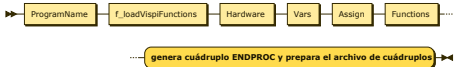


Program:



Program ::= ProgramName f_loadVispiFunctions Hardware Vars Assign Functions 'genera cuádruplo ENDPROC y prepara el archivo de cuádruplos'

no references

ProgramName:



ProgramName ::= 'program' id '\\n' 'genera el GOTO main y guarda el nombre del programa'

referenced by:

- [Program](#)

Hardware:

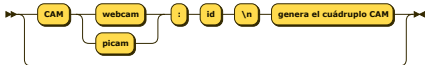


Hardware ::= CamDeclaration InputsDeclaration OutputsDeclaration PwmDeclaration

referenced by:

- [Program](#)

CamDeclaration:



CamDeclaration ::= ('CAM' ('webcam' | 'picam') ':' id '\\n' 'genera el cuádruplo CAM')?

referenced by:

- [Hardware](#)

InputsDeclaration:

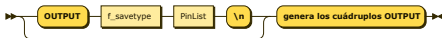


InputsDeclaration ::= ('INPUT' f_savetype PinList '\\n')? 'genera los cuádruplos INPUT'

referenced by:

- [Hardware](#)

OutputsDeclaration:

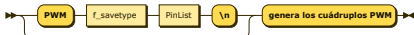


OutputsDeclaration ::= ('OUTPUT' f_savetype PinList '\\n')? 'genera los cuádruplos OUTPUT'

referenced by:

- [Hardware](#)

PwmDeclaration:



PwmDeclaration ::= ('PWM' f_savetype PinList '\\n')? 'genera los cuádruplos PWM'

referenced by:

- [Hardware](#)

PinList:



PinList ::= 'c_int' ':' id (',' 'c_int' ':' id)? 'guarda los id de pines como var. globales'

referenced by:

- [InputsDeclaration](#)
- [OutputsDeclaration](#)
- [PwmDeclaration](#)

Vars:

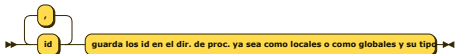


Vars ::= (f_checktab Tipo idList '\\n' f_resetTab)+

referenced by:

- [Program](#)
- [Statement](#)

idList:

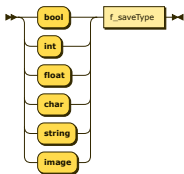


idList ::= 'id' (',' id)? 'guarda los id en el dir. de proc. ya sea como locales o como globales y su tipo'

referenced by:

- [Vars](#)

Tipo:

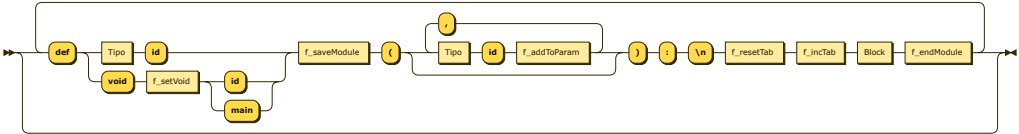


Tipo ::= ('bool' | 'int' | 'float' | 'char' | 'string' | 'image') f_saveType

referenced by:

- Functions
- Vars

Functions:



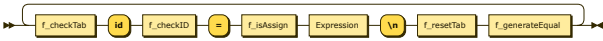
Functions

::= ('def' (Tipo 'id' 'void' f_setVoid ('id' 'main')) f_saveModule (' (Tipo 'id' f_addToParam (',' Tipo 'id' f_addToParam)*)? ')' ':' '\n' f_resetTab f_incTab Block f_endModule)*

referenced by:

- Program

Assign:

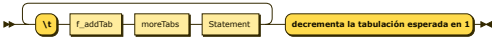


Assign ::= (f_checkTab 'id' f_checkID '=' f_isAssign Expression '\n' f_resetTab f_generateEqual)*

referenced by:

- Program
- Statement

Block:

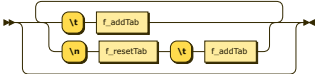


Block ::= ('\t' f_addTab moreTabs Statement)+ 'decrementa la tabulación esperada en 1'

referenced by:

- Condition
- Cycle
- DoCycle
- Functions

moreTabs:

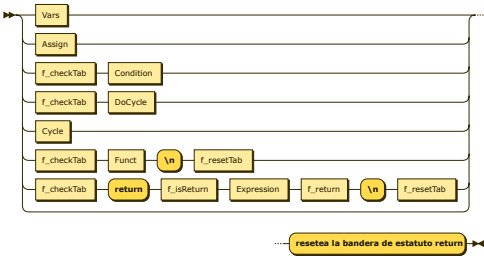


moreTabs ::= ('\t' f_addTab | '\n' f_resetTab '\t' f_addTab)*

referenced by:

- Block

Statement:

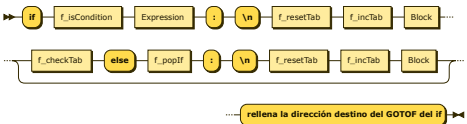


Statement ::= (Vars | Assign | f_checkTab Condition | f_checkTab DoCycle | Cycle | f_checkTab Funct '\n' f_resetTab | f_checkTab 'return' f_isReturn Expression f_return '\n' f_resetTab)? 'resetea la bandera de estatuto return'

referenced by:

- Block

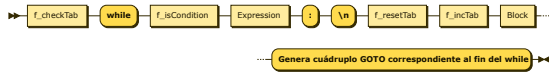
Condition:



Condition ::= 'if' f_isCondition Expression ':' '\n' f_resetTab f_incTab Block (f_checkTab 'else' f_popIf ':' '\n' f_resetTab f_incTab Block)? 'rellena la dirección destino del GOTOF del if'

referenced by:

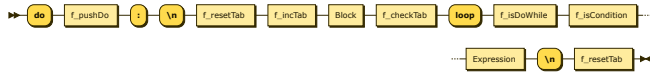
- Statement

Cycle:

Cycle ::= 'f_checkTab 'while' f_isCondition Expression ':' 'f_resetTab f_incTab Block 'Genera cuádruplo GOTO correspondiente al fin del while'

referenced by:

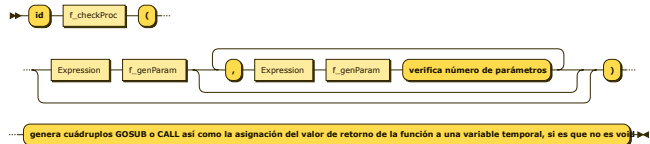
- [Statement](#)

DoCycle:

DoCycle ::= 'do' f_pushDo ':' 'f_resetTab f_incTab Block f_checkTab 'loop' f_isDoWhile f_isCondition Expression ':' 'f_resetTab'

referenced by:

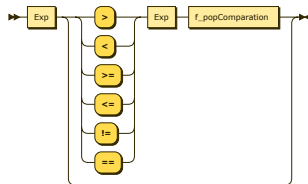
- [Statement](#)

Funct:

Funct ::= 'id' f_checkProc '(' (Expression f_genParam (',' Expression f_genParam 'verifica número de parámetros')*)? ')' 'genera cuádruplos GOSUB o CALL así como la asignación del valor de retorno de la función a una variable temporal, si es que no es void'

referenced by:

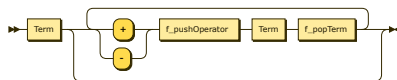
- [Factor](#)
- [Statement](#)

Expression:

Expression ::= Exp (('>' | '<' | '>=' | '<=' | '!=' | '==') Exp f_popComparison)?

referenced by:

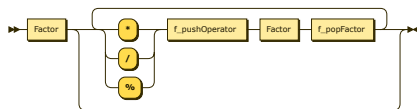
- [Assign](#)
- [Condition](#)
- [Cycle](#)
- [DoCycle](#)
- [Factor](#)
- [Funct](#)
- [Statement](#)

Exp:

Exp ::= Term (('+' | '-' | '*' | '/') f_pushOperator Term f_popTerm)*

referenced by:

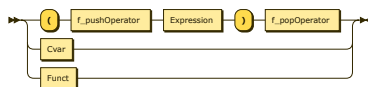
- [Expression](#)

Term:

Term ::= Factor (('*' | '/' | '%') f_pushOperator Factor f_popFactor)*

referenced by:

- [Exp](#)

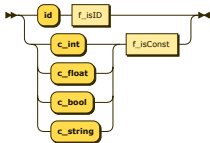
Factor:

Factor ::= '(' f_pushOperator Expression ')' f_popOperator
| Cvar
| Funct

referenced by:

- [Term](#)

Cvar:



Cvar ::= 'id' f_isID
| { 'c_int' | 'c_float' | 'c_bool' | 'c_string' } f_isConst

referenced by:

- [Factor](#)

f_loadVispiFunctions:

➡ **carga las funciones predefinidas de nuestro lenguaje en el dir. de procedimientos** ⬅

f_loadVispiFunctions
::= 'carga las funciones predefinidas de nuestro lenguaje en el dir. de procedimientos'

referenced by:

- [Program](#)

f_saveType:

➡ **guarda el tipo de dato correspondiente al HW o variable primitiva declarada para su uso futuro** ⬅

f_saveType
::= 'guarda el tipo de dato correspondiente al HW o variable primitiva declarada para su uso futuro'

referenced by:

- [Tipo](#)

f_setVoid:

➡ **guarda "void" en el tipo de dato** ⬅

f_setVoid
::= 'guarda "void" en el tipo de dato'

referenced by:

- [Functions](#)

f_saveModule:

➡ **verifica si el id definido ya existe. Declara un nuevo procedimiento en el directorio** ⬅

f_saveModule
::= 'verifica si el id definido ya existe. Declara un nuevo procedimiento en el directorio'

referenced by:

- [Functions](#)

f_addToParam:

➡ **agrega el parametro declarado a su lugar en el dir. de procedimientos** ⬅

f_addToParam
::= 'agrega el parametro declarado a su lugar en el dir. de procedimientos'

referenced by:

- [Functions](#)

f_resetTab:

➡ **reinicia el contador de tabs a 0** ⬅

f_resetTab
::= 'reinicia el contador de tabs a 0'

referenced by:

- [Assign](#)
- [Condition](#)
- [Cycle](#)
- [DoCycle](#)
- [Functions](#)
- [Statement](#)
- [Vars](#)
- [moreTabs](#)

f_incTab:

➡ **incrementa en 1 la tabulación esperada a partir de este punto** ⬅

f_incTab ::= 'incrementa en 1 la tabulación esperada a partir de este punto'

referenced by:

- [Condition](#)
- [Cycle](#)
- [DoCycle](#)
- [Functions](#)

f_decTab:

➡ **decrementa en 1 la tabulación esperada a partir de este punto** ⬅

f_decTab ::= 'decrementa en 1 la tabulación esperada a partir de este punto'

no references

f_endModule:

➡ **guarda el tamaño de la funcion declarada, reinicia contadores de variables y genera el cuad. RE** ⬅

f_endModule
::= 'guarda el tamaño de la funcion declarada, reinicia contadores de variables y genera el cuad. RET'

referenced by:

- [Functions](#)

f_checkTab:

►► verifica que el contador de tabs corresponda con lo esperado ◀◀

f_checkTab ::= 'verifica que el contador de tabs corresponda con lo esperado'

referenced by:

- [Assign](#)
- [Condition](#)
- [Cycle](#)
- [DoCycle](#)
- [Statement](#)

f_checkID:

►► verifica que el id ya haya sido declarado como local o global. Si sí, inserta operando y tipo en sus pilas respectivas ◀◀

f_checkID ::= 'verifica que el id ya haya sido declarado como local o global. Si sí, inserta operando y tipo en sus pilas respectivas'

referenced by:

- [Assign](#)

f_isAssign:

►► enciende una bandera para saber que se procesa una asignación ◀◀

f_isAssign ::= 'enciende una bandera para saber que se procesa una asignación'

referenced by:

- [Assign](#)

f_generateEqual:

►► hace 2 pop de la pila de tipos y de operandos, verifica tipos y genera el cuádruplo de la asignación ◀◀

f_generateEqual ::= 'hace 2 pop de la pila de tipos y de operandos, verifica tipos y genera el cuádruplo de la asignación'

referenced by:

- [Assign](#)

f_addTab:

►► incrementa en 1 el contador de tabs y verifica que corresponda con la tabulación esperada ◀◀

f_addTab ::= 'incrementa en 1 el contador de tabs y verifica que corresponda con la tabulación esperada'

referenced by:

- [Block](#)
- [moreTabs](#)

f_return:

►► marca error si hay return en función void. Verifica que el valor de retorno (top de pila) sea del tipo adecuado. Genera el cuádruplo RETURN ◀◀

f_return ::= 'marca error si hay return en función void. Verifica que el valor de retorno (top de pila) sea del tipo adecuado. Genera el cuádruplo RETURN'

referenced by:

- [Statement](#)

f_isReturn:

►► enciende una bandera que indica que el estatuto es un return ◀◀

f_isReturn ::= 'enciende una bandera que indica que el estatuto es un return'

referenced by:

- [Statement](#)

f_isCondition:

►► enciende una bandera para indicar que se está procesando una condición o ciclo. Si es del tipo while, se inserta el número de cuádruplo en pilaDeBranch ◀◀

f_isCondition ::= 'enciende una bandera para indicar que se está procesando una condición o ciclo. Si es del tipo while, se inserta el número de cuádruplo en pilaDeBranch'

referenced by:

- [Condition](#)
- [Cycle](#)
- [DoCycle](#)

f_popIf:

►► dado que vino un else, genera el GOTO del final del if e inserta en una pilaDeBranch el número del siguiente cuádruplo ◀◀

f_popIf ::= 'dado que vino un else, genera el GOTO del final del if e inserta en una pilaDeBranch el número del siguiente cuádruplo'

referenced by:

- [Condition](#)

f_pushDo:

►► genera un cuádruplo DO que indica el inicio de un do..loop e inserta su número en la pilaDeBranch ◀◀

f_pushDo ::= 'genera un cuádruplo DO que indica el inicio de un do..loop e inserta su número en la pilaDeBranch'

referenced by:

- [DoCycle](#)

f_isDoWhile:

►► enciende una bandera que indica que se está procesando un do..loop ◀◀

f_isDoWhile ::= 'enciende una bandera que indica que se está procesando un do..loop'

referenced by:

- [DoCycle](#)

f_checkProc:

verifica que el módulo llamado esté definido y genera el cuádruplo ERA

f_checkProc ::= 'verifica que el módulo llamado esté definido y genera el cuádruplo ERA'

referenced by:

- Funci

f_genParam:

dados los parámetros de la función, genera los cuádruplos PARAM verificando el tipo

f_genParam ::= 'dados los parámetros de la función, genera los cuádruplos PARAM verificando el tipo'

referenced by:

- Funci

f_popComparison:

si se hizo una comparación: verifica tipo en el cubo semántico, genera una temporal con el resultado (a la pilaOperandos) y genera el cuádruplo correspondiente

f_popComparison ::= 'si se hizo una comparación: verifica tipo en el cubo semántico, genera una temporal con el resultado (a la pilaOperandos) y genera el cuádruplo correspondiente'

referenced by:

- Expresión

f_pushOperator:

siempre que la bandera de condición esté apagada inserta el operador en su pila

f_pushOperator ::= 'siempre que la bandera de condición esté apagada inserta el operador en su pila'

referenced by:

- Exp
- Factor
- Term

f_popTerm:

si la operación fue + o -: verifica tipos en el cubo semántico, genera una temporal con el resultado (a la pilaOperandos) y genera el cuádruplo

f_popTerm ::= 'si la operación fue + o -: verifica tipos en el cubo semántico, genera una temporal con el resultado (a la pilaOperandos) y genera el cuádruplo'

referenced by:

- Exp

f_popFactor:

si es * / o % la operación: verifica tipos en el cubo semántico y genera el cuádruplo

f_popFactor ::= 'si es * / o % la operación: verifica tipos en el cubo semántico y genera el cuádruplo'

referenced by:

- Term

f_popOperator:

si se procesa una condición: verifica que el tipo del operando en la cima del stack sea bool o int, luego genera GOTOF y hace push a pilaDeBranch si es if o while, o hace pop a pilaDeBranch y genera GOTOT si es do. Si no es condición sólo hace pop del operador

f_popOperator ::= 'si se procesa una condición: verifica que el tipo del operando en la cima del stack sea bool o int, luego genera GOTOF y hace push a pilaDeBranch si es if o while, o hace pop a pilaDeBranch y genera GOTOT si es do. Si no es condición sólo hace pop del operador'

referenced by:

- Factor

f_isID:

verifica que el id esté declarado como local o global

f_isID ::= 'verifica que el id esté declarado como local o global'

referenced by:

- CVar

f_isConst:

guarda la constante en la tabla de constantes junto con su tipo y los inserta en la pila de operandos y tipos

f_isConst ::= 'guarda la constante en la tabla de constantes junto con su tipo y los inserta en la pila de operandos y tipos'

referenced by:

- CVar