## Code Specification

|  |  |
| --- | --- |
| **Functions** | **Code Templates** |
| run⟦program⟧ | run⟦program → name:string types:structDefinition\* vars:varDefinition\* builders:functionBuilder\* features:functionDefinition\* runCall:runCall⟧ = |
| metadata⟦program⟧ | metadata⟦program → name:string types:structDefinition\* vars:varDefinition\* builders:functionBuilder\* features:functionDefinition\* runCall:runCall⟧ = |
| run⟦runCall⟧ | run⟦runCall → name:string args:expression\*⟧ = |
| ƒ2⟦structDefinition⟧ | ƒ2⟦structDefinition → name:structType fields:fieldDefinition\*⟧ = |
| ƒ3⟦functionDefinition⟧ | ƒ3⟦functionDefinition → name:string params:varDefinition\* returnType:type? vars:varDefinition\* sentences:sentence\*⟧ = |
| ƒ4⟦fieldDefinition⟧ | ƒ4⟦fieldDefinition → name:string tipo:type⟧ = |
| ƒ5⟦varDefinition⟧ | ƒ5⟦varDefinition → name:string tipo:type⟧ = |
| ƒ6⟦functionBuilder⟧ | ƒ6⟦functionBuilder → name:string⟧ = |
| execute⟦sentence⟧ | execute ⟦functionCallSent:sentence → name:string args:expression\*⟧ = |
| execute ⟦assignment:sentence → left:expression right:expression⟧ = |
| execute ⟦loop:sentence → from:assignment\* until:expression body:sentence\*⟧ = |
| execute ⟦ifElse:sentence → condition:expression trueBlock:sentence\* falseBlock:sentence\*⟧ = |
| execute ⟦read:sentence → input:expression\*⟧ = |
| execute ⟦print:sentence → op:string input:expression\*⟧ =  #LINE (end.line)  value[[expression]]  **OUT**<maplTypeSuffix(expression.type)> |
| execute ⟦return:sentence → value:expression?⟧ = |
| address⟦expression⟧ | address ⟦intConstant:expression → value:string⟧ =  #Error |
| address ⟦realConstant:expression → value:string⟧ =  #Error |
| address ⟦charConstant:expression → value:string⟧ =  #Error |
| address ⟦variable:expression → name:string⟧ = |
| address ⟦castExpr:expression → castType:type value:expression⟧ =   #Error |
| address ⟦arithmeticExpr:expression → op1:expression operator:string op2:expression⟧ =  #Error |
| address ⟦logicalExpr:expression → op1:expression operator:string op2:expression⟧ =  #Error |
| address ⟦comparationExpr:expression → op1:expression operator:string op2:expression⟧ =  #Error |
| address ⟦minusExpr:expression → op:expression⟧ =   #Error |
| address ⟦notExpr:expression → op:expression⟧ =   #Error |
| address ⟦functionCallExpr:expression → name:string args:expression\*⟧ = |
| address ⟦fieldAccess:expression → root:expression field:string⟧ = |
| address ⟦arrayAccess:expression → array:expression index:expression⟧ = |
|  |
| value⟦expression⟧ | value ⟦intConstant:expression → value:string⟧ =   value ⟦realConstant:expression → value:string⟧ =   value ⟦charConstant:expression → value:string⟧ =   value ⟦variable:expression → name:string⟧ =   value ⟦castExpr:expression → castType:type value:expression⟧ =   value ⟦arithmeticExpr:expression → op1:expression operator:string op2:expression⟧ =  value[[left]]  value[[right]]  maplOperator(operator, leftType)  value ⟦logicalExpr:expression → op1:expression operator:string op2:expression⟧ =   value ⟦comparationExpr:expression → op1:expression operator:string op2:expression⟧ =   value ⟦minusExpr:expression → op:expression⟧ =   value ⟦notExpr:expression → op:expression⟧ =   value ⟦functionCallExpr:expression → name:string args:expression\*⟧ =   value ⟦fieldAccess:expression → root:expression field:string⟧ =   value ⟦arrayAccess:expression → array:expression index:expression⟧ = |
| ƒ9⟦type⟧ | ƒ9⟦intType:type → ε⟧ =   ƒ9⟦doubleType:type → ε⟧ =   ƒ9⟦charType:type → ε⟧ =   ƒ9⟦voidType:type → ε⟧ =   ƒ9⟦structType:type → name:string⟧ =   ƒ9⟦arrayType:type → dimension:intConstant tipo:type⟧ = |

|  |  |
| --- | --- |
| Name | Description |
| **maplType**(type) | Returns the MAPL equivalent to the MLang type argument:     intType -> *int*     floatType -> *float* |
| **maplSuffixType**(type) | Returns de MAPL equivalent to the Mlang type argument  intType -> *“i"* floatType -> “f” |
| **maplOperator**(String, Type) | Returns the equivalent MAPL operator  “+” -> “ADD” + <Type>  “-“ -> “SUB” + <Type>  “\*” -> “MUL” + <Type>  “/” -> “DIV” + <Type> |