FORTRAN 77

(**ANSI** y `extensiones FORTRAN/9000`)

Unidades de programa

BLOCK DATA [nombre]
END
ENTRY nombre[([arg1 [,arg2] ...])
[tipo] **FUNCTION** nombre[*long]([arg1[,arg2] ...])
PROGRAM nombre [(lista)]
SUBROUTINE nombre [([arg1 [,arg2] ...])]

Instrucciones de Especificación

COMMON [/[block1]/] lista1 [[,]/[block2]/ lista2] ...
DIMENSION nombre1(limites) [,nombre2(limites)] ...
EQUIVALENCE (lista1) [,(lista2)] ...
EXTERNAL proced1 [,proced2] ...
IMPLICIT tipo(rang1[,rang2] ...)[,tipo(rang1 [,rang2] ...)]...
`IMPLICIT NONE`
`INCLUDE` nombre
INTRINSIC fun1[,fun2]...
nombre([param1 [,param2] ... **]) = expresión**
`MAP` definición_campo ... `END MAP`
`NAMELIST` /grupo1/lista-var[[,]/grupo2/lista-var2] ...
PARAMETER(cte1=exp_cte1[,cte2=exp_cte2] ...)
`RECORD` /nomb_estr/ reg[,reg] ... [,/nomb_estr/ lista_reg] ...
RETURN [nro_retorno]
SAVE [var1 [,var2] ...]
`STRUCTURE /nombre/` def_campo ... `END STRUCTURE`
`UNION`
`VOLATILE` lista

Asignación

ASSIGN etiqueta **TO** variable
var = expresión
DATA list_var1/list_con1/[[,]list_var2/list_con2/] ...
tipo nombre1 [/lista_con1/]**[,nombre2** [/lista_con2/]] ...

Entrada/Salida

`ACCEPT` Equivalente a READ(UNIT=5).
BACKSPACE unidad
BACKSPACE ([`UNIT`=]unidad[,`IOSTAT`=ios][,`ERR`=etiq])
CLOSE([`UNIT`=]unidad[,`IOSTAT`=ios][,`ERR`=etiq])[,`STATUS`=e])
`DECODE`(c,[`FMT`=]fmto,[`UNIT`=]u[,`IOSTAT`=ios][,`ERR`=etiq])[lista]
`ENCODE`(c,[`FMT`=]fmto,[`UNIT`=]u[,`IOSTAT`=ios][,`ERR`=etiq])[lista]
ENDFILE unidad
ENDFILE ([`UNIT`=]unidad[,`IOSTAT`=ios][,`ERR`=etiq])
etiqueta **FORMAT** (desc1[,desc2] ...)
INQUIRE([`UNIT`=]u[,`FILE`=nomb[,`IOSTAT`=ios][,`ERR`=e][lista])
OPEN([`UNIT`=]u [,`FILE`=nomb[,`IOSTAT`=ios][,`ERR`=etiq]...)
PRINT formato [,lista]
PRINT * [,lista]
PRINT nml
READ([`UNIT`=]u[`FMT`=]fmto[,`END`=e[,`ERR`=e][,`IOSTAT`=i])[lista]
READ fmto[,lista]
READ([`UNIT`=]u,* [,`END`=etiq[,`ERR`=etiq][,`IOSTAT`=ios])[lista]
READ * [,lista]
`READ`([`UNIT`=]u,[`NML`=]nml [,`END`=e][,`ERR`=e][,`IOSTAT`=ios])
`READ` nml
READ([`UNIT`=]u [,`END`=etiq[,`ERR`=etiq][,`IOSTAT`=ios])[lista]
READ([`UNIT`=]u,[`FMT`=]f [,`REC`=n] [,`ERR`=e][,`IOSTAT`=i])[list]
READ([`UNIT`=]u [,`REC`=nro_reg] [,`ERR`=e][,`IOSTAT`=ios])[list]
READ([`UNIT`=]ui,[`FMT`=]f [,`END`=e] [,`ERR`=e][,`IOSTAT`=i])[list]
`READ`([`UNIT`=]ui, * [,`END`=etiq] [,`ERR`=etiq][,`IOSTAT`=ios])[lista]
REWIND
REWIND ([`UNIT`=]u [,`IOSTAT`=ios][,`ERR`=etiq])
`TYPE` – Ver `PRINT`
WRITE([`UNIT`=]u,[`FMT`=]fmto[,`ERR`=etiq][,`IOSTAT`=ios])[lista]
WRITE([`UNIT`=]u,* [,`ERR`=etiq][,`IOSTAT`=ios])[lista]
`WRITE`([`UNIT`=]u,[`NML`=] nml [,`ERR`=etiq][,`IOSTAT`=ios])
WRITE([`UNIT`=]u [,`ERR`=etiq][,`IOSTAT`=ios])[lista]
WRITE([`UNIT`=]u,[`FMT`=]f [,`REC`=n] [,`ERR`=e][,`IOSTAT`=i])[list]
WRITE([`UNIT`=]u [,`REC`=nro_reg] [,`ERR`=e][,`IOSTAT`=i])[list]
WRITE([`UNIT`=]ui,[`FMT`=]fmto [,`ERR`=etiq][,`IOSTAT`=i])[list]
`WRITE`([`UNIT`=]ui, * [,`ERR`=etiq][,`IOSTAT`=ios])[lista]

Interrupción

PAUSE [n]
STOP [n]

Sentencias de control

CALL nombre[([arg1 [,arg2] ...])]
RETURN [nro_retorno]
CONTINUE

DO [etiqueta [,]] índice=inicio,final[,paso]
`DO` índice=inicio,final[,paso] ... `END DO`
`DO` [etiqueta [,]] `WHILE` (expr_lógica) ... `END DO`

GOTO etiqueta (`GOTO` *incondicional*)
GOTO (etiq1,etiq2,...)[,] expresión (*calculado*)
GOTO var_entera [[,] (etiq1[,etiq2] ...)] (*asignado*)

IF (expr_aritm) etiq[–],etiq[0],etiq[+] (`IF` *aritmético*)
IF (expr_lógica) sentencia (*lógico*)
IF (expr_lógica) **THEN** ...
[**ELSE IF** (expr_lógica) **THEN** ...]
[**ELSE** ...]
ENDIF (*bloque IF*)

EJEMPLOS DE COMPILACIÓN CON f77

f77 -C -L -o test test.f > test.lst Compila el fichero fuente *test.f*, con el chequeo de rangos activado, produciendo el ejecutable *test*, con el listado enviado al fichero *test.lst*.

f77 -c -w ftn1.f Compila el fichero fuente *ftn1.f*, suprimiendo el aviso de warnings y produciendo el fichero objeto *ftn1.o* . El linkado se suprime.

f77 quidnum.f -o quidnum Compila el fichero fuente *quidnum.f*, produciendo el ejecutable *quidnum* .

f77 trix.f treats.f Compila los ficheros *trix.f* y *treats.f* produciendo el fichero ejecutable *a.out* .

f77 /usr/jkl/prog.f -o prog Compila el fichero fuente *prog.f* del directorio */usr/jkl*, produciendo el fichero ejecutable *prog* en el directorio actual.

f77 -g prog1.f -o prog1 Compila el fichero fuente *prog1.f*, produciendo el ejecutable *prog1* con la información necesaria para poder ejecutar el debugger (hacer *man xdb(1)*).

f77 -A prog2.f proc.f -o prog2 Compila los ficheros fuentes *prog2.f* y *proc.f*, produciendo el ejecutable *prog2* sólo si los fuentes siguen el ANSI estricto (hacer *man f77(1)*).

f77 -G prog3.f -o prog3 Compila el fichero fuente *prog3.f*, produciendo el ejecutable *prog3* con la información necesaria para generar el *gmon.out* a efectos de optimización de código (hacer *man gprof(1)*).

Instrucciones de Declaración

Tipo : ocupación Declaraciones	Rango de valores

Entero : 4 bytes
`INTEGER` *v1*`[,`*v2* ...`]`
`INTEGER*4` *v1*`[,`*v2* ...`]*`
	$-2\,147\,483\,648 \quad a \quad +2\,147\,483\,647$

Entero corto : 2 bytes `INTEGER*2` *v1*`[,`*v2* ...`]*`	$-32\,768 \quad a \quad +32\,767$

Real : 4 bytes
`REAL` *v1*`[,`*v2* ...`]`
`REAL*4` *v1*`[,`*v2* ...`]*`
	$-3.402\,823 \times 10^{+38}$ a $-1.175\,495 \times 10^{-38}$
	ó
	$+1.175\,495 \times 10^{-38}$ a $+3.402\,823 \times 10^{+38}$

Doble precisión : 8 bytes
`DOUBLE PRECISION`
`REAL*8` *v1*`[,`*v2* ...`]*`
	$-1.797\ 693\ 134\ 862\ 31 \times 10^{+308}$ a $-2.225\ 073\ 858\ 507\ 21 \times 10^{-308}$
	ó
	$+2.225\ 073\ 858\ 507\ 21 \times 10^{-308}$ a $+1.797\ 693\ 134\ 862\ 31 \times 10^{+308}$

Cuádruple p.: 16 bytes `REAL*16` *v1*`[,`*v2* ...`]*`	$\pm 3.362\,103\,143\,112\,093\,506\ldots$ $\ldots 262\,677\,817\,321\,753 \times 10^{-4932}$ a $\pm 1.189\,731\,495\,357\,231\,765\ldots$ $\ldots 085\,759\,326\,628\,007 \times 10^{+4932}$

Complejo : 8 bytes
`COMPLEX` *v1*`[,`*v2* ...`]*`
`COMPLEX*8` *v1*`[,`*v2* ...`]*`
	mismo que real

Complejo doble : 16 bytes
`DOUBLE COMPLEX*`
`COMPLEX*16` *v1*`[,`*v2* ...`]*`
	mismo que doble precisión

Lógico : 4 bytes
`LOGICAL` *v1*`[,`*v2* ...`]`
`LOGICAL*4` *v1*`[,`*v2* ...`]*`
	.TRUE. o .FALSE.
	mismo que entero

Lógico corto : 2 bytes `LOGICAL*2` *v1*`[,`*v2* ...`]*`	.TRUE. o .FALSE. mismo que entero corto

Byte : 1 byte
`BYTE` *v1*`[,`*v2* ...`]*`
`LOGICAL*1` *v1*`[,`*v2* ...`]*`
	.TRUE. o .FALSE.
	$-128 \quad a \quad +127$
	conjunto completo car. ASCII (8 bits)

Carácter : longitud `CHARACTER*`*long* `CHARACTER*(*)`	conjunto completo car. ASCII (8 bits)

* (extensión HP)

Especificaciones de formato

Descriptores de edición

Dptor	Función
BN	Ignora blancos en entrada numérica
BZ	Trata los blancos como ceros en entrada numérica
NL*	Finaliza la salida con newline
NN*	Suprime newline al final de la salida
$*	Misma que NN
Q*	Retorna el no.de bytes restante registro actual
S	El compilador determina el signo de la salida
SP	Imprime un signo "+" opcional en salida
SS	Inhibe el signo "+" opcional en salida
'...'	Apóstrofes para edición literal en salida
"..."*	Comillas para edición literal de salida
*n*H	Edición de caracteres Hollerith (sólo salida)
*n*X	Salta n columnas a la derecha
T*c*	Salta a la columna c
TL*c*	Salta c columnas a la izquierda
TR*c*	Salta c columnas a la derecha
/	Termina el registro actual y comienza un nuevo reg.
:	Termina el control formato si resto lista vacia
*k*P	Factor de escala para entrada y salida

* (extensión HP)

Descriptores de formato

Descriptor	Explicación
[r]I[w[.d]]	Entero y entero corto*
[r]F[w[.d]]	Formato de coma fija (real sin exponente)
[r]D[w[.d]]	Formato de coma flotante
[r]E[w[.d]][Ee]	Idem (real con exponente)
[r]G[w[.d]][Ee]	Real general(comas fija y flotante)
[r]A[w]	tipo carácter, justif. izq. en memoria
[r]R[w]*	tipo carácter, justif. dcha. en memoria
[r]L[w]	dato tipo lógico, entero corto* y byte*
[r]K[w]*	octal; entero y entero corto
[r]@[w]*	octal; entero y entero corto
[r]O[w]*	octal; entero y entero corto
[r]Z[n[.m]]*	hexadecimal;cualquier dato

* (extensión HP)

Funciones Intrínsecas

ABS	BSHFTC[4]	DDINT[1]	HIXOR[1]	IMAX0[3]	JMOD[3]	QFLOTI[3]
ACOS	BSIGN[4]	DEXP	HMOD[1]	IMAX1[3]	JNINT[3]	QFLOTJ[3]
ACOSD[3]	BTEST[2,6]	DFLOAT[3]	HMVBITS[1]	IMIN0[3]	JNOT[3]	QINT[3]
ACOSH[1]	CABS	DFLOTI[3]	HNOT[1]	IMIN1[3]	JNUM[4]	QLOG[3]
AIMAG	CCOS	DFLOTJ[3]	HSHFT[1]	IMOD[3]	JZEXT[1]	QLOG10[3]
AIMAX0[3]	CDABS[3]	DIM	HSHFTC[1]	INDEX	LEN	QMAX1[3]
AIMIN0[3]	CDCOS[3]	DIMAG[1]	HSIGN[1]	ININT[3]	LGE	QMIN1[3]
AINT	CDEXP[3]	DINT	HTEST[1]	INUM[4]	LGT	QMOD[3]
AJMAX0[3]	CDLOG[3]	DLOG	IABS	INOT[3]	LLE	QNINT[3]
AJMIN0[3]	CDSIN[3]	DLOG10	IAND[2,6]	INT	LLT	QNUM[3]
ALOG	CDSQRT[3]	DMAX1	IARGC[1]	IOR[2,6]	LOG	QPROD[3]
ALOG10	CEXP	DMIN1	IBCLR[2,6]	IQINT[3]	LOG10	QSIGN[3]
AMAX0	CHAR	DMOD	IBITS[2,6]	IQNINT[3]	MAX	QSIN[3]
AMAX1	CLOG	DNINT	IBSET[2,6]	IRAND[4]	MAX0	QSIND[3]
AMIN0	CMPLX	DNUM[4]	ICHAR	IRANP[4]	MAX1	QSINH[3]
AMIN1	CONJG	DPROD	IDIM	ISHFT[2,6]	MIN	QSQRT[3]
AMOD	COS	DREAL[3]	IDINT	ISHFTC[2,6]	MIN0	QTAN[3]
ANINT	COSD[3]	DSIGN	IDNINT	ISIGN	MIN1	QTAND[3]
ASIN	COSH	DSIN	IEOR[2,6]	IXOR[1]	MOD	QTANH[3]
ASIND[3]	CSIN	DSIND[3]	IFIX	IZEXT[1]	MVBITS[2,6]	RAND[4]
ASINH[1]	CSQRT	DSINH	IGETARG[1]	JIABS[3]	NINT	REAL
ATAN	CTAN[1]	DSQRT	IIABS[3]	JIAND[3]	NOT[2]	RNUM[4]
ATAN2	DABS	DTAN	IIAND[3]	JIBCLR[3]	QABS[3]	SIGN
ATAN2D[3]	DACOS	DTAND[3]	IIBCLR[3]	JIBITS[3]	QACOS[3]	SIN
ATAND[3]	DACOSD[3]	DTANH	IIBITS[3]	JIBSET[3]	QACOSD[3]	SIND[3]
ATANH[1]	DACOSH[1]	EXP	IIBSET[3]	JIDIM[3]	QACOSH[3]	SINH
BABS[4]	DASIN	FLOAT	IIDIM[3]	JIDINT[3]	QASIN[3]	SIZEOF[4]
BADDRESS[4]	DASIND[3]	FLOATI[3]	IIDINT[3]	JIDNNT[3]	QASIND[3]	SNGL
BBCLR[4]	DASINH[1]	FLOATJ[3]	IIDNNT[3]	JIEOR[3]	QASINH[3]	SNGLQ[3]
BBITS[4]	DATAN	FNUM[1]	IIEOR[3]	JIFIX[3]	QATAN[3]	SRAND[4]
BBSET[4]	DATAN2	FSET[4]	IIFIX[3]	JINT[3]	QATAND[3]	SQRT
BBTEST[4]	DATAN2D[3]	FSTREAM[1]	IINT[3]	JIOR[3]	QATANH[3]	TAN
BDIM[4]	DATAND[3]	GETARG[4]	IIOR[3]	JIQINT[3]	QATAN2[3]	TAND[3]
BIAND[4]	DATANH[1]	GRAN[4]	IIQINT[3]	JIQNNT[3]	QATAN2D[3]	TANH
BIOR[4]	DBLE	HABS[1]	IIQNINT[3]	JISHFT[3]	QCOS[3]	ZABS[1]
BITEST[3]	DBLEQ[3]	HBCLR[1]	IIQNNT[3]	JISHFTC[3]	QCOSD[3]	ZCOS[1]
BIXOR[4]	DCMPLX[1]	HBITS[1]	IISHFT[3]	JISIGN[3]	QCOSH[3]	ZEXP[1]
BJTEST[3]	DCONJG[1]	HBSET[1]	IISHFTC[3]	JIXOR[3]	QDIM[3]	ZLOG[1]
BMVBITS[4]	DCOS	HDIM[1]	IISIGN[3]	JMAX0[3]	QEXP[3]	ZSIN[1]
BMOD[4]	DCOSD[3]	HIAND[1]	IIXOR[3]	JMAX1[3]	QEXT[3]	ZSQRT[1]
BNOT[4]	DCOSH	HIEOR[1]	IJQNINT[3]	JMIN0[3]	QEXTD[3]	ZTAN[1]
BSHFT[4]	DDIM	HIOR[1]	IMAG	JMIN1[3]	QFLOAT[3]	

1. Extensión HP al estándar ANSI
2. Estándar MIL-STD-1753
3. Extensión compatible con el estándar ANSI
4. Extensión disponible con la opción de compilación `+800`
5. Extensión disponible con la opción de compilación `+apollo`
6. Extensión correspondiente al Fortran 90