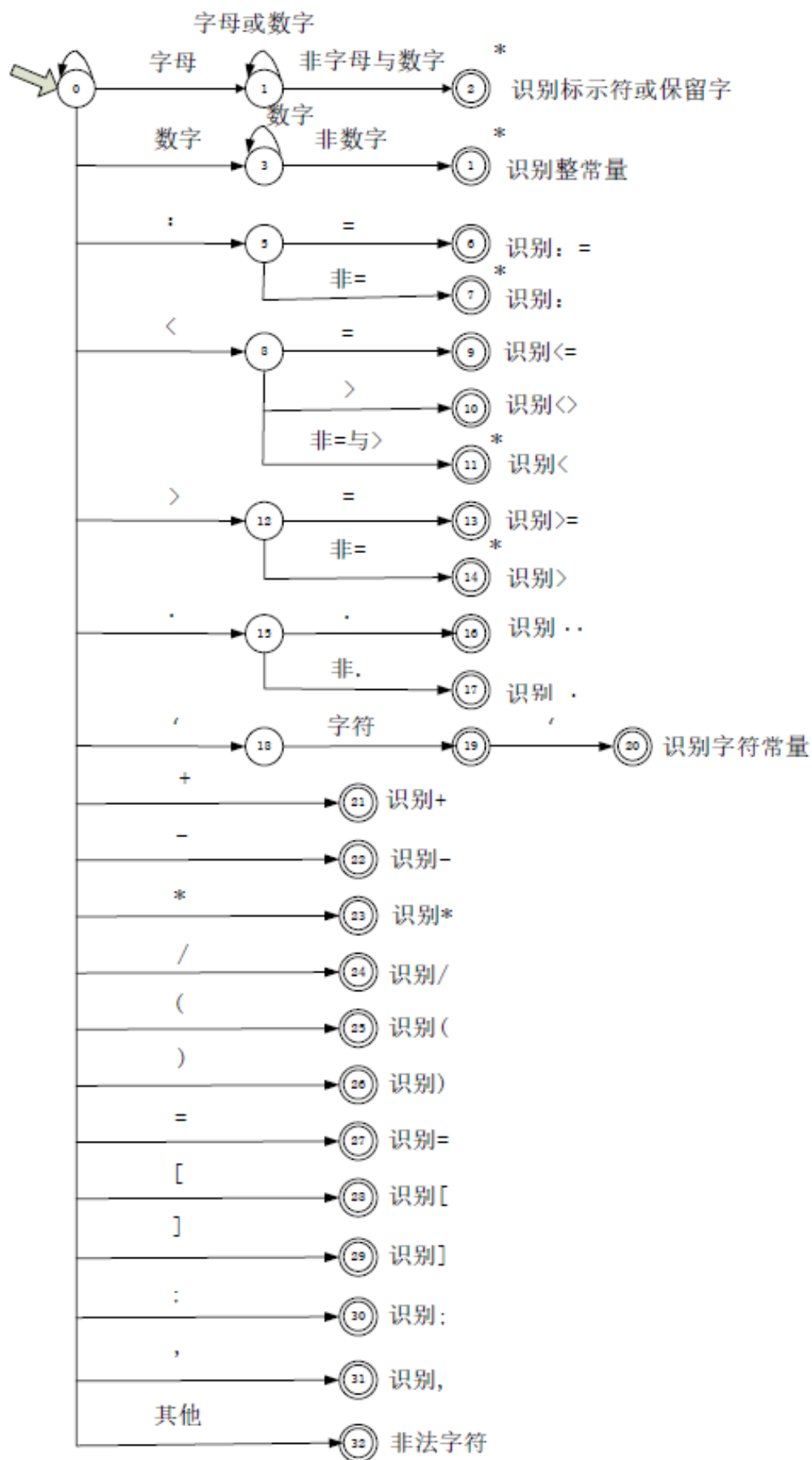


图 3: PL 单词状态图

分析说明: 首先用 `\'` 匹配单引号 `'`，即字符常量的开头和结尾。重点在于内部字符常量的内容识别逻辑。

- `[^'\\]` 表明这是一个字符类，用来匹配任何不是单引号 `'` 和反斜杠 `\` 的字符。也就是说，这里匹配除了单引号和反斜杠以外的普通字符。
- `\\.` 表示匹配反斜杠 `\` 后跟任意字符的情况。由于字符常量可能包括转义字符，如 `\'` 或 `\\`，因此这一部分用于处理这些转义字符。与前者用 `|` 连接。
- 用 `*` 表示前面部分可以匹配零次或多次，即允许字符常量的内容部分可以包含任意长度的符合规则的字符。

设计出的正则表达式为:

1

CHARCON

`\'([^\'\\]|\\.)*\'`

该正则表达式可以匹配由单引号包围的字符常量，内容部分可以是除单引号和反斜杠外的任意字符，或者是反斜杠后跟的转义字符（如 `\'`，`\\` 等）。整个表达式确保字符常量以单引号开头和结尾，并且允许其中包含合法的转义序列。

3 词法分析程序的设计与实现 (C/C++)

3.1 程序设计说明

设计并实现一个 C 语言的词法分析程序。

要求:

1. 识别单词符号并以记号形式输出，并标出该单词符号所在行数。应考虑的单词符号见输出格式；
2. 能够识别并跳过注释；
3. 能够检查到错误的词法；
4. 能够统计行数、各个单词符号的类别数，以及词法错误数。

本题的状态转化图与书上类似，因此我以书上的状态转化实现为基础，如 Figure 4 所示，进行拓展完善，实现了自动机的状态转化，并实现了与教材一致的配对缓冲区。设计思路也保持与教材实现一致。具体可参考代码。

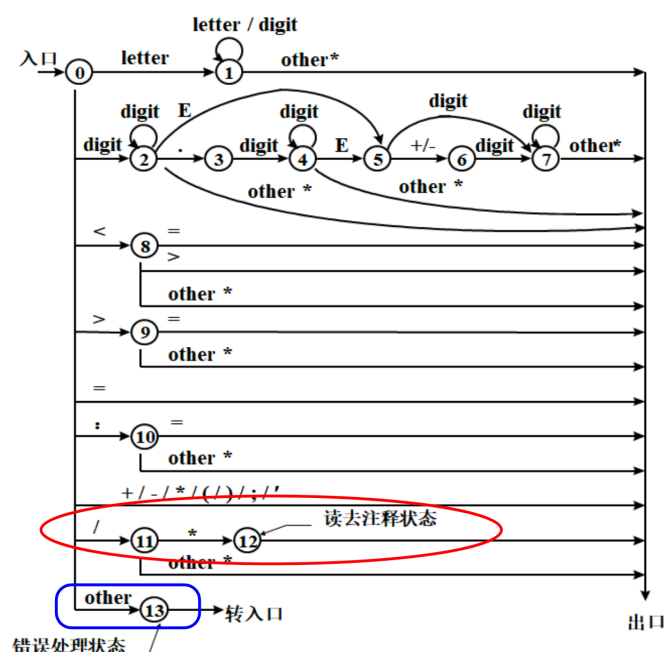


图 4: 基本状态图

3.1.1 配对缓冲区实现

缓冲器分为大小相同的两半, 每半包含 N 个字符, 且每半区带有结束标记, 减少比较次数.

```

1 void get_char()
2 {
3     if (buffer[forwardptr] == '\0') {
4         if (forwardptr == HALF_SIZE - 1) {
5             file.read(buffer + HALF_SIZE, HALF_SIZE - 1);
6             add_eof(HALF_SIZE);
7             ++forwardptr;
8         } else if (forwardptr == BUFFER_SIZE - 1) {
9             file.read(buffer, HALF_SIZE - 1);
10            add_eof(0);
11            forwardptr = 0;
12        }
13
14        if (buffer[forwardptr] == '\0') {

```

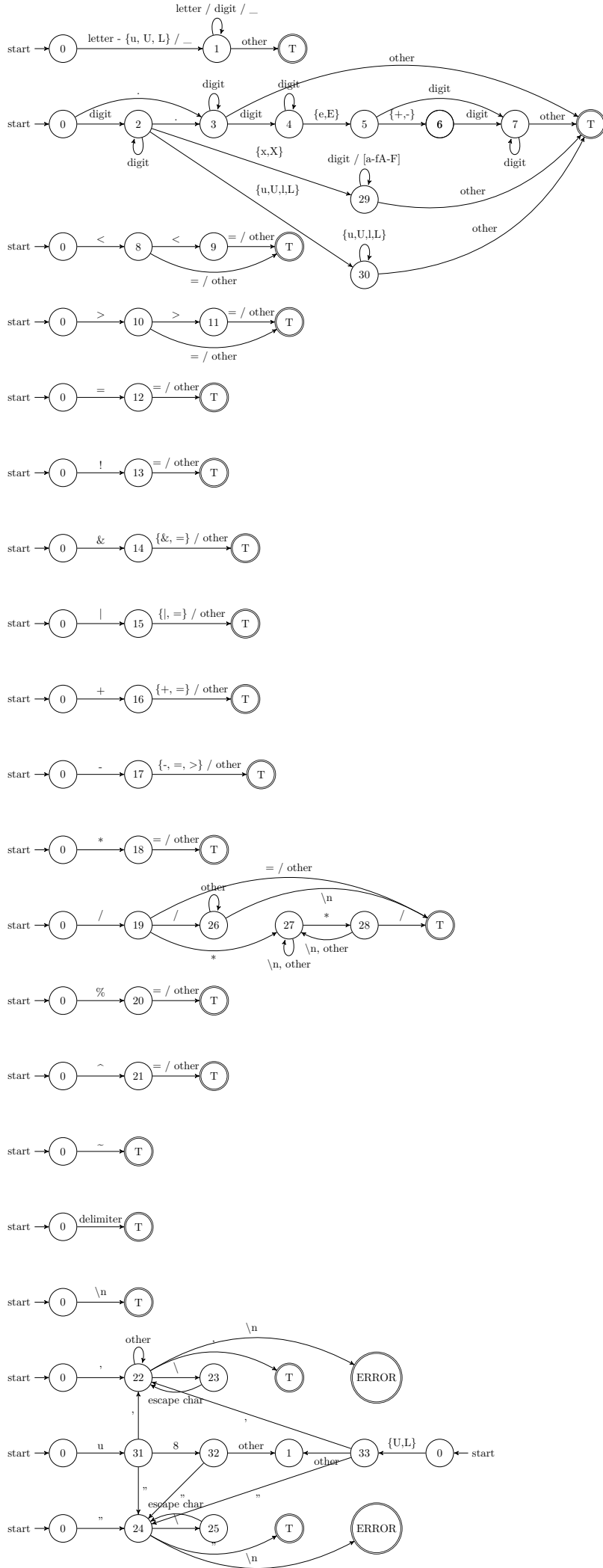
```
15         file.close();
16         cout << line_cnt << '\n';
17         cout << keywords_cnt << ' ' << identifiers_cnt << ' ' <<
           ↪ operators_cnt << ' ' << delimiters_cnt << ' ' << charcons_cnt
           ↪ << ' ' << strings_cnt << ' ' << numbers_cnt << '\n';
18         cout << errors_cnt;
19         exit(0);
20     }
21 }
22 C = buffer[forwardptr];
23 ++forwardptr;
24 }
```

3.1.2 常量表数据结构

在存储如关键字和保留字常量表时, 为了实现高效率, 我们使用了 STL 中的 `unordered_set` 来存储, 其内部采用哈希表实现, 能够在 $O(1)$ 的时间复杂度内完成查找操作, 相较于普通的 `set` 有性能上的优势.

3.1.3 完整转换图

根据 C11 语言 (参考草案: [ISO/IEC 9899:201x](#)) 标准, 我们可以构造出如下的状态转移图:



根据状态转移图, 我们可以很轻松的用 `switch case` 语句实现有限状态自动机.

3.2 测试报告

基础的前 20 个测试集较为简单, 只需对照状态转化图实现即可, 我们主要分析扩展要求中的 21-25 测试集.

3.2.1 对合法的不带后缀的八进制数与十六进制数的识别

输入: `0x1ABCD2F`

输出: `<NUMBER,0X3abcde2>`

分析说明: 由于八进制数与十六进制数的特殊性, 我们需要对其进行特殊处理. 八进制数以 0 开头, 十六进制数以 0x 或 0X 开头. 识别到开头后, 循环识别数字即可, 需要注意的是, 十六进制数字可能包含 A-F 或 a-f.

3.2.2 对合法的带后缀的十进制数的识别

输入: `107LLu`

输出: `<NUMBER,107LLu>`

分析说明: 带后缀的十进制数以 L 或 l 结尾, 也可以以 U 或 u 结尾, 也可以同时出现. 识别到数字后, 循环识别后缀即可.

3.2.3 考虑完整的浮点型常量的识别

输入: `.001e2f`

输出: `<NUMBER,.001e2f>`

分析说明: 相较于一般的浮点数, 这里完整的浮点数识别还包括后缀, 用同样的方法添加后缀识别即可. 此外, 还需要注意可以省略最开头的 0, 因此识别到 `.` 时, 需要增加对数字的跳转.

3.2.4 带前缀的字符串常量的识别

输入: `u8"ABC"`

输出: `<STRING,u8"ABC">`

分析说明: 带前缀的字符串常量以 u8, u, U, L 开头, 识别到前缀后, 识别到 `"` 后, 循环识别字符串即可. 因此不能将上述几个字符作为普通的 `letter` 处理, 应该进行额外的判断.

3.2.5 完整的字符常量的识别

输入: 'abc'

输出: <CHARCON, 'abc'>

分析说明: 这里定义的字符常量比较奇怪, 可以包括任意长度的字符, 和字符串区别不大, 原来在这里纠结了很久为什么可以这么表达, 查阅了 C 语言的相关资料也并没有相关的定义, 因此这里只能最后只能按照题目要求进行实现. 最后字符常量 CHARCON 和字符串 STRING 的差异不大了, 区别体现在对换行的报错.

3.2.6 字符串

输入:

1

2


```
1  /*
2  */
```

输出: 无

分析说明: 用 /* 开头, 且用 */ 结尾的注释是支持换行的, 但是测试集中并没有涉及到换行的情况.

4 线上测试例通过情况截图

总体评价



唐梓楠

按时通关

学号: 2022211404

分班: 2022211312

截止前完成关卡: 3/3

最新完成关卡: 3/3

完成率: --

课堂最高完成率: --

通关时间	计时规则	实训总耗时	评测次数	查重扣分	补交扣分	最终成绩	总评
2024-09-30 17:21	页面停留时长	3 时 55 分 52 秒	97	--	--	100.0/100.0	优秀

阶段成绩

关卡	任务名称	开启时间	代码修改行数	评测次数	完成时间	实训耗时	是否查看答案	经验值	关卡得分	调分
1	什么是 lex/flex?	2024-09-23 20:51	1	1	2024-09-23 21:14	22 分 16 秒	否	100/100	33.33/33.33	33.33
2	用 flex 生成 PL 语言的词法分析器	2024-09-23 21:14	83	33	2024-09-24 08:30	1 时 28 分 12 秒	否	400/400	33.33/33.33	33.33
3	词法分析程序的设计与实现 (C/C++)	2024-09-24 08:31	920	63	2024-09-30 17:21	2 时 5 分 24 秒	否	500/500	33.34/33.33	33.34

图 5: 全部通过 (包括附加内容)