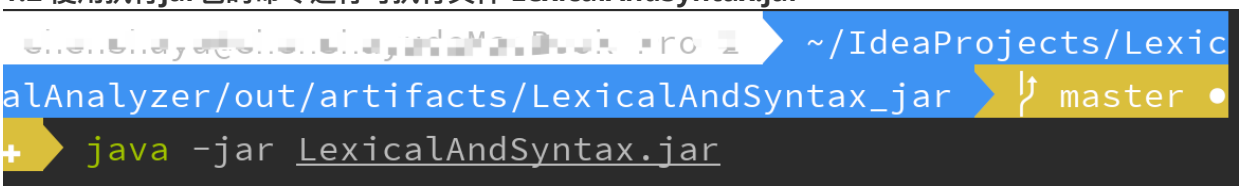
LL1 语法分析使用说明

`jar` `terminal`

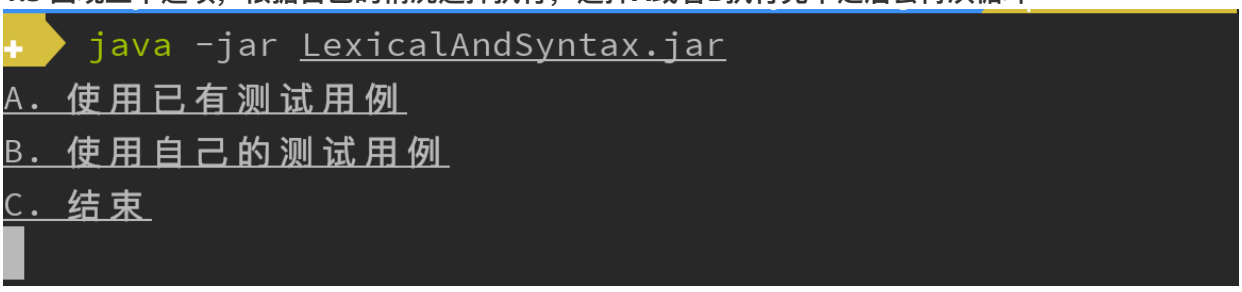
1. 执行

1.1 进入jar包所在目录

1.2 使用执行jar包的命令运行可执行文件 LexicalAndSyntax.jar



1.3 出现三个选项，根据自己的情况选择执行，选择A或者B执行完毕之后会再次循环



2.选择

- A. 使用已有测试用例

会直接使用内置的文法和测试用例，输出词法分析的结果和语法分析的每一步

- B. 使用自己的测试用例

需要再输入测试用例所在绝对路径

- C. 结束

退出程序的执行。

```
A. 使用已有测试用例
B. 使用自己的测试用例
C. 结束
C
```

3. 输出

- 会输出构造预测分析表的if else冲突和词法分析的结果

```
else_result|else|$|else elseif_stmt
elseif_stmt|if|block_stmt|if_control_stmts
void: keyword
main: keyword
(: boundary
): boundary
{: boundary
int: keyword
i: IDN
```

- 最后输出预测分析的每一步的符号栈和动作
  - 归约：用产生式
  - 移进：终结符匹配
  - 语法分析成功或者失败

  部分截图如下：

```
符号栈：[# S]    所用产生式： S -> funcs
符号栈：[# funcs]    所用产生式： funcs -> func funcs
符号栈：[# funcs func]    所用产生式： func -> void IDN ( arg ) func_body
符号栈：[# funcs func_body ) arg ( IDN void]    移进： void
符号栈：[# funcs func_body ) arg ( IDN]    移进： IDN
符号栈：[# funcs func_body ) arg (]    移进： (
符号栈：[# funcs func_body ) arg]    所用产生式： arg -> $
符号栈：[# funcs func_body )]    移进： )
符号栈：[# funcs func_body]    所用产生式： func_body -> block
符号栈：[# funcs block]    所用产生式： block -> { statements }
符号栈：[# funcs } statements {]    移进： {
符号栈：[# funcs } statements]    所用产生式： statements -> statement statements
符号栈：[# funcs } statements statement]    所用产生式： statement -> init_exp ;
符号栈：[# funcs } statements ; init_exp]    所用产生式： init_exp -> type IDN fun_v
alue_exp
符号栈：[# funcs } statements ; fun_value_exp IDN type]    所用产生式： type -> int
符号栈：[# funcs } statements ; fun_value_exp IDN int]    移进： int
符号栈：[# funcs } statements ; fun_value_exp IDN]    移进： IDN
符号栈：[# funcs } statements ; fun_value_exp]    所用产生式： fun_value_exp -> init
符号栈：[# funcs } statements ; init]    所用产生式： init -> = expression
符号栈：[# funcs } statements ; expression =]    移进： =
符号栈：[# funcs } statements ; expression]    所用产生式： expression -> value exp
符号栈：[# funcs } statements ; exp value]    所用产生式： value -> item value1
符号栈：[# funcs } statements ; exp value1 item]    所用产生式： item -> INT
符号栈：[# funcs } statements ; exp value1 INT]    移进： INT
符号栈：[# funcs } statements ; exp value1]    所用产生式： value1 -> $
```

```
符号栈：[# funcs } statements ; expression]    所用产生式： expression -> value exp
符号栈：[# funcs } statements ; exp value]    所用产生式： value -> item value1
符号栈：[# funcs } statements ; exp value1 item]    所用产生式： item -> IDN fun_val
ue
符号栈：[# funcs } statements ; exp value1 fun_value IDN]    移进： IDN
符号栈：[# funcs } statements ; exp value1 fun_value]    所用产生式： fun_value -> $
符号栈：[# funcs } statements ; exp value1]    所用产生式： value1 -> $
符号栈：[# funcs } statements ; exp]    所用产生式： exp -> $
符号栈：[# funcs } statements ;]    移进： ;
符号栈：[# funcs } statements]    所用产生式： statements -> $
符号栈：[# funcs }]    移进： }
符号栈：[# funcs]    所用产生式： funcs -> $
符号栈：[#]    大公告成，匹配成功辽～
移进： #
```

`ps` 除了使用命令行执行jar包，也可以用IDEA等JAVA IDE打开源代码运行