Tabla de traducción de lenguaje algorítmico a lenguaje C

booleano
cierto, **falso**
:=
no, y, o, =, ?, <, =, >, =
	bool[1]
	TRUE, FALSE
	=
	!, &&, ||, ==, !=, <, <=, >, >=
	
	[1] hace falta definer previamente el tipo como:
	typedef enum {FALSE, TRUE} bool;
		si *expresión* ***entonces***
		acciónA
		sino
		acciónB
		fsi
			if (*expresión*) {
			acciónA
			} **else** {
			acciónB
			}

caracter
'a', 'W', '1'
:=
=, ?, <, =, >,=
	char
	'a', 'W', '1'
	=
	==, !=, <, <=, >, >=
		mientras *expresión* ***hacer***
		acción
		fmientras
			while (*expresión*) {
			acción
			}

entero
3, 465434, -2134567
- (cambio de signo), +, -, *, div, mod, :=
=, ?, <, =, >, =
	int
	3, 465434, -2134567
	- (cambio de signo), +, -, *, /, %, =
	==, !=, <, <=, >, >=
		para *indice := v_ini* ***hasta*** *v_fi* ***hacer***
		acción
		fpara
			para (indice = *v_ini*; *indice* <= *v_fi*; *indice*++)
			{
			acción
			}

real
0.6, 5.0E-8, -49.22E+0.8, 1.0E5, 4.0
- (cambio de signo), +, -, *, /, :=
=, ?, <, =, >, =
	float
	0.6, 5.0E-8, -49.22E+0.8, 1.0E5, 4.0
	- (cambio de signo), +, -, *, /, =
	==, !=, <, <=, >, >=
		acción *nombre*(pm$_1$, *pm$_2$*.., *p$_{mn}$*)
		... cuerpo de la acción ...
		facción
		nom(*obj$_1$*, *obj$_2$*..., *obj$_{jn}$*);
			void *nombre*(pm$_1$, *pm$_2$*..., *p$_{mn}$*) {
			... cuerpo de la acción
			}
			nom(*obj$_1$*, *obj$_2$*..., *ob$_{jn}$*);

realAEntero(r), enteroAReal(e)
caracterACodigo(c), codigoACaracter(e)
	(**int**)r, (**float**)e
	(**int**)c, (**char**)e
		funcion *nombre*(pm$_1$, *pm$_2$*.., *p$_{mn}$*): *tipo*
		... cuerpo de la función
		devuelve *expresión;*
		ffuncion
		
		nombre(*obj$_1$*, *obj$_2$*..., *obj$_n$*)
			tipo nom(*pm$_1$*, *pm$_2$*..., *p$_{mn}$*)
			{
			... cuerpo de la función
			return *expresión;*
			}
			nombre(*obj$_1$*, *obj$_2$*..., *obj$_n$*)

e := leerEntero(); escribirEntero(e);
r := leerReal(); escribirReal(r);
c := leerCaracter(); escribirCaracter(c);
	scanf("%d", &e); printf("%d, e);
	scanf("%f", &r); printf("%f, r);
	scanf("%c", &c); printf("%c ", c);
	
	#include <stdio.h> (al principio del programa)		

const *nombre*: *tipo = valor*; **fconst**	**#define** *NOMBRE valor* (*o también*) **const** *tipo nombre = valor*;	***ent*** *nombre: tipo* ***sal*** *nombre: tipo* ***entsal*** *nombre: tipo*	*tipo nombre* *tipo *nombre* (**nombre* en el cuerpo, *&nombre* en la llamada[2]) *tipo *nombre* (**nombre* en el cuerpo, *&nombre* en la llamada[2]) [2] **Consultar excepción en el manual de C (pág. 22)**
tipo *nombre* = {v$_1$, v$_2$ ... ,v$_{vn}$}; **ftipo** **tipo** *nombre = definición*; **ftipo**	**typedef enum** {v$_1$, v$_2$ ... ,v$_n$} nombre; **typedef** *definición nombre*;	***ent*** *nombre: tipuo_tabla* ***sal*** *nombre: tipo_tabla* ***entsal*** *nombre: tipo_tabla*	**const tipo_tabla** *nombre* *tipo_tabla nombre* *tipus_tabla nombre*
var *nombre*: *tipo*; **fvar** **var** *n$_1$*, *n$_2$*, *n$_3$*: *tipo*; **fvar**	*tipo nombre*; *tipo n$_1$*, *n$_2$*, *n$_3$*;	raiz(r)	**#include <math.h>** (al principio del programa) sqrt(r);

nombre: **tabla**[*tamaño*] **de** *tipo*;
nombre: **taula**[*tamaño$_1$*, *tamaño$_2$*..., *tamaño$_n$*] **de** *tipo*;
nombre: **taula**[*tamaño$_1$*][*tamaño$_2$*]...[*tamaño$_n$*] **de** *tipo*;
índice de 1 a tamaño
	tipo nom[*tamaño*];
	tipo nom[*tamaño$_1$*][*tamaño$_2$*]...[*tamaño$_n$*];
	tipo nom[*tamaño$_1$*][*tamaño$_2$*]...[*tamaño$_n$*];
	índice de 0 a tamaño-1 ***(MUY IMPORTANTE)***
		funcion *nombre*(pm$_1$..., *p$_m$*): *tipo_tabla*
		... cuerpo de la función ...
		devuelve *nombre_tabla;*
		ffuncion
			void nombre(*pm$_1$*, ...,*p$_{mn}$*, *tipo_tabla nombre_tabla*)
			{
			... cuerpo de la función ...
			}

nombre: **tupla**
campo$_1$: *tipo$_1$*;
campo$_2$: *tipo$_2$*;
...
ftupla
 	
nombre.campo
	struct {
	tipo1 campo$_1$;
	tipo2 campo$_2$;
	...
	} *nombre*;
	
	nombre.campo
		accion **nombre**(entsal *nombre tipo_tupla*)
		nombre.campo
		faccion
			void *nombre*(tipo_tupla **nombre*)
			{
			(**nombre*).*campo*
			(o también)
			nombre->campo
			}