## Especificación de Código

|  |  |
| --- | --- |
| **Función de Código** | **Plantillas de Código** |
| run : Programa → Instruccion\* | run [[ Programa  →  definiciones:Definicion\* ]] =  define[[definicionesi]] |
|  |  |
| define : Definicion → Instruccion\* | define[[ DefFuncion  →  nombre:String  parametros:DefinicionVariable\*   retorno:Tipo  locales:DefinicionVariable\*  sentencias:Sentencia\* ]] =  ejecuta[[sentenciasi]]  define[[DefVariable 🡪 nombre:String tipo:Tipo]] =  define[[DefStruct 🡪 nombre:String defCampoStruct\*]]= |
|  |  |
| ejecuta : Sentencia → Instruccion\* |  |
|  | ejecuta [[ Asignacion  →  left:Expresion  right:Expresion ]] =  #LINE {end.line}  dirección[[left]]  valor[[right]]  STORE |
|  | ejecuta [[ Print  →  expr:Expresion ]] =  #LINE {end.line}  valor[[expr]]  OUT  ejecuta[[**Condicional** -> expresion sentif:sentencia\* sentelse:sentencia\* ]] =  #LINE {end.line}  valor[[condición]]  JZ else  ejecuta[[sentencia]]  JMP final  else: ejecuta[[sentencia]]  final:  ejecuta[[**Bucle** -> expresion sentencia\*]] =  #LINE {end.line}  valor[[condición]]  JZ final  ejecuta[[sentenciai]]  final:  ejecuta[[**Read** -> expresion]] =  #LINE {end.line}  dirección[[expr]]  IN  STORE  ejecuta[[**Return** -> expresion]] =  #LINE{end.line}  valor[[expr]]  ejecuta[[**InvocaFuncSent** -> nombre:string argumentos:expresion\*]]=  CALL invocaFunc.definicion.nombre  si invocaFuncSent.definicion.tipo ¡= null  POP |
|  |  |
| valor: Expresion → Instruccion\* | valor[[**LiteralInt** -> valor:String]] =  #LINE{end.line} |
|  | PUSH {valor}  valor[[**LiteralReal** -> valor:String]] =  #LINE{end.line}  PUSH {valor}  valor[[**Caracter** -> valor:String]] =  #LINE{end.line}  PUSH{valor}  valor[[**Variable** -> nombre:String]] =  #LINE{end.line}  PUSHA {variable.definicion.direccion}  LOAD  valor[[**AccesoCampoStruct** -> left:expresion right:String]] =  #LINE{end.line}  PUSHA {left[nombre==”right”].dir}  LOAD  valor[[**AccesoArray** -> variable:expresion acceso:expresión]] =  #LINE{end.line}  dirección[[variable]]  valor[[acceso]]  PUSH {variable.tipo.size}  MUL  ADD  LOAD  valor[[**Cast** -> tipo:tipo expresion:expresión]] =  #LINE{end.line}  dirección[expresión]  LOAD  {expresión.tipo.sufijo}2{tipo.sufijo}  Valor[[**Negacion** -> expresion:expresión]] =  #LINE{end.line}  valor[[expresión]]  JZ esCero:  PUSH 0  JMP final  esCero: PUSH 1  final:    valor[[**ExpresionBinaria** -> left:expresión operador:string right:expresión]] =  #LINE{end.line}  valor[[left]]  valor[[right]]  si operador == ‘+’  ADD  si operador == ‘-‘  SUB  si operador == ‘/’  DIV  si operador == ‘\*’  MUL  valor[[**InvocaFunc** 🡪 nombre:string argumentos:expresion\*]] =  CALL invocaFunc.definicion.nombre |
| dirección: expresión -> Intruccion\* | dirección[**Variable** -> nombre:String]] =  #LINE{end.line}  PUSHA {variable.definicion.dir} |

NOTA: Lo que está en naranja es temporal para esta clase y habrá que ampliarlo en la siguiente.