



## CS 315: Programming Languages

Lexical Analyser for a Programming Language  
for an Integer Language

Language: *HazardCat*

13.10.2034

Section: 2

Group: 14?

❖ Musa Yiğit Yayla. 22003108

❖ Maryam Azimli. 22101528

Instructor:

Aynur Dayanik

# Bilkent University | Department of Computer Engineering

1.  $\langle \text{main} \rangle ::= \text{execute begin } \langle \text{program} \rangle \text{ end}$
2.  $\langle \text{program} \rangle ::= \langle \text{statements} \rangle$
3.  $\langle \text{statements} \rangle ::= \langle \text{statement} \rangle | \langle \text{statements} \rangle + \langle \text{statement} \rangle$
4.  $\langle \text{statement} \rangle ::= \langle \text{cond\_statement} \rangle | \langle \text{loop} \rangle |$   
 $\langle \text{single\_statement} \rangle | \langle \text{statement} \rangle$
5.  $\langle \text{cond\_statement} \rangle ::= \text{if } (\langle \text{exprs} \rangle) \{ \langle \text{statements} \rangle \} | \text{else\_if } (\langle \text{exprs} \rangle) \{ \langle \text{statements} \rangle \} | \text{else } \{ \langle \text{statements} \rangle \}$
6.  $\langle \text{loop} \rangle ::= \langle \text{for} \rangle | \langle \text{while} \rangle$
7.  $\langle \text{for} \rangle ::= \text{for } (\langle \text{varName} \rangle = \langle \text{expr} \rangle ; \langle \text{cond\_statement} \rangle) \{ \langle \text{statements} \rangle \} ;$
8.  $\langle \text{while} \rangle ::= \text{while } (\langle \text{expr} \rangle) \{ \text{statements} \} ;$
9.  $\langle \text{single\_statement} \rangle ::= \langle \text{varDeclaration} \rangle | \langle \text{return\_st} \rangle | \langle \text{arrDec} \rangle | \langle \text{varAssign} \rangle$   
 $\rangle | \langle \text{constIntDecAssign} \rangle | \langle \text{constStringDecAssign} \rangle |$   
 $\langle \text{varDecAssign} \rangle | \langle \text{func\_call} \rangle$
10.  $\langle \text{varDeclaration} \rangle ::= \text{let } \langle \text{varName} \rangle = \langle \text{expr} \rangle ;$
11.  $\langle \text{varAssign} \rangle ::= \langle \text{varName} \rangle = \langle \text{number} \rangle ;$
12.  $\langle \text{varDecAssign} \rangle := \text{let } \langle \text{varAssign} \rangle ;$
13.  $\langle \text{constIntDecAssign} \rangle ::= \text{const } \langle \text{varName} \rangle = \langle \text{number} \rangle ;$
14.  $\langle \text{constStringDecAssign} \rangle ::= \text{string } \langle \text{varName} \rangle = \langle \text{string} \rangle ;$
15.  $\langle \text{return\_st} \rangle ::= \text{return } \langle \text{exprs} \rangle ;$
16.  $\langle \text{arrDec} \rangle ::= \text{list } \langle \text{varName} \rangle ;$
17.  $\langle \text{arraySizeSpecifier} \rangle ::= \sim$
18.  $\langle \text{func\_call} \rangle ::= \text{func } \text{varName}(\text{parameters}) \{ \langle \text{statements} \rangle \langle \text{return\_st} \rangle \}$
19.  $\langle \text{exprs} \rangle ::= \langle \text{expr} \rangle + \langle \text{expr} \rangle | \langle \text{expr} \rangle * \langle \text{expr} \rangle | \text{expr} ;$
20.  $\langle \text{expr} \rangle ::= \langle \text{varName} \rangle | \langle \text{varName} \rangle + \langle \text{expr} \rangle | \langle \text{constant} \rangle ;$
21.  $\langle \text{varName} \rangle ::= \langle \text{lower\_lets} \rangle | \langle \text{lower\_lets} \rangle + \langle \text{upper\_let} \rangle | \langle \text{lower\_lets} \rangle + \langle \text{upper\_lets} \rangle + \langle \text{nums} \rangle ;$
22.  $\langle \text{parameters} \rangle ::= \text{int } \langle \text{varName} \rangle | \text{int } \langle \text{varName} \rangle , \langle \text{parameters} \rangle$

## Bilkent University | Department of Computer Engineering

- 23.<read>::=read <read\_sign> <exprs>;
- 24.<read\_sign>::= <<
- 25.<write>::= write <write\_sign> <exprs>;
- 26.<write\_sign>::= >>
- 27.<string>::=<lower\_lets>|upper\_lets>|<digits>|<special\_buts>
- 28.<lower\_lets>::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z |<lower\_lets>
- 29.<upper\_let>::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|<upper\_let>
- 30.<special\_buts>::=/.|,|\*|&|^|%|\$|#|@|!|~
- 31.<compare>::=< | > | <= | >= | == | !=
- 32.<increment>::=<varName>++|++<varName>|++
- 33.<decrement>::=<varName>--|--<varName>|--
- 34.<plusEqual><varName>+=
- 35.<minusEqual><varName>-=
- 36.<negative>::=-(number)
- 37.<arithmeticOperator >::= <sum>| <subtract>| <multiply>| <divide>| <mod> | <pow>
- 38.<sum>::=<varName>+<varName>|<varName>+<constant>|<constant> + <constant>
- 39.<subtract>::=<varName>-<varName>|<varName>-<constant>| <constant> - <constant>
- 40.<multiply>::=<varName>\*<varName>|<varName>\*<constant>| <constant> \* <constant>
- 41.<divide>::=<varName>/<varName>|<varName>/<constant>| <constant> / <constant>
- 42.<mod>::=<varName>%<varName>|<varName>%<constant>| <constant> % <constant>

## Bilkent University | Department of Computer Engineering

- 43. <pow>::=<varName>^<varName>|<varName>^<constant>|  
    <constant> ^ <constant>
- 44. <boolOperator>::= <not>|<or>|<and>|<xor>
- 45. <not>::=!
- 46. <or>::= ||
- 47. <and>::= &&
- 48. <xor>::= ^^
- 49. <number>::=<digits><digits>|<digits>| <negative><digits>|<negative>
- 50. <digits>::=0|1|2|3|4|5|6|7|8|9
- 51. <comments>::= #<string>
- 52. <print>::=print(<number>|<expr>|<string>)
- 53. <print\_line>::=print\_line+<print>