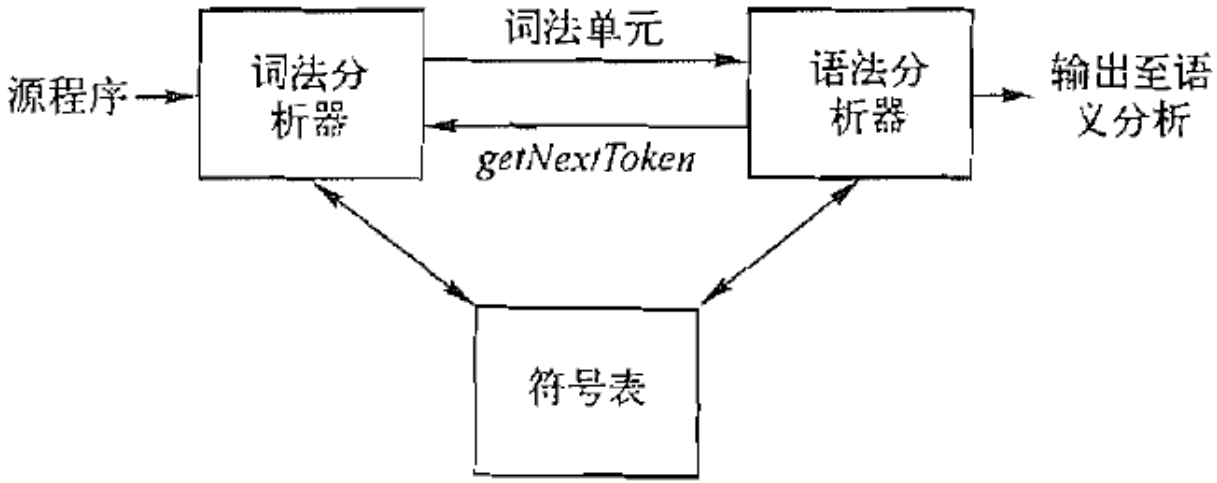
(1. ANTLR4)

hfwei@nju.edu.cn

20211105 ()

: / s & **(token)**



:

```
CharStream input = CharStreams.fromStream(is);
SimpleExprLexer lexer = new SimpleExprLexer(input);

CommonTokenStream tokens = new CommonTokenStream(lexer);

SimpleExprParser parser = new SimpleExprParser(tokens);
ParseTree tree = parser.prog();
```

SimpleExpr.g4

```c
int main(void)
{
    printf("hello, world\n");
}
```

```c
int main(void)
{
    printf("hello, world\n");
}
```

int ws **main/id** **LP** **void** **RP** ws

LB ws

ws **id** **LP** **literal** **RP** **SC** ws

RB

```c
int main(void)
{
    printf("hello, world\n");
}
```

int ws **main/id** **LP** **void** **RP** ws

LB ws

ws **id** **LP** **literal** **RP** **SC** ws

RB

, (/)

()

(gcc)

master

gcc / gcc / c-family / **c-lex.c**

iains Objective-C/C++ : Improve '@' keyword locations. ···

19 contributors +7

1435 lines (1278 sloc) | 38.8 KB

master

gcc / libcpp / **lex.c**

urnathan cpplib: EOF in pragmas ···

25 contributors +13

4364 lines (3825 sloc) | 119 KB


ANTLR

:

```
antlr4 SimpleExpr.g4
```

:

```
antlr4 SimpleExpr.g4
```

:

- `SimpleExprLexer.java`
- `SimpleExpr.tokens`

:

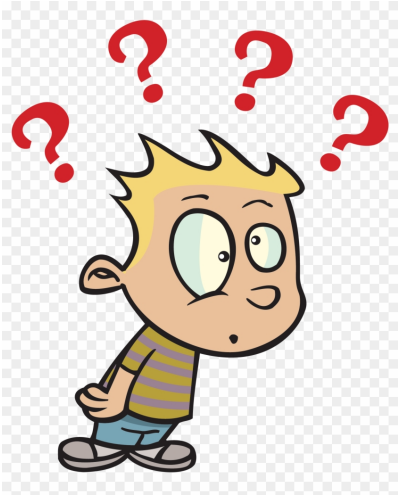
```
antlr4 SimpleExpr.g4
```

:

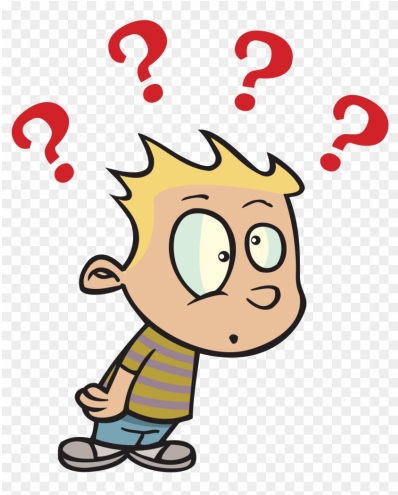
- SimpleExprLexer.java
- SimpleExpr.tokens

```
javac SimpleExpr*.java
grun simpleexpr.SimpleExpr prog -tokens
```

词法单元	非正式描述	词素示例
if	字符 i, f	if
else	字符 e, l, s, e	else
comparison	< 或 > 或 <= 或 >= 或 == 或 !=	<=, !=
id	字母开头的字母 / 数字串	pi, score, D2
number	任何数字常量	3.14159, 0, 6.02e23
literal	在两个 "之间，除" 以外的任何字符	"core dumped"



词法单元	非正式描述	词素示例
if	字符 i, f	if
else	字符 e, l, s, e	else
comparison	< 或 > 或 <= 或 >= 或 == 或 !=	<=, !=
id	字母开头的字母 / 数字串	pi, score, D2
number	任何数字常量	3.14159, 0, 6.02e23
literal	在两个 "之间，除" 以外的任何字符	"core dumped"



词法单元	非正式描述	词素示例
if	字符 i, f	if
else	字符 e, l, s, e	else
comparison	< 或 > 或 <= 或 >= 或 == 或 !=	<=, !=
id	字母开头的字母 / 数字串	pi, score, D2
number	任何数字常量	3.14159, 0, 6.02e23
literal	在两个 "之间，除" 以外的任何字符	"core dumped"

id: /

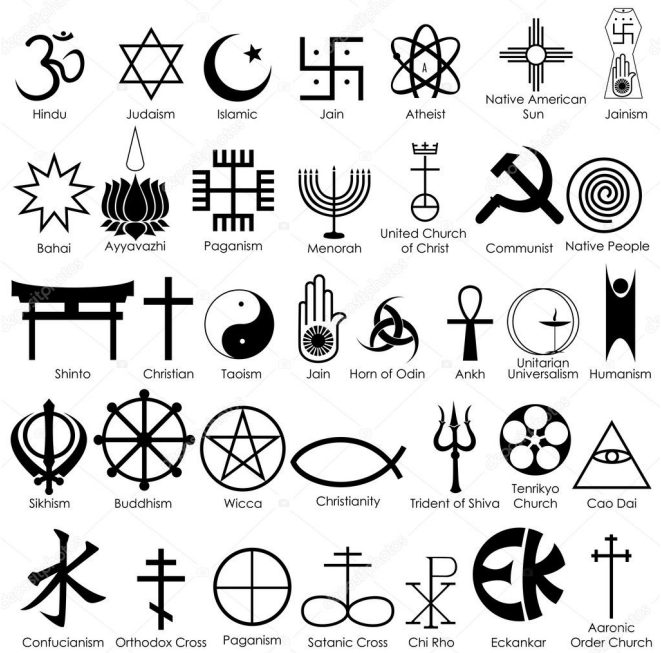
id , **(Language)**

, **(Alphabet)**

(,) **(String)**

Definition ()

Σ



Definition ()

Σ (s) Σ

: $|\epsilon| = 0$

Definition (“”)

$$x = dog, y = house \qquad xy = doghouse$$

$$s\epsilon = \epsilon s = s$$

Definition (“”)

$$x = dog, y = house \qquad xy = doghouse$$

$$s\epsilon = \epsilon s = s$$

Definition (“”)

$$s^0 \triangleq \epsilon$$

$$s^i \triangleq ss^{i-1}, i > 0$$

Definition ()

Σ

$$\emptyset$$

$$\{\epsilon\}$$

Definition ()

Σ

$$\emptyset$$

$$\{\epsilon\}$$

$$\textbf{id} : \{a, b, c, a1, a2, \dots\}$$

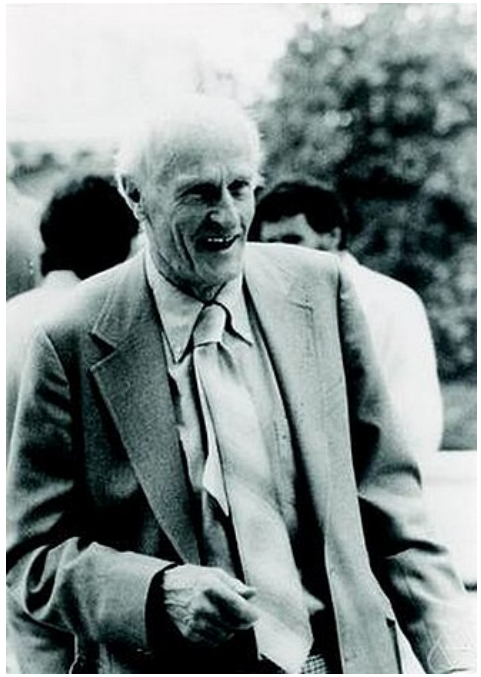
$$\textbf{ws} : \{\textbf{blank}, \textbf{tab}, \textbf{newline}\}$$

$$\textbf{if} : \{\texttt{if}\}$$

,

运算	定义和表示
L和M 的并	$L \cup M \doteq \{s \mid s$ 属于L 或者s 属于 $M\}$
L和M 的连接	$LM = \{st \mid s$ 属于 L且　t 属于 $M\}$
L的 Kleene 闭包	$L^* = \cup_{i=0}^{\infty} L^i$
L的正闭包	$L^+ = \cup_{i=1}^{\infty} L^i$

L^* (L^+)



Stephen Kleene
(1909 $\sim$ 1994)

$$L = \{A, B, \ldots, Z, a, b, \ldots, z\}$$

$$D = \{0, 1, \ldots, 9\}$$

运算	定义和表示
L和M 的并	$L \cup M \doteq \{s \mid s$ 属于L 或者s 属于 $M\}$
L和M 的连接	$LM = \{st \mid s$ 属于 L且　t 属于 $M\}$
L的 Kleene 闭包	$L^* = \cup_{i=0}^{\infty} L^i$
L的正闭包	$L^+ = \cup_{i=1}^{\infty} L^i$

$$L = \{A, B, \ldots, Z, a, b, \ldots, z\}$$

$$D = \{0, 1, \ldots, 9\}$$

运算	定义和表示
L和M 的并	$L \cup M \doteq \{s \mid s$ 属于L 或者s 属于 $M\}$
L和M 的连接	$LM = \{st \mid s$ 属于 L且　t 属于 $M\}$
L的 Kleene 闭包	$L^* = \cup_{i=0}^{\infty} L^i$
L的正闭包	$L^+ = \cup_{i=1}^{\infty} L^i$

$$L \cup D \qquad LD \qquad L^4 \qquad L^* \qquad D^+$$

$$L(L \cup D)^*$$

$$L = \{A, B, \ldots, Z, a, b, \ldots, z\}$$

$$D = \{0, 1, \ldots, 9\}$$

运算	定义和表示
L和M 的并	$L \cup M \doteq \{s \mid s$ 属于L 或者s 属于 $M\}$
L和M 的连接	$LM = \{st \mid s$ 属于 L且　t 属于 $M\}$
L的 Kleene 闭包	$L^* = \cup_{i=0}^{\infty} L^i$
L的正闭包	$L^+ = \cup_{i=1}^{\infty} L^i$

$$L \cup D \qquad LD \qquad L^4 \qquad L^* \qquad D^+$$

$$L(L \cup D)^*$$

$$\textbf{id} : L(L \cup D)^*$$

antlr4: **id** ?

id : $L(L \cup D)^*$

antlr4: **id** ?

r $L(r)$

,

Definition ()

Σ, Σ :

(1) ϵ ;

(2) $\forall a \in \Sigma$, a ;

(3) r , (r) ;

(4) r s , $r|s$, rs, r^*

: $() \succ * \succ \succ |$

$$(a)|((b)^*(c)) \equiv a|b^*c$$

r $L(r)$

Definition ()

$$L(\epsilon) = \{\epsilon\} \tag{1}$$

$$L(a) = \{a\}, \forall a \in \Sigma \tag{2}$$

$$L((r)) = L(r) \tag{3}$$

$$L(r|s) = L(r) \cup L(s) \quad L(rs) = L(r)L(s) \quad L(r^*) = (L(r))^* \tag{4}$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$L((a|b)(a|b))$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$L((a|b)(a|b))$$

$$L(a^*)$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$L((a|b)(a|b))$$

$$L(a^*)$$

$$L((a|b)^*)$$

$$\Sigma = \{a, b\}$$

$$L(a|b) = \{a, b\}$$

$$L((a|b)(a|b))$$

$$L(a^*)$$

$$L((a|b)^*)$$

$$L(a|a^*b)$$


So
EASY

表达式	匹配	例子
c	单个非运算符字符 c	a
\c	字符 c 的字面值	*
"s"	串 s 的字面值	"**"
.	除换行符以外的任何字符	a.*b
^	一行的开始	^abc
$	行的结尾	abc$
[s]	字符串 s 中的任何一个字符	[abc]
[^s]	不在串 s 中的任何一个字符	[^abc]
$r*$	和 r 匹配的零个或多个串连接成的串	a*
$r+$	和 r 匹配的一个或多个串连接成的串	a+
$r?$	零个或一个 r	a?
$r\{m,n\}$	最少 m 个，最多 n 个 r 的重复出现	a{1,5}
r_1r_2	r_1 后加上 r_2	ab
$r_1 \mid r_2$	r_1 或 r_2	a|b
(r)	与 r 相同	(a|b)
r_1/r_2	后面跟有 r_2 时的 r_1	abc/123

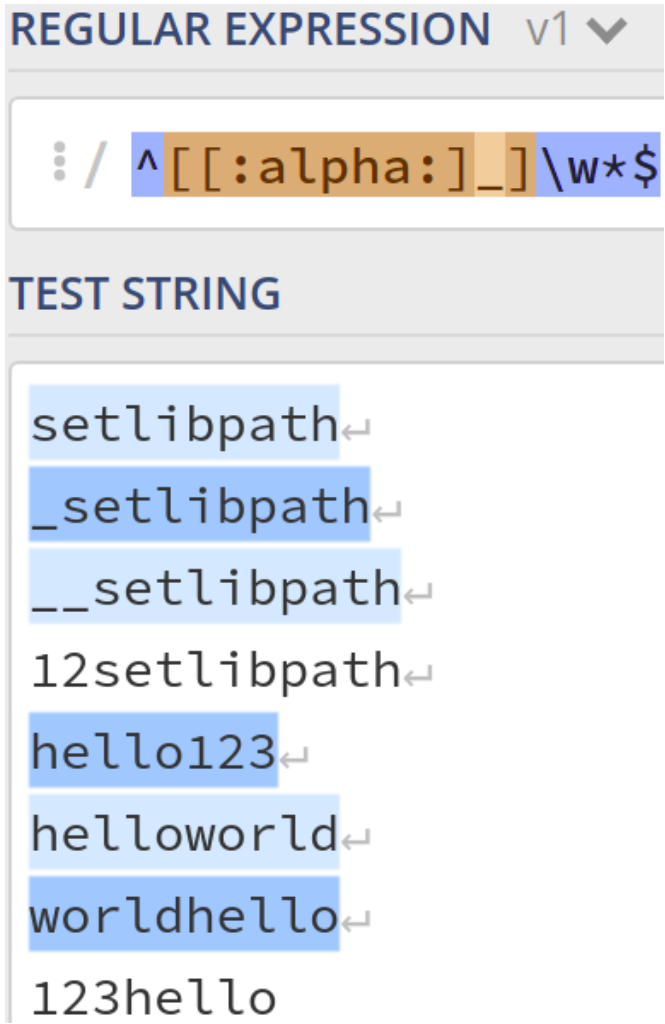
表达式	匹配	例子
c	单个非运算符字符 c	a
\c	字符 c 的字面值	*
"s"	串 s 的字面值	"**"
.	除换行符以外的任何字符	a.*b
^	一行的开始	^abc
$	行的结尾	abc$
[s]	字符串 s 中的任何一个字符	[abc]
[^s]	不在串 s 中的任何一个字符	[^abc]
$r*$	和 r 匹配的零个或多个串连接成的串	a*
$r+$	和 r 匹配的一个或多个串连接成的串	a+
$r?$	零个或一个 r	a?
$r\{m,n\}$	最少 m 个，最多 n 个 r 的重复出现	a{1,5}
r_1r_2	r_1 后加上 r_2	ab
$r_1 \mid r_2$	r_1 或 r_2	a|b
(r)	与 r 相同	(a|b)
r_1/r_2	后面跟有 r_2 时的 r_1	abc/123

$[0-9]$ $[a-zA-Z]$ ^ $

Vim	Java	ASCII	Description
	\p{ASCII}	[\x00-\x7F]	ASCII characters
	\p{Alnum}	[A-Za-z0-9]	Alphanumeric characters
\w	\w	[A-Za-z0-9_]	Alphanumeric characters plus "_"
\W	\W	[^A-Za-z0-9_]	Non-word characters
\a	\p{Alpha}	[A-Za-z]	Alphabetic characters
\s	\p{Blank}	[\t]	Space and tab
\< \>	\b	(?<=\W)(?=\w)|(?<=\w)(?=\W)	Word boundaries
	\B	(?<=\W)(?=\W)|(?<=\w)(?=\w)	Non-word boundaries
	\p{Cntrl}	[\x00-\x1F\x7F]	Control characters
\d	\p{Digit} or \d	[0-9]	Digits
\D	\D	[^0-9]	Non-digits
	\p{Graph}	[\x21-\x7E]	Visible characters
\l	\p{Lower}	[a-z]	Lowercase letters
\p	\p{Print}	[\x20-\x7E]	Visible characters and the space character
	\p{Punct}	[][!"#$%&'()*+,./:;<=>?@\^_`{|}~-]	Punctuation characters
_s	\p{Space} or \s	[\t\r\n\v\f]	Whitespace characters
\S	\S	[^ \t\r\n\v\f]	Non-whitespace characters
\u	\p{Upper}	[A-Z]	Uppercase letters
\x	\p{XDigit}	[A-Fa-f0-9]	Hexadecimal digits

C ?

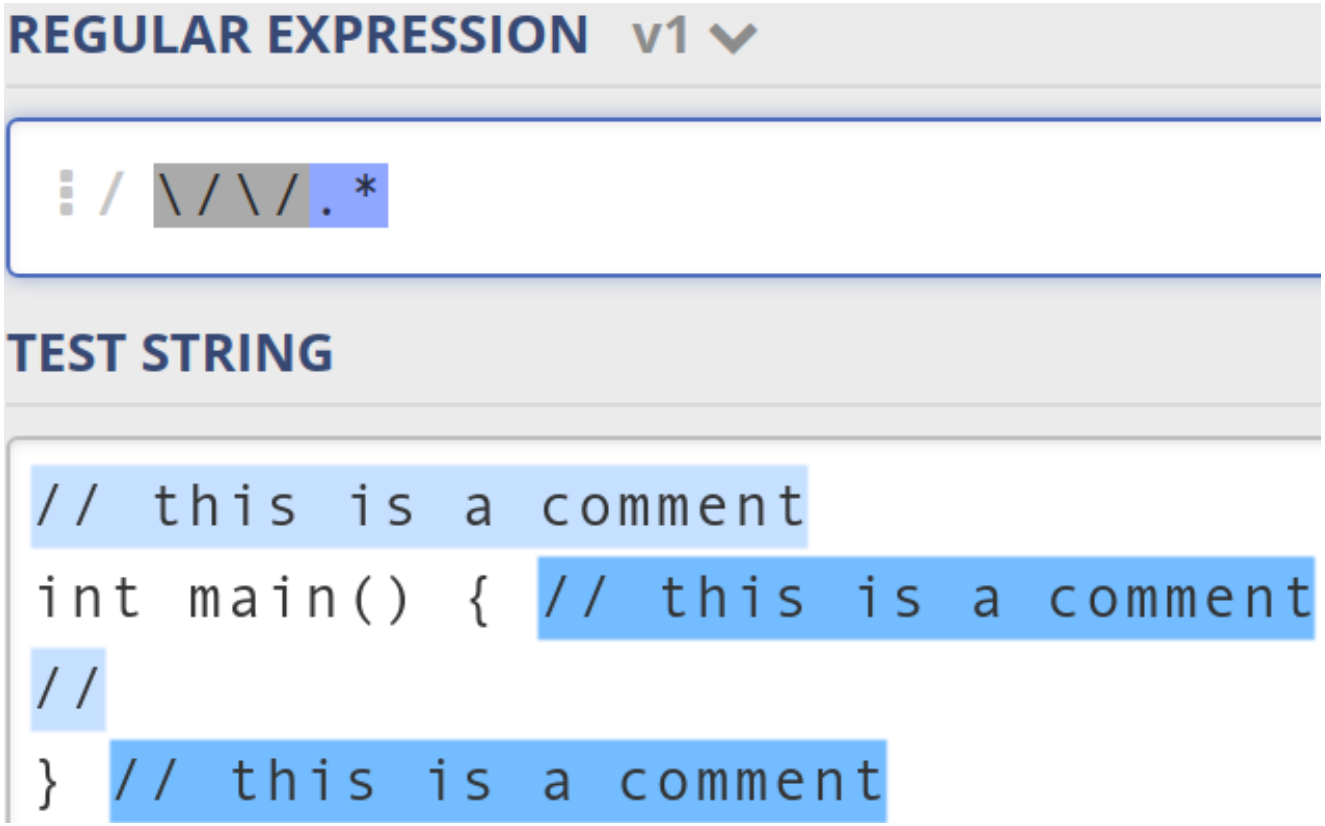
C ?



https://regex101.com/r/lVoghd/1

C ?

C ?



https://regex101.com/r/ED4qgC/2

$L1$?

一种是使用“/*”以及“*/”进行多行注释，在这种情况下，在“/*”与之后最先遇到的“*/”之间的所有字符都被视作注释内容。需要注意的是，“/*”与“*/”是不允许嵌套的：即在任意一对“/*”和“*/”之间不能再包含成对的“/*”和“*/”，否则编译器需要进行报错。

$L1$?

一种是使用“/*”以及“*/”进行多行注释，在这种情况下，在“/*”与之后最先遇到的“*/”之间的所有字符都被视作注释内容。需要注意的是，“/*”与“*/”是不允许嵌套的：即在任意一对“/*”和“*/”之间不能再包含成对的“/*”和“*/”，否则编译器需要进行报错。

```
LCOMMENT : '//' .*? '\n' -> skip;    // non-greedy

MLCOMMENT : '/*' .*? '*/' -> skip;   // non-greedy
```

$L1$?

一种是使用“/*”以及“*/”进行多行注释，在这种情况下，在“/*”与之后最先遇到的“*/”之间的所有字符都被视作注释内容。需要注意的是，“/*”与“*/”是不允许嵌套的：即在任意一对“/*”和“*/”之间不能再包含成对的“/*”和“*/”，否则编译器需要进行报错。

```
LCOMMENT : '//' .*? '\n' -> skip;    // non-greedy

MLCOMMENT : '/*' .*? '*/' -> skip;   // non-greedy
```

antlr4 , ’.’ ()

$$\Big(0|\big(1(01^*0)^*1\big)\Big)^*$$

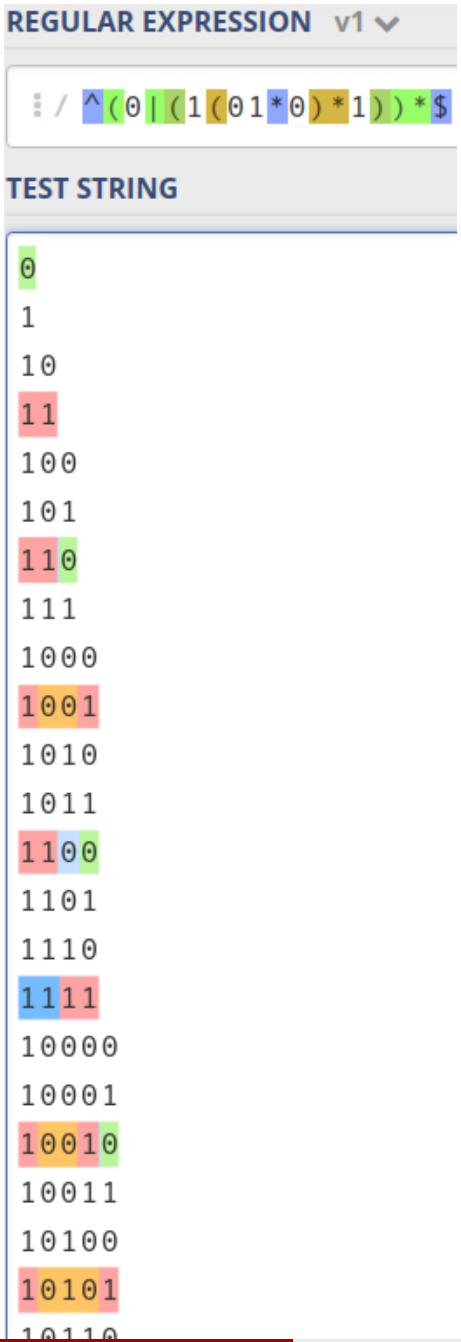
https://regex101.com/r/ED4qgC/1

REGULAR EXPRESSION v1

/ ^(0|(1(01*0)*1))*$

TEST STRING

0
1
10
11
100
101
110
111
1000
1001
1010
1011
1100
1101
1110
1111
10000
10001
10010
10011
10100
10101


REGULAR EXPRESSION v1
/ ^(0|(1(01*0)*1))*$
TEST STRING
0
1
10
11
100
101
110
111
1000
1001
1010
1011
1100
1101
1110
1111
10000
10001
10010
10011
10100
10101

3 ()

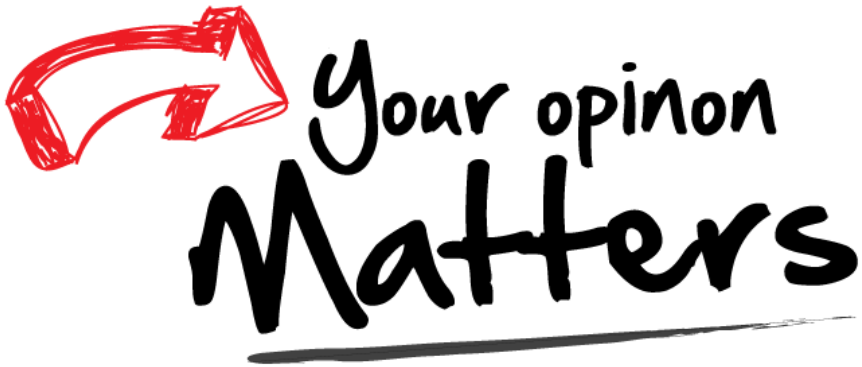
```
IF : 'if' ;
ID : (LETTER | '_') WORD* ;
INT : DIGIT | (NONZERODIGIT DIGIT+) ;
WS  : [ \t\r\n]+ -> skip ;

ASSIGN : '=' ;
LBRACE : '(' ;
RBRACE : ')' ;
ADD : '+' ;
SUB : '-' ;
MUL : '*' ;
DIV : '/' ;
```

Antlr4

: *vs.*

: `>=`, `ifhappy`, `thenext`, `1.23`


Thank
You!



Office 926

hfwei@nju.edu.cn