## Especificación de Código

|  |  |
| --- | --- |
| **Función de Código** | **Plantillas de Código** |
| run : Programa → Instruccion\* | run [[ Programa  →  definiciones:Definicion\* ]] =  #SOURCE {file}  define[[definicionesi]] |
|  |  |
| define : Definicion → Instruccion\* | define[[ DefinicionFuncion  →  nombre:String  parametros:DefinicionVariable\*   retorno:Tipo  locales:DefinicionVariable\*  sentencias:Sentencia\* ]] =  ejecuta[[sentenciasi]] |
|  |  |
| ejecuta : Sentencia → Instruccion\* | // Por ahora, hacer solo Asignación y Print |
|  | ejecuta [[ Asignacion  →  left:Expresion  right:Expresion ]] =  #LINE {end.line}  /\* Seguir \*/ |
|  | ejecuta [[ Print  →  expr:Expresion ]] =  #LINE {end.line}  /\* Seguir \*/ |
|  |  |
| valor: Expresion → Instruccion\* | // Hacer todas las expresiones ***excepto invocación de funciones*** |
|  |  |

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

|  |  |  |
| --- | --- | --- |
| **Función** | **Plantillas de Código** | |
| run[[**programa**]] | run [[ Programa  →  definicion:Definicion\* ]] =  #SOURCE {file}  define[[definicioni]] | |
|  |  | |
| define[[**definicion**]] | define[[**defVariable**  →  *ident*:String  *tipo*:tipo ]] =  #LINE {end.line}  si defVariable.ambito == AmbitoVariable.GLOBAL  #GLOBAL {ident, tipo.getMAPLName} | |
|  | define[[**defStruct**  →  *ident*:String  *cuerpostruct*:cuerpoStruct\* ]] =  #TYPE {ident} ‘:{‘  defineCuerpoStruct[[cuerpoStructi]]  ‘}’ | |
|  | define[[**defFuncion**  →  *ident*:String  *parametrosFuncion*:defVariable\*       *tipo*:tipo  *definiciones*:defVariable\*  *sentencias*:sentencia\* ]] =  #FUNC {ident}  #PARAM { parametrosFuncioni.ident } ‘:’ { parametrosFuncioni.tipo}  #RET { tipo }  #LOCAL { definicionesi.ident } ‘:’ { definicionesi.tipo}  {ident}:  ENTER {∑definicionesi.tipo.size}  ejecuta[[sentenciasi]]  si tipo == VOID:  RET 0, {∑definicionesi.tipo.size}, {∑parametrosFuncioni.tipo.size} | |
| defineCuerpoStruct[[**cuerpoStruct**]] | defineCuerpoStruct[[**cuerpoStruct**  →  *ident*:String  *tipo*:tipo ]] =  {ident} ‘:’ {tipo} | |
|  |  | |
| ejecuta[[**sentencia**]] | ejecuta[[**return**  →  *expresion*:expresion ]] =  #LINE {end.der}  Si expresion ≠ null:  valor[[expresion]]  RET {return.funcionEnLaQueEstoy.tipo.getSize},  {∑return.funcionEnLaQueEstoy.definicionesi.tipo.size},  {∑return.funcionEnLaQueEstoy.parametrosFuncioni.tipo.size} | |
|  | ejecuta[[**asignacion**  →  *izq*:expresion  *der*:expresion ]] =  #LINE {end.line}  direccion[[izq]]  valor[[der]]  STORE<izq.tipo> | |
|  | ejecuta[[**printsp**  →  *expresion*:expresion ]] =    #LINE {end.line}  valor [[expresion]]  OUT  PUSHB 32  OUTB | |
|  | ejecuta[[**print**  →  *expresion*:expresion ]] =  #LINE {end.line}  valor[[expresion]]  OUT | |
|  | ejecuta[[**println**  →  *expresion*:expresion ]] =  #LINE {end.line}  valor [[expresion]]  OUT  PUSHB 10  OUTB | |
|  | ejecuta[[**read**  →  *expresion*:expresion ]] =    #LINE {end.line}  direccion[[expresion]]  IN  STORE | |
|  | ejecuta[[**bucleWhile**  →  *condicion*:expresion  *cuerpo*:sentencia\* ]] =  bucleWhilex:  valor[[condicion]]  jz finBucleWhilex  ejecuta[[cuerpo]]  jmp bucleWhilex    finBucleWhilex: | |
|  | ejecuta[[**sentenciaCondicional**  →  *condicion*:expresion  *cuerpoIf*:sentencia\*  *cuerpoElse*:sentencia\* ]] =  #LINE{end.line}  sentenciaCondicionalx:  valor[[condicion]]  jz elsex  ejecuta[[cuerpoIf]]  jmp finSentenciaCondicionalx  elsex:  ejecuta[[cuerpoElse]]  finSentenciaCondicionalx: | |
|  | ejecuta[[**invocacionFuncionSentencia**  →  *id*:String  *parametros*:expresion\* ]] =    #LINE{end.line}  valor[[parametros]]  CALL {id}  si invocacionFuncionSentencia.funcionEnLaQueEstoy.tipo ≠ VOID  POP<invocacionFuncionSentencia.funcionEnLaQueEstoy.tipo> | |
| valor[[**expresion**]] | valor[[**accesoStruct**  →  *expresion*:expresion  *ident*:String ]] =  #LINE {end.line}  direccion[[accesoStruct]]  LOAD<accesoStruct.tipo> | |
|  | valor[[**accesoArray**  →  *ident*:expresion  *posicion*:expresion ]] =  #LINE {end.line}  direccion[[accesoArray]]  LOAD<accesoArray.tipo> | |
|  | valor[[**variable**  →  *ident*:String ]] =  #LINE {end.line}  direccion[[variable]]  LOAD<variable.definicion.tipo> | |
|  | valor[[**literalInt**  →  *value*:String ]] =  #LINE {end.line}  PUSH {value} | |
|  | valor[[**literalReal**  →  *value*:String ]] =  #LINE {end.line}  PUSHF {value} | |
|  | valor[[**literalChar**  →  *value*:String ]] =  #LINE {end.line}  PUSHB {value} | |
|  | valor[[**cast**  →  *tipo*:tipo  *expresion*:expresion ]] =  #LINE {end.line}  valor[[expresion]]  <expresion.tipo>2<tipo> | |
|  | valor[[**negacion**  →  *expresion*:expresion ]] =  #LINE {end.line}  valor[[expresion]]  NOT | |
|  | valor[[**expresionAritmetica**  →  *izquierda*:expresion  *operador*:String  *derecha*:expresion ]] =  #LINE {end.line}  valor[[izquierda]]  valor[[derecha]]  <operador> | |
|  | valor[[**expresionBinaria**  →  *izquierda*:expresion  *operador*:String  *derecha*:expresion ]] =  #LINE {end.line}  valor[[izquierda]]  valor[[derecha]]  <operador><derecha.tipo> | |
|  | valor[[**expresionLogica**  →  *izquierda*:expresion  *operador*:String  *derecha*:expresion ]] =  #LINE {end.line}  valor[[izquierda]]  valor[[derecha]]  <operador> | |
|  | valor[[**invocacionFuncionExpresion**  →  *id*:String  *parametros*:expresion\* ]] =  #LINE {end.line}  valor[[parametrosi]]  CALL{id} | |
| direccion[[expresion]] |  | direccion[[**accesoStruct**  →  *expresion*:expresion  *ident*:String ]] =  #LINE {end.line}  direccion[[expresion]]  PUSH {expresion.definicion.cuerpoStruct[ident].address}  ADD |
|  |  | direccion [[**accesoArray**  →  *ident*:expresion  *posicion*:expresion ]] =    #LINE {end.line}  direccion[[ident]]  valor[[posicion]]  PUSH ident.tipo.tipo.size  MUL  ADD |
|  |  | direccion [[**variable**  →  *ident*:String ]] =  #LINE {end.line}  si variable.definicion.ambito == AmbitoVariable.GLOBAL  PUSHA {variable.definicion.address}  sino  PUSH BP  PUSH {variable.definicion. address }  ADD |
|  |  | direccion [[**literalInt**  →  *value*:String ]] =  error |
|  |  | direccion [[**literalReal**  →  *value*:String ]] =  error |
|  |  | direccion [[**literalChar**  →  *value*:String ]] =  error |
|  |  | direccion [[**cast**  →  *tipo*:tipo  *expresion*:expresion ]] =    error |
|  |  | direccion [[**negacion**  →  *expresion*:expresion ]] =    error |
|  |  | direccion [[**expresionAritmetica**  →  *izquierda*:expresion  *operador*:String  *derecha*:expresion ]] =    error |
|  |  | direccion [[**expresionBinaria**  →  *izquierda*:expresion  *operador*:String  *derecha*:expresion ]] =    error |
|  |  | direccion [[**expresionLogica**  →  *izquierda*:expresion  *operador*:String  *derecha*:expresion ]] =    error |
|  |  | direccion [[**invocacionFuncionExpresion**  →  *id*:String  *parametros*:expresion\* ]] = |

error