## Especificación de Código

|  |  |
| --- | --- |
| **Función de Código** | **Plantillas de Código** |
| run[[Programa]] | run⟦Programa  →  definiciones:Definicion\*⟧ =  #SOURCE {file}  define⟦definicionesi⟧ |
|  |  |
| define[[Definicion]] | define⟦DefinicionFuncion  →  nombre:String  params:DefinicionVariable\*   retorno:Tipo  locales:DefinicionVariable\*  sentencias:Sentencia\*⟧ =  # LINE {end.line}  # FUNC {nombre}  Int tamañoLocales  Si locales.isEmpty()  tamañoLocales = 0  Else  tamañoLocales = -locales.get(locales.size()-1).direccion  ENTER {tamañoLocales}  Int tamañoParams = params.stream()  .mapToInt(param -> param.tipo.tamaño)  .sum()  Int tamañoReturn = tipo.tamaño  ejecuta⟦sentenciasi⟧ (tamañoReturn, tamañoLocales, tamañoParams) |
|  | define⟦DefinicionVariable  →  nombre:String tipo:Tipo⟧ = |
|  | define⟦DefinicionStruct  →  nombre:String campos:Campo\*⟧ = |
|  | define⟦**Campo**  →  nombre:String tipo:Tipo⟧ = |
|  |  |
| ejecuta[[Sentencia]] | ejecuta⟦Print  →  expr:Expresion *tipo\_print*:String⟧ =  #LINE {end.line}  OUT <expr.tipo> valor[[expr]]  si tipo\_print == “printsp”  OUTB { }  si tipo\_print == “println”  OUTB {\n} |
|  | ejecuta[[**Read** → expresion:Expresion]] =  #LINE {end.line}  IN<expresion.tipo> |
|  | ejecuta⟦Asignacion  →  izquierda:Expresion  derecha:Expresion⟧ =  #LINE {end.line}  dirección[[izquierda]]  valor[[derecha]]  STORE <izquierda.tipo> |
|  | ejecuta[[**If** → *condicion*:Expresion *verdadero*:Sentencia\* *falso*:Sentencia\*]] =  #LINE {end.line} |
|  | ejecuta[[**While** → *condicion*:Expresion *sentencias*:Sentencia\*]] =  #LINE {end.line} |
|  | ejecuta[[**Invocacion** → *nombre*:String *params*:Expresion\*]] =  #LINE {end.line}  valor[[paramsi]] CALL {nombre}  Si invocacion.definicion.tipo != TipoVoid  POP <invocacion.definicion.tipo> |
|  | ejecuta[[**Return**→ *expresiones*:Expresion\*]]  (tamañoReturn, tamañoLocales, tamañoParams) =  #LINE {end.line}  Si expresiones.size != 0  value[[expresiones.get(0)]]  RET {tamañoReturn, tamañoLocales, tamañoParams} |
|  |  |
| valor[[Expresion]] | valor[[**ConstanteEntero**→  *valor*:int ]] =  PUSH {valor} |
|  | valor[[**ConstanteReal**→  *valor*:double ]] =  PUSHF {valor} |
|  | valor[[**ConstanteChar**→  *valor*:String ]] =  PUSHB {valor} |
|  | valor[[**Variable** →  *nombre*:String ]] =  address[[variable]]  LOAD<variable.tipo> |
|  | valor[[**ExpresionAritmetica**→  *izq*:Expresion  *operador*:String  *der*:Expresion ]] =  value[[izq]]  value[[der]]  si operador == “+”  ADD<arithmeticExpression.tipo>  si operador == “-”  SUB<arithmeticExpression.tipo>  si operador == “\*”  MUL<arithmeticExpression.tipo>  si operador == “/”  DIV<arithmeticExpression.tipo>  si operador == “%”  MOD |
|  | valor[[**ExpresionLogica**→  *izq*:Expresion  *operador*:String  *der*:Expresion ]] =  value[[izq]]  value[[der]]  si operador == “&&”  AND  si operador == “||”  OR |
|  | valor[[**Comparacion**→  *izq*:Expresion  *operador*:String  *der*:Expresion ]] =  value[[izq]]  value[[der]]  si operador == “<”  LT  si operador == “>”  GT  si operador == “≥”  GT  value[[izq]]  value[[der]]  EQ  OR  si operador == “≤”  LT  value[[izq]]  value[[der]]  EQ  OR |
|  | valor[[**ExpresionUnaria**→  *expresion*:Expresion  *operador*:String ]] =  value[[expresion]]  NOT |
|  | valor[[**Conversion**→  *nuevoTipo*:Tipo  *expresion*:Expresion ]] =  value[[expresion]]  si expresion.tipo == TipoEntero  si nuevoTipo == TipoReal  I2F  si nuevoTipo == TipoChar  I2B  si expresion.tipo == TipoReal  si nuevoTipo == TipoEntero  F2I  si nuevoTipo == TipoChar  F2I  I2B  si expresion.tipo == TipoChar  si nuevoTipo == TipoEntero  B2I  si nuevoTipo == TipoReal  B2I  I2F |
|  | valor[[**InvocacionExpresion**→  *nombre*:String  *params*:Expresion\* ]] =  valor[[paramsi]] CALL {nombre} |
|  | valor[[**AccesoArray**→  *array*:Expresion  *indice*:Expresion ]] =  dirección[[accesoArray]]  LOAD<accesoArray.tipo> |
|  | valor[[**AccesoCampo**→  *struct*:Expresion  *campo*:Expresion ]] =  dirección[[accesoCampo]]  LOAD<accesoCampo.tipo> |