## Especificación de Código

|  |  |
| --- | --- |
| **Función de Código** | **Plantillas de Código** |
| run : Programa → Instruccion\* | run [[ Programa  →  definiciones:Definicion\* ]] =  CALL main  HALT  define[[definicionesi]] |
|  |  |
| define : Definicion → Instruccion\* | define[[ DefinicionFuncion  →  nombre:String  parametros:DefinicionVariable\*   retorno:Tipo  locales:DefinicionVariable\*  sentencias:Sentencia\* ]] =  {nombre}:  ENTER {Σlocalesi.tipo.memSize}  ejecuta[[sentenciasi]]  si tipo == null  RET 0, {Σlocalesi.tipo.memSize}, {Σparametrosi.tipo.memSize} |
|  |  |
| ejecuta : Sentencia → Instruccion\* |  |
|  | ejecuta [[ Asignacion  →  left:Expresion  right:Expresion ]] =  #LINE {end.line}  dirección[[left]]  Valor[[right]]  STORE |
|  | ejecuta [[ Write  →  expr:Expresion ]] =  #LINE {end.line}  valor[[expr]]  OUT  ejecuta[[return  →  expr:expr ]] =  si expr == null  RET 0, {Σlocalesi.tipo.memSize}, {Σparametrosi.tipo.memSize}  Sino  RET {expr.tipo.menSize}, {Σlocalesi.tipo.memSize}, {Σparametrosi.tipo.memSize}  ejecuta[[invocacion  →  nombre:String  expr:expr\*  ambito:String ]] =  CALL {nombre}  Si ámbito != llamada && def.tipo != null  Pop  Ejecuta[[if  →  expr:expr  verdadero:sent\_func\*  falso:sent\_func\* ]] =  Valor[[expr]]  JZ else{n}  Ejecuta[[verdaderoi]]  JMP finif{n}  Else{n}:  Ejecuta[[falsoi]]  Finif{n}:  Ejecuta[[while  →  expr:expr  sent\_func:sent\_func\* ]] =  While{n}:  Valor[[expr]]  JZ fin-while-{n}  Ejecuta[[sent\_funci]]  JMP while{n}  Fin-while{n}  Ejecuta[[read  →  expr:expr ]] =  Dirección[[expr]]  IN  STORE |
|  |  |
| valor: Expresion → Instruccion\* | Valor[[lintchar  →  string:String ]] =  PUSH {string}  Valor[[lintreal   →  string:String ]] =  PUSHF {string}  Valor[[lintent   →  string:String ]] =  PUSHB {string}  Valor[[var  →  string:String ]] =  Dirección[[var]]  LOAD  Valor[[expresionNumerica  →  izq:expr  string:String  der:expr ]] =  Valor[[izq]]  Valor[[der]]  Si string==’+’  ADD  Sino si string ==’-’  SUB  Sino si string ==’\*’  MUL  Sino si string ==’/’  DIV  Sino si string ==’<’  LT  Sino si string ==’>’  GT  Sino si string ==’==’  EQ  Sino si string ==’!=’  NE  Sino si string ==’>=’  GE  Sino si string ==’<=’  LE  Valor[[expresionLogica  →  izq:expr  string:String  der:expr ]] =  Valor[[izq]]  Valor[[der]]  Si string==’&&’  AND  Sino si string ==’||’  OR  Valor[[**operacionUnaria**  →  *string*:String  *der*:expr]]=  Si string == ‘!’  Valor[[der]]  NOT  Valor[[**cast**  →  *tipo*:tipo  *expr*:expr]] =  Valor[[expr]]  Si tipo == intType && expr.tipo == charType  b2i  sino si tipo == intType && expr.tipo == realType  f2i  sino si tipo == charType && expr.tipo == intType  i2b  sino si tipo == realType && expr.tipo == intType  i2f  valor[[**accesoArray**  →  *izq*:expr  *der*:expr ]]=  dirección[[izq]]  PUSH {izq.tipo.tipo.menSize}  Valor[[der]]  MUL  ADD  LOAD  valor[[**accesoStruct**  →  *struct*:expr  *string*:String]]=  dirección[[struct]]  PUSH {struct.tipo.definicion.defVar[string].direccion}  ADD  LOAD |
| dirección: **Expresion** -> Instrucción\* | dirección[[var  →  string:String ]] =  si def.ambito == var  PUSHA BP  PUSH {def.direccion}  Sino si def.ambito == param  PUSHA BP  PUSH {def.parametro.direccion}  ADD  Sino  PUSHA {def.direccion}  direccion[[**accesoStruct**  →  *struct*:expr  *string*:String]]=  dirección[[struct]]  PUSH {struct.tipo.definicion.defVar[string].direccion}  ADD  direccion[[**accesoArray**  →  *izq*:expr  *der*:expr ]]=  dirección[[izq]]  PUSH {izq.tipo.tipo.menSize}  Valor[[der]]  MUL  ADD |
|  |  |