## Especificación de Código *(Para MAPL)*

|  |  |
| --- | --- |
| **Función** | **Plantillas de Código** |
| run[[programa]] | run[[programa  →  definicion:definicion\* ]] =  CALL main  HALT  define[[definicioni]] |
|  |  |
|  |  |
| defineCuerpo[[cuerpo]] | defineCuerpo [[cuerpo  →  defvariable:defVariable\*  sentencia:sentencia\* ]] =  ENTER { ∑ defVariable.tipo.size}  ejecuta[[ sentenciai ]] |
|  |  |
|  |  |
| define[[definicion]] | define[[defVariable  →  nombre:String  tipo:tipo  ambito:int ]] =  *si { ámbito } == GLOBAL {*  *‘#GLOBAL’ { nombre } ‘:’ { tipo }*  *}* |
| define [[defFuncion  →  nombre:String  parametros:defVariable\*       retorno:retorno  cuerpo:cuerpo ]] =  *‘#FUNC’ { nombre }*  *‘#PARAM’ { parametrosi.nombre } ‘:’ { parametrosi.tipo}*  *‘#RET’ { retorno.tipo }*  *‘#LOCAL’ { cuerpo.defVariablei.nombre } ‘:’ {cuerpo.defVariablei.tipo}*  { nombre }:  defineCuerpo[[cuerpo]]  si retorno == VOID  RET 0, {∑ cuerpo.defVariablei.tipo.size }, {∑ parametrosi.tipo.size } |
| define [[defEstructura  →  nombre:String  defcampo:defCampo\* ]] =  *‘#TYPE’ { nombre } ‘:{‘*  *{ defCampoi.nombre } ‘:’ { defCampoi.tipo }*  *‘}’* |
|  |  |
| ejecuta[[sentencia]] | ejecuta [[while  →  condicion:expresion  cierto:sentencia\* ]] =  whilex:  valor[[condicion]]  jz finWhilex  ejecuta[[ciertoi]]  jmp whilex  finWhilex: |
| ejecuta [[ifElse  →  condicion:expresion  cierto:sentencia\*  falso:sentencia\* ]] =  valor[[condicion]]  jz elsex  ejecuta[[ciertoi]]  jmp finIfx  elsex:  ejecuta[[falsoi]]  finIfx: |
| ejecuta [[return  →  expresion:expresion ]] =  valor [[expresion]]  RET expresion.tipo.size, {∑ return.definicion.cuerpo.defVariablei.tipo.size }, {∑ return.definicion.parametrosi.tipo.size } |
| ejecuta [[print  →  expresion:expresion ]] =  valor [[expresion]]  OUT |
| ejecuta [[println  →  expresion:expresion ]] =  valor [[expresion]]  OUT  PUSHB 10  OUTB |
| ejecuta [[printsp  →  expresion:expresion ]] =  valor [[expresion]]  OUT  PUSHB 32  OUTB |
| ejecuta [[read  →  expresion:expresion ]] =  dirección [[expresion]]  IN  STORE |
| ejecuta [[asigna  →  left:expresion  right:expresion ]] =  direccion [[left]]  valor [[right]]  STORE |
| ejecuta[[llamadaFuncionSentencia  →  nombre:String  argumentos:expresion\* ]] =  valor[[argumentosi]]  CALL {nombre}  Si llamadaFuncionSentencia.definicion.retorno.tipo ≠ VOID  POP |
|  |  |
| valor[[expresion]] | valor[[expresionAritmética  →  left:expresion  operador:String  right:expresion ]] =  valor[[left]]  valor [[right]]  <operador> |
| valor[[expresionBooleana  →  left:expresion  operador:String  right:expresion ]] =  valor[[left]]  valor [[right]]  <operador> |
| valor[[invocacion  →  nombre:String  argumentos:expresion\* ]] =  valor[[argumentosi]]  CALL {nombre} |
| valor[[variable  →  nombre:String ]] =  direccion[[nombre]]  LOAD<variable.tipo> |
| valor [[literalInt  →  valor:String ]] =  PUSH { valor } |
| valor [[literalReal  →  valor:String ]] =  PUSHF { valor } |
| valor [[literalChar  →  valor:String ]] =  PUSHB { valor } |
| valor [[varArray  →  identificacion:expresion  posicion:expresion ]] =  dirección[[varArray]]  LOAD<varArray.tipo> |
| valor [[cast  →  tipo:tipo  expresion:expresion ]] =  valor[[expresion]]  <expresión.tipo>2<tipo> |
| valor [[navega  →  expresion:expresion  nombre:String ]] =  dirección[[navega]]  LOAD<navega.tipo> |
| valor [[conParentesis  →  expresion:expresion ]] =  valor[[expresión]] |
| valor [[negacion  →  expresion:expresion ]] =  valor[[expresion]]  NOT |
| direccion[[expresion]] | dirección[[expresionAritmética  →  left:expresion  operador:String  right:expresion ]] = |
| dirección [[expresionBooleana  →  left:expresion  operador:String  right:expresion ]] = |
| dirección [[invocacion  →  nombre:String  argumentos:expresion\* ]] = |
| dirección [[variable  →  nombre:String ]] =  si variable.definicion.ambito == GLOBAL  PUSHA {variable.definicion.direccion}  sino  PUSH BP  PUSH {variable.definicion.direccion}  ADD |
| dirección [[literalInt  →  valor:String ]] = |
| dirección [[literalReal  →  valor:String ]] = |
| dirección [[literalChar  →  valor:String ]] = |
| dirección [[varArray  →  identificacion:expresion  posicion:expresion ]] =  dirección[[identificación]]  valor[[posición]]  PUSH identificación.tipo.tipo.size  MUL  ADD |
| dirección [[cast  →  tipo:tipo  expresion:expresion ]] = |
| dirección [[navega  →  expresion:expresion  nombre:String ]] =  dirección[[expresión]]  PUSH { expresión.definición.campos[nombre].dirección }  ADD |
| dirección [[conParentesis  →  expresion:expresion ]] =  dirección[[expresión]] |
|  | dirección [[negacion  →  expresion:expresion ]] = |
|  |  |