## Especificación de Código

|  |  |
| --- | --- |
| **Función de Código** | **Plantillas de Código** |
|  |  |
| run[[program]] | run[[program → bloque:bloque\* ]] =  #SOURCE {file}  CALL main  HALT  define[[bloquei]] |
|  |  |
| define[[bloque]] | define [[funcion → nombre:String parametros:parametro\* retorno:tipo\* locales:definicion\_variable\_local\* sentencias:sentencia\* ]] =  {nombre}:  ENTER {∑localesi.tipo.size}  ejecuta[[sentenciasi]]  si retorno == VOID  RET 0, {∑localesi.tipo.size}, {∑parametrosi.tipo.size} |
|  |  |
| ejecuta[[sentencia]] | ejecuta [[sentencia\_asignacion → izquierda:expr derecha:expr ]] =  #LINE {end.line}  address[[izquierda]]  value[[derecha]]  STORE<izquierda.tipo> |
|  | ejecuta [[sentencia\_print → expresiones:expr ]] =  #LINE {end.line}  value[[expresiones]]  OUT< expresiones.tipo> |
|  | ejecuta [[sentencia\_read → expresiones:expr ]] =  #LINE {end.line}  value[[expresiones]]  IN<expresiones.tipo> |
|  | ejecuta [[sentencia\_if → condicion:expr sentencias:sentencia\* sino:sentencia\* ]] =  {contadorIF = ++contadorGeneralIF}  #LINE {end.line}  if{contadorIf}:  valor[[condicion]]  jz else{contadorIf}  ejecuta[[sentenciasi]]  jmp finIf{contadorIf}  else{contadorIf}:  ejecuta[[sinoi]]  finIf{contadorIf}: |
|  | ejecuta [[sentencia\_while → condicion:expr sentencias:sentencia\* ]] =  {contadorWhile = ++contadorGeneralWhile}  #LINE {end.line}  while{contadorWhile}:  valor[[condicion]]  jz finWhile{contadorWhile}  ejecuta[[sentenciasi]]  jmp while{contadorWhile}  finWhile{contadorWhile}: |
|  | ejecuta [[sentencia\_llamada\_funcion → nombre:String parametros:expr\* ]] =  #LINE {end.line}  valor[[parametrosi]]  CALL {nombre}  si sentencia\_llamada\_funcion.definicion.retorno != tipoVoid  POP< sentencia\_llamada\_funcion.definición.retorno> |
|  | ejecuta [[sentencia\_return → expresion:expr ]] =  #LINE {end.line}  si expresion ≠ null  valor[[expr]]  RET {sentencia\_return.funcion.retorno.size},  {∑sentencia\_return.funcion.localesi.tipo.size},  {∑sentencia\_return.funcion.parametrosi.tipo.size} |
|  |  |
| valor[[expr]] | valor [[expr\_int → string:String ]] =  PUSHI {value} |
|  | valor [[expr\_real → string:String ]] =  PUSHF {value} |
|  | valor [[expr\_char → string:String ]] =  PUSHB {value} |
|  | valor [[expr\_ident → string:String ]] =  address [[expr\_ident]]  LOAD< expr\_ident.type> |
|  | valor [[expr\_binaria → izquierda:expr operador:operador derecha:expr ]] =  value[[izquierda]]  value[[derecha]]  {operador.instruccion} |
|  | valor [[expr\_vector → fuera:expr dentro:expr ]] =  address[[fuera]]  value[[dentro]]  PUSHA {tipo.size}  MUL  ADD |
|  | valor [[expr\_punto → izquierda:expr derecha:expr ]] =  address[[izquierda]]  address[[derecha]]  ADD |
|  | valor [[expr\_parentesis → expr:expr ]] =  value[[expr]] |
|  | valor[[expr\_cast → tipo\_convertido:tipo expr:expr ]] =  value[[expr]]  {expr.tipo}2{tipo\_convertido} |
|  | valor[[expr\_llamada\_funcion → nombre:String parametros:expr\* ]] =  valor[[parametrosi]]  CALL {nombre} |
|  |  |
| address[[expr]] | address[[variable → name:String ]] =  PUSHA {variable.definition.address} |

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