# **Grammar**

|  |  |  |
| --- | --- | --- |
| Program | ::= | **ProgramStart** Block **ProgramEnd** |
| Block | ::= | **declare** **start** Declatation **declareEnd** **do** Comand **doEnd** |
| Declaration | ::= | (FunctionDeclaration |InitializationTo0)\* |
| FunctionDeclaration | ::= | FunctionName **(**(Type VarName (,Type VarName)\*)?**)** Block**;** |
| InitializationTo0 (oneDeclaration ) | ::= | Type(**[**(LiteralNumber)?**]**)? VarName (,VarName)\***;** |
| Comand | ::= | (AssignmentOrFunctionCallAlone|LoopBlock|ConditionBlock| | **end ;**| **giveBackWith** Expression **;**| **giveBackWith nothing ;**)\* |
| AssignmentOrFunctionCallAlone | ::= | Assignment| FunctionCallAlone |
| Expression | ::= | (varName**;**|Operation|LiteralNumber|FunctionCall|ArrayEntry) |
| Assignment | ::= | ((VarName|ArrayEntry) **<-** Expression **;**|  Expression **->** (VarName|ArrayEntry)**;**) |
| LoopBlock | ::= | (ForLoop| WhileLoop) |
| ConditionBlock | ::= | (IfStatement|SwitchStatement) |
| FunctionCallAlone | ::= | FunctionName**(**((VarName| LiteralNumber) (,(VarName| LiteralNumber))\*)?**);** |
| FunctionCall | ::= | FunctionName**(**((VarName| LiteralNumber) (,(VarName| LiteralNumber))\*)?**)** |
| Operation | ::= | ((varName| LiteralNumber|Negation) (Operator(varName| LiteralNumber|Negation| FunctionCall))\*)**;** |
| Negation | ::= | **.not** Expression |
| ForLoop | ::= | **for(**Assignment**;** Operation**;** Operator LiteralNumber**) start** Comand **forEnd** |
| WhileLoop | ::= | **while(**Operation**;) start** Comand\* **whileEnd** |
| IfStatement | ::= | **if(**Operation**;) start** Comand\* (**elif (**Operation**;) start** Comand\*)\*(**else** Comand\*)? **fi** |
| SwitchStatement | ::= | **switch (**VarName**):** (**case** Value **start** Comand\* **end**)+ (**default start** Comand\* **end**) **switchEnd** |
| Operator | ::= | **.**(**-**|**\***|**/**|**%**|**and**|**or**|**<**|**<=**|**>**|**>=**|**=**) |
| LiteralNumber | ::= | (**-**)?[**0**-**9**]+ |
| VarName | ::= | ([**a**-**z**]|[**A**-**Z**]|\_) ([**a**-**z**]|[**A**-**Z**]|[**0**-**9**]|\_)\* |
| ArrayEntry | ::= | VarName**[**Expression**;]** |