FORMATO DEL FICHERO CON LOS CUARTETOS OBTENIDOS POR EL GENERADOR DE CÓDIGO INTERMEDIO

Para facilitar la depuración y la corrección de la Práctica de Traductores de Lenguajes y, como se indica en la especificación de la misma (http://dlsiis.fi.upm.es/traductores/Practica.html), el Traductor deberá entregar obligatoriamente los resultados generados en varios archivos de texto (lista de cuartetos, programa en ensamblador). Se detalla aquí el formato que ha de tener el fichero donde se almacenarán los cuartetos obtenidos por el Generador de Código Intermedio.

El lenguaje no diferencia mayúsculas de minúsculas en los cuartetos, pero sí en el nombre de las etiquetas.

El fichero puede tener líneas vacías. Se permiten comentarios precedidos por dos barras (//). El comentario finaliza con el fin de línea (RC). Los comentarios pueden estar formados por cualquier carácter. Un comentario puede situarse en una línea tras el cuarteto o en una línea que tenga sólo el comentario.

Formato de los Cuartetos

Debe escribirse un único cuarteto por cada línea y un cuarteto no puede ocupar más de una línea. Los cuartetos estarán en el fichero en el orden en que son generados por el Generador de Código Intermedio, es decir, el primer cuarteto del fichero será el primer cuarteto que se haya generado.

Los *cuartetos* tendrán el siguiente formato:

```
del* ( del* Operador del* , del* Arg1 del* , del* Arg2 del* , del* Result del* ) del* RC
```

Donde:

- **del*** → cualquier cantidad de espacios en blanco o tabuladores, o nada.
- **Operador** → nombre del operador especificado por el grupo. Está formado por letras, dígitos y subrayados, siendo el primero siempre una letra. Los operadores se consideran palabras reservadas.
- **Arg1**, **Arg2**, **Result** → argumentos del cuarteto. En caso de que el cuarteto no lo requiera, se indica dejándolo vacío o poniendo un guion (–). La descripción detallada de estos campos se muestra en el siguiente apartado.
- **RC** → salto de línea.

***Nota:** Como se ha dicho, en aquellos cuartetos en los que no se requiera rellenar los argumentos, estos estarán vacíos o podrán llevar un guion (–), pero se mantendrán las comas (,) que separan dichos campos siguiendo el formato establecido.

Ejemplo: `(RETURN, , ,)  (RETURN, -, -, -)`

Formato de los argumentos

Los tres campos se representan mediante tuplas *{Clase , Valor}*. Al igual que los operadores de los cuartetos, el campo *Clase* de los argumentos son palabras reservadas especificadas por cada grupo de prácticas.

Los argumentos tendrán el siguiente formato:

```
del* { del* Clase del* , del* Valor del* } del*
```

Donde:

- **del*** → cualquier cantidad de espacios en blanco o tabuladores, o nada.

- **`Clase`** → nombre de la clase del argumento especificado por el grupo en DRACO. Está formado por letras, dígitos y subrayados, siendo el primero siempre una letra. Los nombres de clase se consideran palabras reservadas, y no pueden coincidir con las palabras reservadas de los cuartetos.
- **`Valor`** → es el valor asociado a la clase. En función de la clase, este campo puede tener los siguientes formatos:
    - número: `[+|-]d+` → constante entera con signo opcional (puede usarse para representar un desplazamiento en la Tabla de Símbolos (TS), un valor entero…). El valor absoluto del número no puede ser mayor de 32767.
    - cadena: `"c*"` → constante de tipo cadena entre comillas dobles (puede usarse para representar una cadena, una etiqueta…). No puede contener ni saltos de línea ni EOF.
    - dos números: *número* `del*` , `del*` *número* → dos números como los descritos anteriormente (puede usarse para representar la profundidad y el desplazamiento en las variables no locales).

Lista de Cuartetos

Seguidamente se indican los cuartetos disponibles y que se pueden utilizar. No es obligatorio usar todos los cuartetos, sino solamente aquellos que sean necesarios para la práctica.

En la tabla siguiente se muestra una breve descripción, una representación en formato de código de 3 direcciones y una representación de cada cuarteto. Para esta representación se utiliza una palabra para el operador (los grupos de prácticas pueden utilizar otras palabras) y los argumentos necesarios para el cuarteto, así como su posición dentro del cuarteto. Deberán respetarse los argumentos usados y su posición dentro de cada tipo de cuarteto.

Descripción	Código 3d	Cuarteto
Operación aritmética SUMA	x = y + z	`(SUMA, y, z, x)`
Operación aritmética RESTA	x = y - z	`(RESTA, y, z, x)`
Operación aritmética PRODUCTO	x = y * z	`(MUL, y, z, x)`
Operación aritmética DIVISIÓN	x = y / z	`(DIV, y, z, x)`
Operación aritmética MÓDULO	x = y mod z	`(MOD, y, z, x)`
Operación aritmética EXPONENTE	x = y ^ z	`(EXP, y, z, x)`
Operación lógica AND	x = y AND z	`(AND, y, z, x)`
Operación lógica OR	x = y OR z	`(OR, y, z, x)`
Concatenación de cadenas	x = y concat z	`(CONCAT, y, z, x)`
Menos unario	x = - y	`(MENOS, y, , x)`
Operación lógica NOT	x = not y	`(NOT, y, , x)`
Asignación entre datos enteros	x = y	`(ASIG, y, , x)`
Asignación entre datos cadena	x =c y	`(ASIG_CAD, y, , x)`
Salto incondicional	goto etiqueta	`(GOTO, , , etiqueta)`
Salto condicional si iguales	if x == y goto etiqueta	`(GOTO_IG, x, y, etiqueta)`
Salto condicional si distintos	if x != y goto etiqueta	`(GOTO_DIST, x, y, etiqueta)`
Salto condicional si mayor	if x > y goto etiqueta	`(GOTO_MAY, x, y, etiqueta)`
Salto condicional si menor	if x < y goto etiqueta	`(GOTO_MEN, x, y, etiqueta)`
Salto condicional si mayor o igual	if x >= y goto etiqueta	`(GOTO_MAY_IG, x, y, etiqueta)`
Salto condicional si menor o igual	if x <= y goto etiqueta	`(GOTO_MEN_IG, x, y, etiqueta)`
Paso de parámetro entero	param x	`(PARAM, x, , )`
Paso de parámetro cadena	paramc x	`(PARAM_CAD, x, , )`
Paso de parámetro por referencia	param& x	`(PARAM_REF, x, , )`
Llamada a un procedimiento	call etiqueta	`(CALL, etiqueta, , )`
Llamada a una función con retorno de entero	x = callf	`(CALL_FUN, etiqueta, , x)`
Llamada a una función con retorno de cadena	x = callfc	`(CALL_FUN_CAD, etiqueta, , x)`
Retorno	return	`(RETURN, , , )`

Descripción	Código 3d	Cuarteto
Retorno de entero	return x	(RETURN_ENT, , , x)
Retorno de cadena	returnc x	(RETURN_CAD, , , x)
Imprime un dato entero	print x	(PRINT_ENT, , , x)
Imprime un dato cadena	printc x	(PRINT_CAD, , , x)
Lee un dato entero	input x	(INPUT_ENT, , , x)
Lee un dato cadena	inputc x	(INPUT_CAD, , , x)
Etiqueta	etiqueta :	(ETIQ, etiqueta, ,)
Asignación de una dirección	x = &y	(ASIG_DIR, y, , x)
Asignación del valor de un puntero	x = *y	(ASIG_PTR, y, , x)
Asignación del valor de un puntero que apunta a una cadena	x =c *y	(ASIG_PTR_CAD, y, , x)
Asignación indirecta	*x = y	(PTR_ASIG, y, , x)
Asignación indirecta de cadena	*x =c y	(PTR_ASIG_CAD, y, , x)
Parada de ejecución	halt	(HALT, , ,)

Estructura de los Argumentos

Seguidamente se indica la estructura de los argumentos disponibles para los cuartetos y que se pueden utilizar. No es obligatorio usar todas las clases de argumentos, sino solamente aquellas que sean necesarias para la práctica. Así mismo, es posible tener varias clases con el mismo nombre.

En la tabla siguiente se muestra una breve descripción de las distintas clases de argumentos con la indicación de los valores permitidos, así como una representación del argumento. Para esta representación se utiliza una palabra para la clase de argumento (los grupos de prácticas pueden utilizar otras palabras) y los valores necesarios.

Descripción clase-valor	Tupla
Variable global - desplazamiento en la TS del ámbito global	{VAR_GLOBAL, desplazamiento}
Variable local - desplazamiento en la TS del ámbito local	{VAR_LOCAL, desplazamiento}
Variable no local - diferencia de profundidades - desplazamiento en la TS de dicha variable	{VAR_NOLOCAL, profundidad, desplazamiento}
Variable temporal - desplazamiento en la TS del ámbito local	{VAR_TEMP, desplazamiento}
Parámetro por valor - desplazamiento en la TS del ámbito local	{PAR, desplazamiento}
Parámetro por referencia - desplazamiento en la TS del ámbito local	{PAR_REF, desplazamiento}
Constante entera - valor de la constante	{CTE_ENT, entero}
Constante cadena - la propia cadena	{CTE_CAD, cadena}
Etiqueta - la propia etiqueta	{ET, etiqueta}

Consideraciones Adicionales

En el fichero debe aparecer el cuarteto (ETIQ, {ET, "main"},,) que define la etiqueta “main” y que deberá asociarse al comienzo del programa principal que contenga el fichero.

Las etiquetas deben empezar por una letra que va seguida de otras letras, dígitos o subrayados. Ningún otro carácter será válido en el nombre de una etiqueta. Se recuerda que en el nombre de las etiquetas sí se diferencia entre mayúsculas y minúsculas.

Ejemplos

Se muestran a continuación algunos ejemplos de ficheros que cumplen el formato (en los ejemplos se usan los operadores y nombres de clase mostrados en las tablas anteriores):

- **Ejemplo 1:**

```
( INPUT_ENT,   ,     , {VAR_GLOBAL , -22})              //input x siendo x una variable global
(suma, {VAR_GLOBAL , -22} , { VAR_TEMP, +2} , {VAR_LOCAL , -1})
                            //suma de una variable global y una temporal en una variable local
 (PRINT_ENT,   ,   ,  {VAR_NOLOCAL,  2 , 3})          //print x siendo x una variable no local
  ( Asig, { CTE_ENT, -12345},     , {Var_Local , 1} )
   (  NOT, {VAR_NOLOCAL , 77, -20},   , {VAR_LOCAL , 2}  )
(NOT, {VAR_LOCAL , 2},  , {VAR_GLOBAL , 2})
(Param_Cad, {VAR_GLOBAL, 23},-,-)
(RETURN_ENT,,, {VAR_TEMP, 32})
 ( halt,,,)
(ETIQ, {ET, "main"},,)                                    //etiqueta del programa principal
(asig, { VAR_TEMP, 2}, , { VAR_LOCAL, 55})
(ASIG, {cte_ent, 00},   , {VAR_LOCAL , 0})
(RETURN, , , )
(PRINT_Cad,  ,  , {CTE_CAD, ""} )
(INPUT_ENT,-,-, {VAR_NoLocal, - 5, + 87})
(RETURN_CAD, - , - , { VAR_Local, -2})
(ASIG, {PAR_REF, 0},  , {var_LOCAL, -321})
(ETIQ, {et , "end" }, ,-)
(ASIG, {Cte_ENT , +23456},,{VAR_LOCAL, 0})
```

- **Ejemplo 2:**

```
// A continuación, se muestra el código intermedio de funcion1:
(GoTo,-,-,{Et , "main"})
(ETIQ ,   { ET , "funcion1" }   ,-    ,   -)
( ASIG, {PAR, 0}, , {VAR_LOCAL, 1})
( PRINT_ent ,    ,  - ,   { VAR_LOCAL ,  1} )
( ASIG, {PAR,0},   , {VAR_LOCAL,2})
( ASIG, {CTE_ENT,5}, , {VAR_LOCAL,3})
(GOTO_May, {VAR_LOCAL , 2 }, {VAR_LOCAL, 3}, {ET, "etiq0"})
( ASIG, {PAR , 0},     , {VAR_LOCAL , 4})
( ASIG, {CTE_ENT, -1},   , {VAR_LOCAL , 5})
( SUMA, {VAR_LOCAL , 4} , { VAR_LOCAL, 5 } , {VAR_TEMP , 6} )
(PARAM , {VAR_TEMP , +06}, , )
(CALL,{ET, "funcion1"},- ,- )
(etiq , { et , "etiq0" } ,,)
  (RETURN,,,)

// Código intermedio del ¡programa principal! que llama a funcion1:
(ETIQ , {ET , "main"} , - , - )
  (ASIG, {CTE_ENT, 0}, , {VAR_LOCAL , 0})
  (PARAM, {VAR_LOCAL , 0}, , )
  (Call,  {et , "funcion1"} ,  -  , -)
  (PRINT_CAD, -, , {CTE_CAD, "<<+ ¡fin! +>>"})
(halt,-,-,-)
```

Seguidamente se muestra un ejemplo de fichero que no cumple el formato (todos los cuartetos son incorrectos):

- **Ejemplo 3:**

```
INPUT_ENT,  ,  , {VAR_GLOBAL , -22}              //ERROR: el cuarteto debe ir entre paréntesis
(NOT, VAR_LOCAL , 2,  , {VAR_GLOBAL , 2} )       //ERROR: faltan llaves en el primer argumento
(PARAM_CAD, {VAR_GLOBAL, 02} )         //ERROR: el cuarteto está incompleto, faltan dos campos
(RETURN, {VAR_TEMP, 2}, , )
                  //ERROR: en la estructura del cuarteto que representa return x no se usa el
                  //       segundo y el tercer campo; el argumento debe situarse en el cuarto
(ASIG, { VAR_TEMP, 2},  { VAR_TEMP, 3}, { VAR_LOCAL, 5} )
                        //ERROR: el segundo argumento debe estar vacío: (ASIG, Arg1, , Result)
(ASIG, {CTE_ENT, 0},   , {VAR_LOCAL , 0})   (RETURN,  ,  ,  )
                                                  //ERROR: dos cuartetos en la misma línea
{SUMA,(Par,8),(CTE_ENT,5),(VAR_LOCAL,4)}        //ERROR: uso incorrecto de llaves y paréntesis
(INPUT_ENT,-,-, {VAR_NOLOCAL, 1})            //ERROR: falta un valor para la variable no local
(RETURN, , , { VAR_LOCAL, -2} )     //ERROR: el cuarteto debe tener los tres argumentos vacíos
(ASIG, {VAR_LOCAL, 0},  , {CTE_ENT , 0})          //ERROR: no se puede asignar a una constante
(ETIQ, {ET , funcion1}, ,-)    //ERROR: funcion1 debe ir entre comillas porque es una etiqueta
(ASIG, {VAR_LOCAL, 0}, _ , {CTE_ENT , 0})  //ERROR: el tercer campo debe ser vacío o guion (-)
(INPUT_CAD,  ,  , {VAR_NO_LOCAL , 1})                    //ERROR: clase VAR_NO_LOCAL no válida
(ETIQ, {VAR_LOCAL,4},,-)        //ERROR: el cuarteto ETIQ espera una etiqueta, no una variable
(PARAM,{ET,"func1"}, , )     //ERROR: el argumento del cuarteto PARAM no puede ser de clase ET
(ASIG, {CTE_ENT, 1},  , {VAR_LOCAL, "cinco"})
                                //ERROR: el valor de la clase VAR_LOCAL no puede ser una cadena
(CALL_FUN, {ET, "etiq0"}, ,)                //ERROR: el último argumento no puede estar vacío
(RETURN_ENT,,, {VAR_LOCAL,-})        //ERROR: no se puede dejar el valor de un argumento vacío
(PRINT_ENT,  ,  ,  {VAR_LOCAL,  1,  4}  )
                        //ERROR: la tupla del argumento de clase VAR_LOCAL solo tiene un valor
(GOTO_May, {VAR_LOCAL , 2 }, {VAR_LOCAL, 3}, {ET, "etiq0"}, {VAR_LOCAL , 8})
                                                  //ERROR: un cuarteto debe tener solo 4 campos
(PRINT_ENT,  ,  , {CTE_CAD, "hola"} )
               //ERROR: el cuarteto PRINT_ENT no puede tener un argumento de la clase CTE_CAD
(PRINT,  ,    , {CTE_ENT, -0})                            //ERROR: no existe el cuarteto PRINT
(ASIG_PTR_CAD,{CTE_CAD,"5"},,{par_ref,4})     //ERROR: el primer argumento no puede ser cadena
(Asig, {CTE_Ent, 33333},, {VAR_LOCAL , 5})    //ERROR: el valor está fuera del rango permitido
(PRINT_CAD,  ,    , {CTE_CAD, hola}  )          //ERROR: la cadena hola debe ir entre comillas
(suma, {VAR_GLOBAL , -22} , { VAR_TEMP, +2},       //ERROR: comentario en medio de un cuarteto
       {VAR_LOCAL , -1} )                    //ERROR: un cuarteto no puede estar en dos líneas
(PARAM, {Cte_Cad, "proc1}, , )                              //ERROR: la cadena no está cerrada
(ASIG,{PAR,-9},--,{Par,9})      //ERROR: un campo vacío se deja en blanco o con un único guion
(PRINT-ENT,,,{VAR_LOCAL,1})     //ERROR: el nombre de operador solo admite letras, dígitos y _
(PRINT_ENT,,,{VAR+LOCAL,1})        //ERROR: el nombre de clase solo admite letras, dígitos y _
                                                  //ERROR: no está declarada la etiqueta “main”
```