

### Original Grammar:

$\langle \text{program} \rangle ::= \langle \text{fdecls} \rangle \langle \text{declarations} \rangle \langle \text{statement\_seq} \rangle.$   
 $\langle \text{fdecls} \rangle ::= \langle \text{fdec} \rangle; \mid \langle \text{fdecls} \rangle \langle \text{fdec} \rangle; \mid \epsilon$   
 $\langle \text{fdec} \rangle ::= \text{def } \langle \text{type} \rangle \langle \text{fname} \rangle (\langle \text{params} \rangle) \langle \text{declarations} \rangle \langle \text{statement\_seq} \rangle \text{ fed}$   
 $\langle \text{params} \rangle ::= \langle \text{type} \rangle \langle \text{var} \rangle \mid \langle \text{type} \rangle \langle \text{var} \rangle, \langle \text{params} \rangle \mid \epsilon$   
 $\langle \text{fname} \rangle ::= \langle \text{id} \rangle$   
 $\langle \text{declarations} \rangle ::= \langle \text{decl} \rangle; \mid \langle \text{declarations} \rangle \langle \text{decl} \rangle; \mid \epsilon$   
 $\langle \text{decl} \rangle ::= \langle \text{type} \rangle \langle \text{varlist} \rangle$   
 $\langle \text{type} \rangle ::= \text{int} \mid \text{double}$   
 $\langle \text{varlist} \rangle ::= \langle \text{var} \rangle, \langle \text{varlist} \rangle \mid \langle \text{var} \rangle$   
 $\langle \text{statement\_seq} \rangle ::= \langle \text{statement} \rangle \mid \langle \text{statement} \rangle; \langle \text{statement\_seq} \rangle$   
 $\langle \text{statement} \rangle ::= \langle \text{var} \rangle = \langle \text{expr} \rangle \mid \text{if } \langle \text{bexpr} \rangle \text{ then } \langle \text{statement\_seq} \rangle \text{ fi} \mid \text{if } \langle \text{bexpr} \rangle \text{ then } \langle \text{statement\_seq} \rangle \text{ else } \langle \text{statement\_seq} \rangle \text{ fi} \mid \text{while } \langle \text{bexpr} \rangle \text{ do } \langle \text{statement\_seq} \rangle \text{ od} \mid$   
 $\text{print } \langle \text{expr} \rangle \mid \text{return } \langle \text{expr} \rangle \mid \epsilon$   
 $\langle \text{expr} \rangle ::= \langle \text{expr} \rangle + \langle \text{term} \rangle \mid \langle \text{expr} \rangle - \langle \text{term} \rangle \mid \langle \text{term} \rangle$   
 $\langle \text{term} \rangle ::= \langle \text{term} \rangle * \langle \text{factor} \rangle \mid \langle \text{term} \rangle / \langle \text{factor} \rangle \mid \langle \text{term} \rangle \% \langle \text{factor} \rangle \mid \langle \text{factor} \rangle$   
 $\langle \text{factor} \rangle ::= \langle \text{var} \rangle \mid \langle \text{number} \rangle \mid (\langle \text{expr} \rangle) \mid \langle \text{fname} \rangle (\langle \text{exprseq} \rangle)$   
 $\langle \text{exprseq} \rangle ::= \langle \text{expr} \rangle, \langle \text{exprseq} \rangle \mid \langle \text{expr} \rangle \mid \epsilon$   
 $\langle \text{bexpr} \rangle ::= \langle \text{bexpr} \rangle \text{ or } \langle \text{bterm} \rangle \mid \langle \text{bterm} \rangle$   
 $\langle \text{bterm} \rangle ::= \langle \text{bterm} \rangle \text{ and } \langle \text{bfactor} \rangle \mid \langle \text{bfactor} \rangle$   
 $\langle \text{bfactor} \rangle ::= (\langle \text{bexpr} \rangle) \mid \text{not } \langle \text{bfactor} \rangle \mid (\langle \text{expr} \rangle \langle \text{comp} \rangle \langle \text{expr} \rangle)$   
 $\langle \text{comp} \rangle ::= < \mid > \mid == \mid <= \mid >= \mid <>$   
 $\langle \text{var} \rangle ::= \langle \text{id} \rangle \mid \langle \text{id} \rangle [\langle \text{expr} \rangle]$   
 $\langle \text{letter} \rangle ::= [\text{a-z}]$   
 $\langle \text{digit} \rangle ::= [0-9]$   
 $\langle \text{id} \rangle ::= \langle \text{letter} \rangle \mid \langle \text{id} \rangle \langle \text{letter} \rangle \mid \langle \text{id} \rangle \langle \text{digit} \rangle$   
 $\langle \text{number} \rangle ::= \langle \text{integer} \rangle \mid \langle \text{double} \rangle$

### Grammar in LL(1) Form:

$\langle \text{program} \rangle ::= \langle \text{fdecls} \rangle \langle \text{declarations} \rangle \langle \text{statement\_seq} \rangle.$   
 $\langle \text{fdecls} \rangle ::= \langle \text{fdec} \rangle; \mid \langle \text{fdecls}' \rangle \mid \epsilon$   
 $\langle \text{fdecls}' \rangle ::= \langle \text{fdecls} \rangle \mid \epsilon$   
 $\langle \text{fdec} \rangle ::= \text{def } \langle \text{type} \rangle \langle \text{fname} \rangle (\langle \text{params} \rangle) \langle \text{declarations} \rangle \langle \text{statement\_seq} \rangle \text{ fed}$   
 $\langle \text{params} \rangle ::= \langle \text{type} \rangle \langle \text{var} \rangle \langle \text{params}' \rangle \mid \epsilon$   
 $\langle \text{params}' \rangle ::= , \langle \text{params} \rangle \mid \epsilon$   
 $\langle \text{fname} \rangle ::= \langle \text{id} \rangle$   
 $\langle \text{declarations} \rangle ::= \langle \text{decl} \rangle; \mid \langle \text{declarations}' \rangle \mid \epsilon$   
 $\langle \text{declarations}' \rangle ::= \langle \text{declarations} \rangle \mid \epsilon$   
 $\langle \text{decl} \rangle ::= \langle \text{type} \rangle \langle \text{varlist} \rangle$   
 $\langle \text{type} \rangle ::= \text{int} \mid \text{double}$

$\langle \text{varlist} \rangle ::= \langle \text{var} \rangle \langle \text{varlist}' \rangle$   
 $\langle \text{varlist}' \rangle ::= , \langle \text{varlist}' \rangle \mid \epsilon$   
 $\langle \text{statement\_seq} \rangle ::= \langle \text{statement} \rangle \langle \text{statement\_seq}' \rangle$   
 $\langle \text{statement\_seq}' \rangle ::= ; \langle \text{statement\_seq}' \rangle \mid \epsilon$   
 $\langle \text{statement} \rangle ::= \langle \text{var} \rangle = \langle \text{expr} \rangle \mid \text{if } \langle \text{bexpr} \rangle \text{ then } \langle \text{statement\_seq} \rangle \langle \text{statement}' \rangle \mid \text{while}$   
 $\langle \text{bexpr} \rangle \text{ do } \langle \text{statement\_seq} \rangle \text{ od} \mid \text{print } \langle \text{expr} \rangle \mid \text{return } \langle \text{expr} \rangle \mid \epsilon$   
 $\langle \text{statement}' \rangle ::= \text{fi} \mid \text{else } \langle \text{statement\_seq} \rangle \text{ fi}$   
 $\langle \text{expr} \rangle ::= \langle \text{term} \rangle \langle \text{expr}'' \rangle$   
 $\langle \text{expr}' \rangle ::= + \langle \text{term} \rangle \mid - \langle \text{term} \rangle$   
 $\langle \text{expr}'' \rangle ::= \langle \text{expr}' \rangle \langle \text{expr}'' \rangle \mid \epsilon$   
 $\langle \text{term} \rangle ::= \langle \text{factor} \rangle \langle \text{term}'' \rangle$   
 $\langle \text{term}' \rangle ::= * \langle \text{factor} \rangle \mid / \langle \text{factor} \rangle \mid \% \langle \text{factor} \rangle$   
 $\langle \text{term}'' \rangle ::= \langle \text{term}' \rangle \langle \text{term}'' \rangle \mid \epsilon$   
 $\langle \text{factor} \rangle ::= \langle \text{id} \rangle \langle \text{factor}' \rangle \mid \langle \text{number} \rangle \mid (\langle \text{expr} \rangle)$   
 $\langle \text{factor}' \rangle ::= \langle \text{var}' \rangle \mid (\langle \text{exprseq} \rangle)$   
 $\langle \text{exprseq} \rangle ::= \langle \text{expr} \rangle \langle \text{exprseq}' \rangle \mid \epsilon$   
 $\langle \text{exprseq}' \rangle ::= , \langle \text{exprseq}' \rangle \mid \epsilon$   
 $\langle \text{bexpr} \rangle ::= \langle \text{bterm} \rangle \langle \text{bexpr}' \rangle$   
 $\langle \text{bexpr}' \rangle ::= \text{or } \langle \text{bterm} \rangle \langle \text{bexpr}' \rangle \mid \epsilon$   
 $\langle \text{bterm} \rangle ::= \langle \text{bfactor} \rangle \langle \text{bterm}' \rangle$   
 $\langle \text{bterm}' \rangle ::= \text{and } \langle \text{bfactor} \rangle \langle \text{bterm}' \rangle \mid \epsilon$   
 $\langle \text{bfactor} \rangle ::= (\langle \text{bfactor}' \rangle \mid \text{not } \langle \text{bfactor} \rangle$   
 $\langle \text{bfactor}' \rangle ::= (\langle \text{bfactor}' \rangle \langle \text{bterm}' \rangle \langle \text{bexpr}' \rangle) \mid \text{not } \langle \text{bfactor} \rangle \langle \text{bterm}' \rangle \langle \text{bexpr}' \rangle) \mid$   
 $\langle \text{term} \rangle \langle \text{expr}'' \rangle \langle \text{comp} \rangle \langle \text{expr} \rangle)$   
 $\langle \text{comp} \rangle ::= < \mid > \mid == \mid <= \mid >= \mid <>$   
 $\langle \text{var} \rangle ::= \langle \text{id} \rangle \langle \text{var}' \rangle$   
 $\langle \text{var}' \rangle ::= [\langle \text{expr} \rangle] \mid \epsilon$   
 $\langle \text{letter} \rangle ::= [\text{a-z}]$   
 $\langle \text{digit} \rangle ::= [0-9]$   
 $\langle \text{id} \rangle ::= \langle \text{letter} \rangle \langle \text{id}'' \rangle$   
 $\langle \text{id}' \rangle ::= \langle \text{letter} \rangle \mid \langle \text{digit} \rangle$   
 $\langle \text{id}'' \rangle ::= \langle \text{id}' \rangle \langle \text{id}'' \rangle \mid \epsilon$   
 $\langle \text{number} \rangle ::= \langle \text{integer} \rangle \mid \langle \text{double} \rangle$

#### First and Follow Sets:

SYMBOL	FIRST	FOLLOW
$\langle \text{program} \rangle$	. ; [a-z] def double if int print return while	\$
$\langle \text{fdecls} \rangle$	def $\epsilon$	. ; [a-z] double if int print

		return while
<fdecls'>	def $\epsilon$	. ; [a-z] double if int print return while
<fdec>	def	;
<params>	double int $\epsilon$	)
<params'>	, $\epsilon$	)
<fname>	[a-z]	(
<declarations>	double int $\epsilon$	. ; [a-z] fed if print return while
<declarations'>	double int $\epsilon$	. ; [a-z] fed if print return while
<decl>	double int	;
<type>	double int	[a-z]
<varlist>	[a-z]	;
<varlist'>	, $\epsilon$	;
<statement_seq>	; [a-z] if print return while $\epsilon$	. else fed fi od
<statement_seq'>	; $\epsilon$	. else fed fi od
<statement>	[a-z] if print return while $\epsilon$	. ; else fed fi od
<statement'>	else fi	. ; else fed fi od
<expr>	( [a-z] doubleNum integerNum	) , . ; ] else fed fi od
<expr'>	+ -	) + , - . ; < <= <> == > >= ] else fed fi od
<expr''>	+ - $\epsilon$	) , . ; < <= <> == > >= ] else fed fi od
<term>	( [a-z] doubleNum integerNum	) + , - . ; < <= <> == > >= ] else fed fi od
<term'>	% * /	% ) * + , - . / ; < <= <> == > >= ] else fed fi od

<term''>	% * / $\epsilon$	) + , - . ; < <= <> == > >= ] else fed fi od
<factor>	( [a-z] doubleNum integerNum	% ) * + , - . / ; < <= <> == > >= ] else fed fi od
<factor'>	( [ $\epsilon$	% ) * + , - . / ; < <= <> == > >= ] else fed fi od
<exprseq>	( [a-z] doubleNum integerNum $\epsilon$	)
<exprseq'>	, $\epsilon$	)
<bexpr>	( not	do then
<bexpr'>	or $\epsilon$	) do then
<bterm>	( not	) do or then
<bterm'>	and $\epsilon$	) do or then
<bfactor>	( not	) and do or then
<bfactor'>	( [a-z] <double> <integer> not	) and do or then
<comp>	< <= <> == > >=	( [a-z] doubleNum integerNum
<var>	[a-z]	) , ; =
<var'>	[ $\epsilon$	% ) * + , - . / ; < <= <> = == > >= ] else fed fi od
<letter>	[a-z]	% ( ) * + , - . / ; < <= <> = == > >= [ [0-9] [a-z] ] else fed fi od
<digit>	[0-9]	% ( ) * + , - . / ; < <= <> = == > >= [ [0-9] [a-z] ] else fed fi od
<id>	[a-z]	% ( ) * + , - . / ; < <= <> = == > >= [ ] else fed fi od
<id'>	[0-9] [a-z]	% ( ) * + , - . / ; < <= <> = == > >= [ [0-9] [a-z] ] else fed fi od

<id">	[0-9] [a-z] €	% ( ) * + , - . / ; < <= <> = == > >= [ ] else fed fi od
<number>	<integer> <double>	% ) * + , - . / ; < <= <> == > >= ] else fed fi od