Plantillas de código

|  |  |
| --- | --- |
| Función de código | Plantilla de código |
| run: Programa -> Instrucción\* | run[[Programa -> definiciones:Definicion\*]]:  #SOURCE ficheroEntrada  CALL main  HALT  define[[definicionesi]] |
| define: Definicion -> Instrucción\* | define[[DefinicionVariable -> nombre:String tipo:Tipo]]:  #VAR <nombre>:<tipo.nombreMAPL>  define[[DefinicionFuncion -> nombre:String retorno:Tipo parametros:DefinicionVariable\* definicionesVariable:DefinicionVariable\* sentencias:Sentencia\*]]:  #FUNC <nombre>  #PARAM <parametrosi.nombre>:<parametrosi.tipo.nombreMAPL>  #LOCAL <definicionesVariablei.nombre>:< definicionesVariablei.tipo.nombreMAPL>  #RET <retorno.nombreMAPL>  <nombre>:  enter <ΣdefinicionesVariablei.tipo.size>  ejecuta[[sentenciasi]]  SI retorno == NULL  RET 0,<ΣdefinicionesVariablei.tipo.size>,<Σlocalesi.tipo.size>  define[[DefinicionStruct -> nombre:String listaCampos:Campo\* primitivo:boolean]]:  #TYPE <nombre>:{  define[[listaCamposi]]  }  define[[Campo -> nombre:String tipo:Tipo]]:  <nombre>:<tipo.nombreMAPL> |
| ejecuta: Sentencia -> Instrucción\* | ejecuta[[Asignacion -> izquierda:Expresion derecha:Expresion]]:  #Line  direccion[[izquierda]]  valor[[derecha]]  STORE<izquierda.tipo.sufijo>  ejecuta[[IF -> expresion:Expresion sentenciasIF:Sentencia\* sentenciasElse:Sentencia\*]]:  #Line  valor[[expresion]]  JZ Else<n>  ejecuta[[sentenciasIFi]]  JMP finIf<n>  else<n>:  ejecuta[[sentenciasElsei]]  finIf<n>:  ejecuta[[InvocacionProcedimiento -> nombre:String expresiones:Expresion\*]]:  #Line  valor[[expresionesi]]  CALL <nombre>  ejecuta[[Print -> expresion:Expresion\*]]:  #Line  valor[[expresion]]  OUT<expresion.tipo.sufijo>  ejecuta[[Read -> expresion:Expresion\*]]:  #Line  direccion[[expresion]]  IN<expresion.tipo.sufijo>  STORE<expresion.tipo.sufijo>  ejecuta[[Return -> expresion:Expresion\*]]:  #Line  SI expresion != NULL  valor[[expresion]]  RET expresion.tipo.size, <Σdefinicion.definicionesVariablei.tipo.size>, <Σdefinicion.localesi.tipo.size>  ELSE  RET 0, <Σdefinicion.definicionesVariablei.tipo.size>, <Σdefinicion.localesi.tipo.size>  ejecuta[[While -> expresion:Expresion\* sentencias:Sentencia\*]]:  #Line  inicioWhile<n>:  valor[[expresion]]  JZ finWhile<n>:  ejecuta[[sentenciasi]]  JMP inicioWhile<n>:  finWhile<n>: |
| valor: Expresion-> Instrucción\* | valor[[AccesoArray -> izquierda:Expresion derecha:Expresion]]:  dirección[[AccesoArray]]  load<izquierda.tipo.tipo.sufijo>  valor[[AccesoCampo -> izquierda:Expresion derecha:Expresion]]:  dirección[[AccesoCampo]]  load  valor[[Aritmetica -> op1:Expresion op2:Expresion operador:String]]:  valor[[op1]]  valor[[op2]]  SI operador == ‘+’  ADD<tipo.sufijo>  SI operador == ‘-’  SUB<tipo.sufijo>  SI operador == ‘\*’  MUL<tipo.sufijo>  SI operador == ‘/’  DIV<tipo.sufijo>  SI operador == ‘+’  MOD<tipo.sufijo>  valor[[Cast -> tipoCast:Tipo expresion:Expresion]]:  valor[[expresion]]  cast(expresión.tipo, tipoCast)  valor[[Comparacion -> operando1:Expresion operando2:Expresion operador:String]]:  valor[[op1]]  valor[[op2]]  SI operador == ‘>’  GT<tipo.sufijo>  SI operador == ‘<’  LT<tipo.sufijo>  SI operador == ‘>=’  GE<tipo.sufijo>  SI operador == ‘<=’  LE<tipo.sufijo>  SI operador == ‘==’  EQ<tipo.sufijo>  SI operador == ‘!=’  NE<tipo.sufijo>  valor[[InvocacionFuncion-> identificador:String listaExpresiones:Expresion\*]]:  valor[[expresionesi]]  CALL <nombre>  valor[[LiteralCaracter-> caracter:Character]]:  PUSH<tipo.sufijo> <caracter>  valor[[LiteralEntero-> valor:Int ]]:  PUSH<tipo.sufijo> <valor>  valor[[LiteralReal-> valor:Double]]:  PUSH<tipo.sufijo> <valor>  valor[[Logica -> operando1:Expresion operando2:Expresion operador:String]]:  valor[[operando1]]  valor[[operando2]]  SI operador == ‘and’  AND  SI operador == ‘or’  OR  valor[[Negacion -> expresion:Expresion]]:  valor[[expresion]]  NOT  valor[[Variable -> nombre:String definicion:Definicion]]:  direccion[[Variable]]  LOAD<tipo.sufijo> |
| direccion: Expresion-> Instrucción\* | dirección[[AccesoArray -> izquierda:Expresion derecha:Expresion]]:  direccion[[izquierda]]  valor[[derecha]]  PUSH <izquierda.tipo.tipo.size>  MUL  ADD  direccion[[AccesoCampo -> izquierda:Expresion derecha:Expresion]]:  dirección[[izquierda]]  PUSH <izquierda.tipo.offsetCampo(derecha)>  ADD  direccion[[Variable -> nombre:String definicion:Definicion]]:  SI definicion.ambito == ‘GLOBAL’  PUSHA <definicion.direccion>  SI NO  PUSHA BP  PUSH <definicion.direccion>  ADD |