# Tabla de traducción de lenguaje algorítmico a lenguaje C

| | | | |
|---|---|---|---|
| **booleano**<br>**cierto**, **falso**<br>:=<br>no, y, o, =, ?, <, =, >, = | **bool**[1]<br>TRUE, FALSE<br>=<br>!, &&, \|\|, ==, !=, <, <=, >, >=<br><br>[1] hace falta definer previamente el tipo como:<br>**typedef enum** {FALSE, TRUE} bool; | ***si*** *expresión* ***entonces***<br>*acciónA*<br>***sino***<br>*acciónB*<br>***fsi*** | **if** (*expresión*) {<br>*acciónA*<br>} **else** {<br>*acciónB*<br>} |
| **caracter**<br>'a', 'W', '1'<br>:=<br>=, ?, <, =, >,= | **char**<br>'a', 'W', '1'<br>=<br>==, !=, <, <=, >, >= | ***mientras*** *expresión* ***hacer***<br>*acción*<br>***fmientras*** | **while** (*expresión*) {<br>*acción*<br>} |
| **entero**<br>3, 465434, -2134567<br>- (cambio de signo), +, -, *, div, mod, :=<br>=, ?, <, =, >, = | **int**<br>3, 465434, -2134567<br>- (cambio de signo), +, -, *, /, %, =<br>==, !=, <, <=, >, >= | ***para*** *indice := v_ini* ***hasta*** *v_fi* ***hacer***<br>*acción*<br>***fpara*** | **para** (indice = *v_ini*; *indice* <= *v_fi*; *indice*++)<br>{<br>*acción*<br>} |
| **real**<br>0.6, 5.0E-8, -49.22E+0.8, 1.0E5, 4.0<br>- (cambio de signo), +, -, *, /, :=<br>=, ?, <, =, >, = | **float**<br>0.6, 5.0E-8, -49.22E+0.8, 1.0E5, 4.0<br>- (cambio de signo), +, -, *, /, =<br>==, !=, <, <=, >, >= | ***acción*** *nombre*(pm$_1$, *pm$_2$*.., *p$_{mn}$*)<br>... *cuerpo de la acción* ...<br>***facción***<br>*nom*(*obj$_1$*, *obj$_2$*..., *ob$_{jn}$*); | **void** *nombre*(pm$_1$, *pm$_2$*..., *p$_{mn}$*) {<br>*... cuerpo de la acción*<br>}<br>*nom*(*obj$_1$*, *obj$_2$*.., *ob$_{jn}$*); |
| realAEntero(r), enteroAReal(e)<br>caracterACodigo(c), codigoACaracter(e) | (**int**)r, (**float**)e<br>(**int**)c, (**char**)e | ***funcion*** *nombre*(pm$_1$, *pm$_2$*.., *p$_{mn}$*): *tipo*<br>*... cuerpo de la función*<br>**devuelve** *expresión;*<br>***ffuncion***<br><br>*nombre*(*obj$_1$*, *obj$_2$*..., *obj$_n$*) | *tipo nom*(*pm$_1$*, *pm$_2$*.., *p$_{mn}$*)<br>{<br>*... cuerpo de la función*<br>**return** *expresión;*<br>}<br>*nombre*(*obj$_1$*, *obj$_2$*.., *obj$_n$*) |
| e := leerEntero(); escribirEntero(e);<br>r := leerReal(); escribirReal(r);<br>c := leerCaracter(); escribirCaracter(c); | scanf("%d", &e); printf("%d, e);<br>scanf("%f", &r); printf("%f, r);<br>scanf("%c", &c); printf("%c ", c);<br><br>**#include <stdio.h>** (al principio del programa) | | |
| **const**<br>*nombre*: *tipo* = *valor*;<br>**fconst** | **#define** *NOMBRE valor*<br><br>(*o también*)<br>**const** *tipo nombre* = *valor*; | ***ent*** *nombre: tipo*<br>***sal*** *nombre: tipo*<br>***entsal*** *nombre: tipo* | *tipo nombre*<br>*tipo *nombre* (*nombre* en el cuerpo, *&nombre* en la llamada[2])<br>*tipo *nombre* (*nombre* en el cuerpo, *&nombre* en la llamada[2])<br>[2] **Consultar excepción en el manual de C (pág. 22)** |
| **tipo** *nombre* = {*v$_1$*, *v$_2$* ... ,*v$_{vn}$*}; **ftipo**<br>**tipo** *nombre* = *definición*; **ftipo** | **typedef enum** {*v$_1$*, *v$_2$* ... ,*v$_n$*} nombre;<br>**typedef** *definición nombre*; | ***ent*** *nombre: tipuo_tabla*<br>***sal*** *nombre: tipo_tabla*<br>***entsal*** *nombre: tipo_tabla* | **const tipo_tabla** *nombre*<br>*tipo_tabla nombre*<br>*tipus_tabla nombre* |
| **var** *nombre*: *tipo*; **fvar**<br>**var** *n$_1$*, *n$_2$*, *n$_3$*: *tipo*; **fvar** | *tipo nombre*;<br>*tipo n$_1$*, *n$_2$*, *n$_3$*; | raiz(r) | **#include <math.h>** (al principio del programa)<br>sqrt(r); |
| *nombre*: **tabla**[*tamaño*] **de** *tipo*;<br>*nombre*: **taula**[*tamaño$_1$*, *tamaño$_2$*.., *tamaño$_n$*] **de** *tipo*;<br>*nombre*: **taula**[*tamaño$_1$*][*tamaño$_2$*]...[*tamaño$_n$*] **de** *tipo*; | *tipo nom*[*tamaño*];<br>*tipo nom*[*tamaño$_1$*][*tamaño$_2$*]...[*tamaño$_n$*];<br>*tipo nom*[*tamaño$_1$*][*tamaño$_2$*]...[*tamaño$_n$*]; | ***funcion*** *nombre*(pm$_1$..., *p$_m$*): *tipo_tabla*<br>*... cuerpo de la función ...*<br>**devuelve** *nombre_tabla;* | **void nombre**(*pm$_1$*, ...,*p$_{mn}$*, *tipo_tabla nombre_tabla*)<br>{<br>*... cuerpo de la función ...* |