## Attribute Grammar – *MemoryAllocation* – Ignacio Fernández Suárez (UO294177)

### Attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | Attribute Name | Java Type | Inherited/Synthesized | Description |
| VarDefinition | address | int | Inherited | Indica la posición de memoria en la que esta localizada |
| Field | offset | int | Synthesized | Indica el desplazamiento de un campo |

### Rules

|  |  |  |
| --- | --- | --- |
| Node | Predicates | Semantic Functions |
| program → classDef global? create feature\* runInvocation |  |  |
| classDef → name:string |  |  |
| runInvocation → procedure |  |  |
| readStmt:stmt → expression\* |  |  |
| printStmt:stmt → expression\* format:string |  |  |
| assignStmt:stmt → assignment |  |  |
| ifStmt:stmt → condition:expression ifStmts:stmt\* elseStmts:stmt\* |  |  |
| fromStmt:stmt → declarations:assignment\* condition:expression stmts:stmt\* |  |  |
| procedureStmt:stmt → procedure |  |  |
| returnStmt:stmt → returnInvoc |  |  |
| assignment → left:expression right:expression |  |  |
| intLiteral:expression → value:string |  |  |
| realLiteral:expression → value:string |  |  |
| charLiteral:expression → value:string |  |  |
| variable:expression → name:string |  |  |
| procedureExpression:expression → procedure |  |  |
| arrayExpression:expression → array:expression index:expression |  |  |
| structExpression:expression → struct:expression field:string |  |  |
| minusExpression:expression → expression |  |  |
| notExpression:expression → expression |  |  |
| cast:expression → dataType expression |  |  |
| arithmeticExpression:expression → left:expression operator:string right:expression |  |  |
| comparisonExpression:expression → left:expression operator:string right:expression |  |  |
| logicExpression:expression → left:expression operator:string right:expression |  |  |
| procedure → name:string expression\* |  |  |
| integerType:dataType → ε |  |  |
| doubleType:dataType → ε |  |  |
| characterType:dataType → ε |  |  |
| structType:dataType → name:string |  |  |
| arrayType:dataType → size:string dataType |  |  |
| voidType:dataType → ε |  |  |
| errorType:dataType → ε |  |  |
| create → idents:string\* |  |  |
| feature → name:string params:varDefinition\* dataType? localBlock? doBlock |  | **Calculamos offset de los parámetros**  **y establecemos la dirección de memoria**  int paramOffset = 4  for(int i=params.size()-1; i>=0; i--){  VarDefinition par = params.get(i)  par.address = paramOffset  paramOffset += par.type.memorySize}  **Calculamos offset de variables locales**  **y establecemos la dirección de memoria**  for(VarDefinition vd : feature.localBlock.varDefinitions){  localOffset -= vd.type.memorySize  vd.address = localOffset} |
| returnInvoc → expression? |  |  |
| localBlock → varDefinition\* |  |  |
| doBlock → stmt\* |  |  |
| global → globalTypes? varsTypes? |  |  |
| globalTypes → deftuple\* |  |  |
| varsTypes → varDefinition\* |  | **Calculamos el offset de las variables globales y establecemos la dirección de memoria**  for(varDefinition:varsTypes.varDefinitions)  { varDefinition.address = globalOffset globalOffset += varDefinition.type.  memorySize} |
| deftuple → name:string field\* |  | **Calculamos el offset de los campos**  for(Field field : deftuple.fields){  field.offset = currentFieldOffset  currentFieldOffset += field.type.memorySize} |
| field → name:string type:dataType |  |  |
| varDefinition → name:string type:dataType |  |  |

Operators samples (cut & paste if needed):  
⇒ ⇔ ≠ ∅ ∈ ∉ ∪ ∩ ⊂ ⊄ ∑ ∃ ∀

### Auxiliary Methods

|  |  |  |
| --- | --- | --- |
| Name | Return | Description |
| getSize() | int | Devuelve el tamaño de un determinado tipo |