Program	::=	Command	**Program**
Command	::=	V-name :=Expression	**AssignCommand**
	|	V-name (Actual-Parameter-Sequence)	**CallCommand**
	|	Command ; Command	**SequentialCommand**
	|	if Expression then Single-Command else Single-Command	**IfCommand**
	|	while Expression do Single-Command	**WhileCommand**
	|	let Declaration in Single-Command	**LetCommand**
	|	begin Command end	**BeginCommand**
Expression	::=	let Declaration in Expression	**LetExpression**
	|	if Expression then Expression else Expression	**IfExpression**
	|	Expression Operator Expression	**BinaryExpression**
	|	Integer-Literal	**IntegerExpression**
	|	Character-Literal	**CharacterExpression**
	|	V-name	**VnameExpression**
	|	Identifier (Actual-Parameter-Sequence)	**CallExpression**
	|	Operator Primary-Expression	**UnaryExpression**
	|	(Expression)	**Expression**
V-name	::=	Identifier	**SimpleVname**
Declaration	::=	Declaration (; Declaration)*	**Declaration or SequentialDeclaration**
	|	const Identifier ~ Expression	**ConstDeclaration**
	|	var Identifier : Type-Denoter	**VarDeclaration**
Actual-Parameter-Sequence	:==	(Actual-Parameter (,Actual-Parameter)*)	**ActualParameterSequence**
Actual-Parameter	::=	Expression	**ConstActualParameter**
	|	var V-name	**VarActualParameter**

Identifier	::=	Letter (Letter|Digit)*	**Identifier**
Integer-Literal	::=	Digit Digit*	**IntegerLiteral**
Character-Literal	::=	**'** Graphic **'**	**CharLiteral**
Operator	::=	**+** | **-** | ***** | **/** | **<** | **>** | **=** | \ | & | @ | % | ^ | ?	**Operator**

Type-denoter ::= Identifier