Noțiuni de teorie începând cu LUCRAREA nr. 7

Utilizarea unităților logice

Instrucțiunile de intrare (citiri) și de ieșire (scrieri) se realizează prin unități logice. Unitatea logică implicită (marcată cu valoarea `*` în instrucțiunile care necesită specificarea) este consola, adică ansamblul alcătuit din tastatură și afișaj (ecranul monitorului) – la intrări se consideră tastatura, iar la ieșiri afișajul. Unitățile logice care trebuie specificate în mod explicit sunt fișierele, respectiv perifericele (imprimanta, unitate cu bandă magnetică etc.), acestora fiind alocată o valoare numerică întreagă. Indicarea unității logice la care se referă o instrucțiune de intrare / ieșire se face prin această valoare numerică. Unele valori au și unități logice predefinite în variantele Fortran mai vechi, ca de exemplu 1, 2, 3 și 4 pentru fișiere numite `FOR00`*n*`.DAT` (unde în locul caracterului *n* va apare în denumirea fișierului cifra corespunzătoare de la 1 la 4), sau 5 pentru dispozitive de intrare (cititor de cartele, tastatură etc.), respectiv 6 pentru dispozitive de ieșire (imprimantă, afișaj etc.). Alocarea acestor referințe (numere) unităților logice se poate face în mod explicit prin instrucțiunea `OPEN`. Sintaxa acestei instrucțiuni executabile este următoarea:

`OPEN (`*parametru*`[,` *parametru*`]… )`

unde *parametru* poate fi un *cuvânt_cheie*, sau de forma *cuvânt_cheie*=*valoare* (fiecare *parametru* poate să fie specificat doar o dată în cadrul listei din paranteză).

Tabel cu parametri din instrucţiunea `OPEN` – cele albastre nu există în compilatorul G95 (în ordine alfabetică, fără cele specifice QuickWIN din Intel Fortran):

cuvânt_cheie	*valoare*	Explicație	Valoare implicită

`ACCESS=`	`"SEQUENTIAL"`
	`"DIRECT"`
	`"APPEND"`
	`"KEYED"`
	`"STREAM"`
		Stabilirea modului de accesare a unității logice:
		- secvențial,
		- direct,
		- adăugare,
		- pe bază de câmpuri cheie.
			`"SEQUENTIAL"` (secvențial, rând după rând).

`ACTION=`
`MODE=`
	`"READ"`
	`"WRITE"`
	`"READWRITE"`
		Modul în care se poate folosi unitatea logică:
		- citire (fără scriere),
		- scriere (fără citire),
		- citire și scriere.
			`"READWRITE"` (citire și scriere).

`ASYNCHRONOUS=`	`"NO"` `"YES"`	Permite specificarea modului intrare/ieşire asincron.	`"NO"` (operații de I/E în mod sincron).
`ASSOCIATEVARIABLE=`	*număr*	Numărul înregistrării următoare în cazul accesului direct (*număr* fiind un întreg pozitiv).	Nu există valoare asociată în mod implicit.
`BLANK=`	`"NULL"` `"ZERO"`	Interpretarea spaţiilor: - nu se transformă (blank), - se transformă în cifre `0` în cazul numerelor.	`"NULL"` (nu se transformă).
`BLOCKSIZE=`	*număr*	Mărimea unui bloc în memoria de tampon la intrare/ieşire (*număr* fiind un întreg pozitiv).	Valoarea este alocată de către sistem.
`BUFFERCOUNT=`	*număr*	Numărul de tampoane de intrare/ieşire (*număr* fiind un întreg pozitiv).	Numărul este alocat de către sistemul de operare.
`BUFFERED=`	`"NO"` `"YES"`	Permite specificarea comportamentului bibliotecilor de rulare (run-	`"NO"` (direct, fără utilizarea unei zone tampon

		time library) după operații de scriere efectuate pe unitatea logică: - fără tampon suplimentar, - cu tampon suplimentar.	suplimentare). Atenție, sistemul de operare va folosi memorie tampon la operații de scriere!
CARRIAGECONTROL=	"FORTRAN" "LIST" "NONE"	Controlul (interpretarea) returului de car (caracterul generat la apăsarea tastei <*Enter*>): - Fortran (primul caracter va fi interpretat și "consumat"), - listă (returul de car este ultimul caracter), - niciuna.	"FORTRAN" în cazul formei "FORMATTED", şi "NONE" în cazul formei "UNFORMATTED".
CONVERT=	"NATIVE" "SWAP" "LITTLE_ENDIAN" "BIG_ENDIAN" "CRAY" "FDX" "FGX" "IBM" "VAXD" "VAXG"	Permite specificarea unui format numeric (de conversie / interpretare) pentru datele neformatate: - nativ (fără convertire), - comută (între formatele LITTLE_ENDIAN și BIG_ENDIAN), - diverse alte formate…	"NATIVE" (nu se convertesc datele).
DEFAULTFILE=	*expresie_caracter*	Setarea unei specificații de fișier implicite.	Nu este.
DELIM=	"NONE" "APOSTROPHE" "QUOTE"	Specificarea caracterului delimitator (pentru constante tip CHARACTER) la operații de I/E: - fără delimitator, - caracterul apostrof, - caracterul ghilimele.	"NONE" (fără delimitator).
DISPOSE= DISP=	"SAVE" "KEEP" "PRINT" "DELETE"	Starea unității logice (de regulă a fişierului) la închidere: - salvează, - păstrează (temporar), - tipărește, - șterge.	"SAVE" (salvarea conținutului unității logice).
ERR=	*etichetă*	Eticheta instrucțiunii la care se va sări în caz de eroare la deschiderea unității logice.	Nu există, nu se face salt.
EXTENDSIZE=	*număr*	Mărimea spațiului de memorare alocat pentru fişier (*număr* fiind un întreg pozitiv).	Dată de sistemul de operare sau de volum (partiţie).
FILE= NAME=	*expresie_caracter*	Specificarea fișierului care va fi utilizat ca unitate logică. Specificatorul de fișier este un șir de caractere, deci se delimitează cu apostrof sau ghilimele în cazul în care se citează, iar în cazul în care	Depinde de unitatea logică şi de sistemul de operare.

		este conținut de o entitate de tip CHARACTER, se poate specifica numele acelei entități.	
FORM=	"FORMATTED" "UNFORMATTED" "BINARY"	Formatul unității logice (a fișierului) accesat: - formatat, - neformatat, - binar.	Depinde de valoarea cuvântului cheie ACCESS. Dacă este "DIRECT" sau "KEYED" atunci va fi considerat "FORMATTED", altfel va fi considerat "UNFORMATTED".
INITIALSIZE=	*număr*	Spaţiul în memorie alocat iniţial pentru fişier (*număr* fiind un întreg pozitiv).	Nu se alocă.
IOSTAT=	*variabilă*	Returnează o valoare scalară de tip INTEGER în *variabilă*, prin care se poate vedea succesul (sau insuccesul) accesării unității logice. În cazul deschiderii unității logice valoarea va fi 0.	Nu este implicit.
KEY=	(*cheia1*[,*cheia2*]...)	Câmpurile de cheie (în ordinea priorității acestora) pentru fişier indexat.	Nu este implicit.
MAXREC=	*număr*	Numărul maxim de înregistrări care se pot transfera în accesul direct (*număr* fiind un întreg pozitiv).	Nu există maxim.
NOSPANBLOCKS		Înregistrările nu traversează blocurile.	Înregistrările pot traversa blocurile.
ORGANIZATION=	"SEQUENTIAL" "RELATIVE" "INDEXED"	Structura (modul de organizare a) unității logice (fișierului): - secvențială, - relativă, - indexată.	"SEQUENTIAL" (organizare secvențială).
PAD=	"YES" "NO"	Specifică dacă la introducerea unei înregistrări se completează cu spații (caractere blank), când formatul necesită mai multe poziții decât are valoarea introdusă, sau dacă nu se completează.	"YES" (se completează cu spații când e necesar).
POSITION=	"ASIS" "REWIND" "APPEND"	Specifică poziționarea într-un fișier: - așa cum este, - revenire la început, - adăugare la sfârșit.	"ASIS" (poziția curentă).
READONLY		Protecţie la scriere (în cazul specificării acestui	Neprotejat la scriere.

		parametru, fișierul nu poate fi șters în momentul închiderii).	
RECL= RECORDSIZE=	*număr*	Lungimea înregistrării din unitatea logică în cazul accesului direct, sau lungimea maximă în cazul accesului secvențial (*număr* este un întreg pozitiv).	Depinde de valoarea specificată la cuvintele cheie: STATUS, ORGANIZATION și RECORDTYPE.
RECORDTYPE=	"FIXED" "VARIABLE" "SEGMENTED" "STREAM" "STREAM_LF" "STREAM_CR"	Structura înregistrării: - fixă (toate înregistrările vor avea lungime identică), - variabilă, - segmentată, - șir, - șir cu salt la rând nou, - șir cu retur de car.	Depinde de valorile cuvintelor cheie: ACCESS, FORM.
SHARE=	"COMPAT" "DENYNONE" "DENYWR" "DENYRD" "DENYRW"	Controlează cum pot accesa alte procese unitatea logică în mod simultan, într-o rețea: - compatibil, - fără restricții, - fără scriere, - fără citire, - fără citire și scriere.	"DENYNONE" (toate procesele, fără restricții).
SHARED		Acces partajat la unitatea logică (la fișier).	Accesul nu este partajat.
STATUS= TYPE=	"OLD" "NEW" "REPLACE" "SCRATCH" "UNKNOWN"	Starea unității logice (a fișierului) la deschidere: - existent (dacă nu există, se obține eroare), - nou (dacă există deja, generează eroare), - suprascierea unui fișier, - temporar (se șterge după închidere), - necunoscut (se deschide dacă există, se crează dacă nu există).	"UNKNOWN" (se deschide dacă există, se crează dacă nu există).
UNIT=	*număr*	Numărul unității logice (asociat fișierului, sau dispozitivului dorit) care se accesează (*număr* este un întreg pozitiv).	Nu este implicit (dacă este primul parametru din paranteză, numărul unității logice se poate specifica și fără cuvântul cheie UNIT=).
USEROPEN=	*nume*	Opțiune pentru un program utilizator.	Nu există opțiune.

Deconectarea unității logice (în cazul fișierelor înseamnă închiderea lor) se poate specifica prin instrucțiunea executabilă CLOSE, a cărei sintaxă este următoarea:

CLOSE (*parametru*[, *parametru*]…)

unde *parametru* este de forma *cuvânt_cheie*=*valoare* (fiecare *parametru* poate să fie specificat doar o dată în cadrul listei din paranteză).

Instrucțiunea CLOSE va determina și înregistrarea (scrierea) marcajului de sfârșit al fișierului (<EOF>, adică *End_Of_File*) în momentul deconectării (închiderii fișierului).

Tabel cu parametri din instrucţiunea CLOSE (în ordine alfabetică, cele albastre nu există în compilatorul G95):

cuvânt_cheie	*valoare*	Explicație	Valoare implicită
DISPOSE= DISP= STATUS=	"SAVE" "KEEP" "PRINT" "DELETE" "PRINT/DELETE" "SUBMIT" "SUBMIT/DELETE"	Starea unității logice (de regulă a fișierului) la închidere: - salvează, - păstrează, - tipărește, - șterge, - tipărește și apoi șterge, - invocă un proces care să execute fișierul, - invocă un proces care să execute și apoi șterge fișierul.	"SAVE" (salvarea conținutului unității logice).
ERR=	*etichetă*	Eticheta instrucțiunii la care se va sări în caz de eroare la deconectarea unității logice (*etichetă* este un număr întreg pozitiv).	Nu există, nu se face salt.
IOMSG=	*variabilă*	Returnează printr-un mesaj conținutul din *variabilă* (care este un scalar de tip CHARACTER).	Nu este implicit.
IOSTAT=	*variabilă*	Returnează o valoare de tip INTEGER scalar în *variabilă*, prin care se poate vedea succesul (sau insuccesul) deconectării unității logice. În cazul închiderii corecte a unității logice, valoarea variabilei va fi 0.	Nu este implicit.
UNIT=	*număr*	Numărul unității logice (asociat fișierului, sau dispozitivului dorit) care se deconectează (*număr* este un întreg pozitiv).	Nu este implicit (dacă este primul parametru din paranteză, numărul unității logice se poate specifica și fără cuvântul cheie UNIT=).

Atenție: Un fișier deschis cu specificarea "SCRATCH" nu poate fi salvat, tipărit (afișat), sau trimis unui proces ("SUBMIT"), o asemenea tentativă va genera eroare la rulare, iar dacă la deschidere s-a specificat "READONLY", atunci fișierul nu va putea fi șters la deconectare (închidere). Un fișier utilizat doar pentru citire nu trebuie neapărat închis, însă unul în care s-a modificat conținutul (s-a scris) trebuie deconectat (închis) folosind instrucțiunea CLOSE, altfel la terminarea rulării programului poate rămâne blocat și cu conținut inaccesibil. Scrierea efectuându-se printr-o memorie tampon, dacă aceasta nu a fost golită în mod expres (prin efectul instrucțiunii CLOSE), atunci nu e sigur că s-au transferat toate înregistrările, iar prin terminarea rulării programului nu va mi fi cine să gestioneze conținutul memoriei tampon (rezultând umplerea memoriei calculatorului cu date inutile).

Exemplu:	Explicații:

`OPEN(3,FILE="TEST.DAT",STATUS="OLD")`
`READ(3,*)n,m`
`CLOSE(3)`
	Deschiderea fișierului existent TEST.DAT, asociat cu unitatea logică numărul 3, urmată de citirea valorilor variabilelor N și M din acest fișier și deconectarea unității logice (închiderea fișierului).

`DIMENSION A(10,10)`
`CHARACTER(12) nume`
 	
`PRINT *,"nume fisier de date: "`
`3 READ(*,"(A)")nume`
`OPEN(1,FILE=nume,STATUS="OLD",ERR=9)`
 	
`! se citeste numărul de linii pt. A`
`READ(1,*) nl`
`! se citeste numărul de coloane pt. A`
`READ(1,*) nc`
`! citire elementele, pe câte un rând:`
`DO i=1,nl`
`READ(1,*)(A(i,j),j=1,nc)`
`ENDDO`
`…`
`OPEN(2,FILE="R.DAT",STATUS="UNKNOWN")`
 	
`! scriere titlu în fisierul R.DAT`
`WRITE(2,*)"tabloul A:"`
`! scrierea elementelor, câte un rând:`
`DO i=1,nl`
`WRITE(2,*)(A(i,j)," ",j=1,nc)`
`ENDDO`
`CLOSE(2)`
`…`
 	
`9 PRINT *,"nu s-a gasit fisierul!"`
`GOTO 3`
`…`
	Declararea unui tablou cu 10x10=100 poziții și a entității NUME cu 12 poziții (caractere). Atenție la `CHARACTER`, litera `C` în prima coloană va marca comentariu!
	Citirea denumirii fișierului în variabila NUME. Deschiderea fișierului (existent) prin asociere cu unitatea logică numărul 1, din care se vor citi datele (a se vedea comentariile din coloana alăturată marcate cu semnul de exclamare). Dacă fișierul nu există, se va sări la instrucțiunea cu eticheta 9.
	
	Utilizarea unui ciclu implicit (J=1,NC) în cadrul unui ciclu explicit (I=1,NL), pentru citirea elementelor dintr-o matrice.
	
	Deschiderea fișierului R.DAT prin asociere cu unitatea logică numărul 2 (dacă fișierul nu există, se va crea, iar dacă există, se va deschide și se va suprascrie conținutul).
	Scrierea elementelor din matricea A, pe rânduri, cu spațiu (" ") după fiecare element.
	
	Închiderea fișierului R.DAT (deconectarea unității logice numărul 2). Unitatea logică numărul 1 nu a suferit modificări și va fi automat deconectată în momentul terminării rulării programului.
	În cazul negăsirii fișierului cu date, după afișarea mesajului specificat se va încerca recitirea denumirii acestuia (sărind la instrucțiunea cu eticheta 3).

Există și alte instrucțiuni suplimentare pentru tratarea fișierelor (cele colorate nefiind recunoscute de către compilatorul G95), cum ar fi:

Sintaxa instrucțiunii:	Explicații:
`UNLOCK([UNIT=]`*u*`[,ERR=`*e*`])` sau `UNLOCK` *u*	Deblocarea unui fișier (după asocierea cu o unitate logică prin instrucțiunea `OPEN`), unde *u* reprezintă numărul unității logice, iar *e* o etichetă.
REWIND([`UNIT=`]*u*[,`ERR=`*e*]) sau `REWIND` *u*	Repoziționarea la începutul fișierului asociat (în prealabil prin instrucțiunea `OPEN`) cu unitatea logică numărul *u*.
`REWRITE([UNIT=]`*u*`[,[FMT=]`*f*`][,ERR=`*e*`])` *listă*	Rescrierea unei înregistrări în fișierul asociat cu unitatea logică *u* (printr-o instrucțiune `OPEN` prealabilă), cu specificația de format *f* (dacă se specifică).

`ENDFILE([UNIT=]`*u*`[,ERR=`*e*`])` sau `ENDFILE` *u*	Scrierea marcajului de sfârșit în fișierul asociat cu unitatea logică *u* (accesat printr-o instrucțiune `OPEN` prealabilă).

Descriptori de format

Descriptorii de format sunt ca niște șabloane aplicate pe datele de intrare sau de ieșire. Utilizarea lor se face de regulă prin specificația de format, a cărei sintaxă este următoare:

etichetă `FORMAT(`*listă_descriptori*`)`

însă, descriptorii pot să apară și sub formă citată în cadrul instrucțiunilor de citire sau de scriere.

Există două categorii de descriptori: pentru editare date, respectiv pentru controlul formatării. Vor fi prezentate în cele ce urmează în tabele separate, cu exemple, folosind următoarele notații:

n – numărul de bucăți;
w – lungimea descriptorului (numărul total de poziții din câmpul respectiv);
m – numărul minim de poziții solicitate (din numărul total), are efect doar la ieșire;
d – numărul pozițiilor pentru partea zecimală (din numărul total);
e – numărul pozițiilor pentru exponent (din numărul total);
c – caracter, respectiv [*c*...] alte caractere opționale;
□ – spațiu (caracterul blank) în exemple.

Tabel cu descriptorii pentru editarea datelor (în ordine alfabetică):

Sintaxa:	Destinația:	Exemple și explicații:		
[*n*]`A`[*w*]	Date alfanumerice (`CHARACTER`)	Intrare: `ABC_D` `ABC_D` `ABC_D`	Format: `A5` `A5` `A5`	Tip entitate: Valoare: `CHARACTER(1):` D `CHARACTER(3):` C_D `CHARACTER(6):` ABC_D□
		Valoare: ABC ABCDE ABCDEFG	Format: `A5` `A5` `A5`	Ieșire (5 poziții): □□ABC ABCDE ABCDE

[*n*]`B`*w*[.*m*]	Date numerice binare	Intrare:
		`1001`
		`1001`
		`1001`
			Format:
			`B4`
			`B2`
			`2B2`
				Valoare (în forma decimală):
				9 (toate cele 4 poziții citite)
				2 (doar primele 2 poziții citite)
				2 și 1 (2 valori distincte)

		Valoare:
		13
		0
		0
			Format:
			`B5`
			`B2`
			`B2.2`
				Ieșire:
				□`1101`
				□`0`
				`00`

		Dacă *w*=0, la ieșire se vor utiliza atâtea poziții, câte sunt necesare afișării valorii (la intrare nu se admite *w*=0).		

[*n*]`D`*w*.*d*	Date numerice în dublă precizie: `REAL(8)` adică `DOUBLE PRECISION`, sau `COMPLEX(8)`, adică `DOUBLE COMPLEX`	Intrare:
		`123.456E3`
		`12345678`
		`123.45678`
			Format:
			`D9.3`
			`D6.2`
			`D7.3`
				Valoare (dublă precizie):
				123456.0D+0
				1234.56D+0
				123.456D+0

		După cum se poate observa, se citesc *w* poziții din intrare, dintre acestea *d* poziții pentru partea zecimală (de la separatorul zecimal spre dreapta – dacă la intrare nu este separator zecimal, atunci partea zecimală va rezulta considerând *d* poziții din capătul celor *w* citite). Marcajul "`D+0`" din capăt indică doar că valorile se vor obține în dublă precizie.		
		Valoare: 123456.789	Format: `D11.2`	Ieșire: □□□`0.12D+06`

		0.0363 D10.3 □0.363D-01 -0.5555 D10.3 -0.556D+00 Afișarea va rezulta pe *w* poziții, din care *d* poziții pentru partea zecimală, însă trebuie avut în vedere că 1 poziție se va consuma pentru semnul valorii, încă 1 pentru separatorul zecimal (punctul), 1 poziție pentru litera descriptorului (D), ultimele 3 poziții pentru semnul și valoare exponentului. Dacă luăm în considerare că prima cifră semnificativă va fi prima zecimală, rezultă că este recomandabil ca *w*−*d* > 6. Dacă nu se respectă această condiție, rezultă depășire de format (pe cele *w* poziții se vor afișa asteriscuri).
[*n*]E*w*.*d*[E*e*]	Date numerice în format exponențial (REAL sau COMPLEX)	Intrare: Format: Valoare: □□123.45□□ E10.2 123.45 123456789 E9.3 123456.789 123.456D3 E9.3 123456.0 (simplă precizie!) Ca și în cazul descriptorului precedent, se citesc *w* poziții din intrare, dintre acestea *d* poziții pentru partea zecimală (de la separatorul zecimal spre dreapta – dacă la intrare nu este separator zecimal, partea zecimală va rezulta considerând *d* poziții din capătul celor *w* citite). În cazul citirii unor valori tip dublă precizie cu acest descriptor (sau cu altele utilizabile, exceptând D) se va obține o valoare convertită în simplă precizie.
		Valoare: Format: Ieșire: 123456.789 E11.5 0.12345E+06 -0.5555 E12.3E3 □-0.556E+000 0.0363 E5.2 ***** (depășire de format!) Afișarea va rezulta pe *w* poziții, din care *d* poziții pentru partea zecimală, însă trebuie avut în vedere că 1 poziție se va consuma pentru semnul valorii, încă 1 pentru separatorul zecimal (punctul), 1 poziție pentru litera descriptorului (E), ultimele 3 poziții pentru semnul și valoare exponentului. Dacă luăm în considerare că prima cifră semnificativă va fi prima zecimală, rezultă că este recomandabil ca *w*−*d* > 6 [+(*e*−2)] (unde *e* este numărul cifrelor exponentului). Dacă nu se respectă această condiție, rezultă depășire de format (pe cele *w* poziții se vor afișa asteriscuri).
[*n*]EN*w*.*d*[E*e*]	Date numerice în format exponențial ”ingineresc” (REAL sau COMPLEX)	Intrare: Format: Valoare: 123.45E+03 EN10.2 12345.0 -12345678 EN9.3 -12345.678 123.456D3 EN9.3 123456.0 (simplă precizie!)
		Valoare: Format: Ieșire: 123456.789 EN11.2 □123.46E+03 -0.5555 EN7.1 ******* (depășire de format!) 0.0363 EN12.3 □363.000E-04 La afișare punctul zecimal va fi după primele 3 cifre.
[*n*]ES*w*.*d*[E*e*]	Date numerice în format exponențial ”științific” (REAL sau COMPLEX)	Intrare: Format: Valoare: □□□1.234E+03 ES12.3 1234.0 -10.234E-03 ES11.3 -0.010234
		Valoare: Format: Ieșire: 123456.789 ES11.2 □□□1.23E+05 -0.5555 ES10.3 -5.555E-01 0.0363 ES12.3 □□□3.630E-02 La afișare punctul zecimal va fi după prima cifră semnificativă.

<table>
<tr><td>[n]Fw.d</td><td>Date numerice (REAL, F vine de la ”Float”)</td><td>
<u>Intrare</u>: <u>Format</u>: <u>Valoare</u>:
12345678 F8.5 123.45678
-12345678 F8.2 -1234.56
24.77E+2 F8.2 2477.0
<u>Valoare</u>: <u>Format</u>: <u>Ieșire</u>:
2.3547188 F8.5 □2.35472
325.03 F5.2 ***** (depășire de format!)
-0.2 F5.2 -0.20
</td></tr>
<tr><td>[n]Gw.d[Ee]</td><td>Date de tip intrinsec (G vine de la ”Generic”)</td><td>
<u>Intrare</u>: <u>Format</u>: <u>Valoare</u>:
-0.05566 G10.3 -0.05566
123456 G10.3 123.456
123456.789 G10.3 123456.79
<u>Valoare</u>: <u>Format</u>: <u>Ieșire</u>:
-45.66 G11.3 □-4.566E+01
123456 G10.3 □□□□123456
123456.78 G10.3 □0.123E+06
Observații: Descriptorul G se poate utiliza pentru oricare dintre valorile de tip intrinsec. În cazul în care se specifică valoarea 0 pentru w, valoarea efectivă a acestuia va fi aleasă de către procesor (în asemenea situații se pot specifica doar variantele G0 sau G0.d). Dacă w este diferit de 0, atunci obligatoriu trebuie specificată și valoarea pentru d. În cazul valorilor de tip INTEGER, CHARACTER și LOGICAL va fi ignorată valoarea specificată prin d, descriptorul se va comporta ca cel corespunzător acestor valori (I, A și L).
</td></tr>
<tr><td>wHc[c...]</td><td>Constante Hollerith (CHARACTER)</td><td>
<u>Intrare</u>: Nu se recomandă utilizarea, pentru că se poate modifica conținutul constantei (compilatorul G95 va da mesaj de eroare!)
<u>Format</u>: <u>Ieșire</u>:
10H#’-’abc”3 #’-’abc”3
21Hconstanta ”Hollerith” constanta□”Hollerith”
Observații: Deși inițial aceste constante au fost definite cu un conținut de până la 2000 de caractere, valoarea w poate fi între 1 și 32767 ($2^{15}-1$) pe sistemele cu arhitectura pe 32 de biți, respectiv între 1 și 2147483647 ($2^{31}-1$) pe platformele pe 64 de biți.
</td></tr>
<tr><td>[n]Iw[.m]</td><td>Date numerice întregi (INTEGER)</td><td>
<u>Intrare</u>: <u>Format</u>: <u>Valoare</u>:
-1234 I4 -123
□□□123 I6 123
1234.6 I6 Eroare! (nu este INTEGER)
<u>Valoare</u>: <u>Format</u>: <u>Ieșire</u>:
0 I3 □□0
0 I3.0 □□□
1 I3.2 □01
-123 I3 *** (depășire de format!)
1.2 I4 Eroare! (nu este INTEGER)
</td></tr>
<tr><td>[n]Lw</td><td>Date logice</td><td>
<u>Intrare</u> – se acceptă valorile logice scrise sub următoarele forme, inclusiv cu caractere mici (nu doar cu majuscule):
.TRUE. sau .T sau T sau dacă primele caractere din intrare sunt .T sau T (pentru adevărat), respectiv .FALSE. sau .F sau F sau sau dacă primele caractere din intrare sunt .F sau F sau conținutul este din spațiu/spații (pentru fals).
<u>Valoare</u>: <u>Format</u>: <u>Ieșire</u>:
.TRUE. L7 □□□□□□T
.FALSE. L1 F
</td></tr>
</table>

		□□□ L3 □□F La ieșire se va obține doar 1 caracter (T sau F) indiferent de lungimea *w* specificată.
[*n*]O*w*[.*m*]	Date numerice octale întregi (cu baza în 8)	Intrare: / Format: / Valoare (decimală): 1111 O2 9 1111 O4 585 □11□ O4 9 191 O3 Eroare! (9 nu este cifră octală) 12 O0 Eroare! (*w* trebuie să fie pozitiv) Valoare (decimală): / Format: / Ieșire: 11 O6.4 □□0013 -11 O6 ****** (depășire de format!) -11 O12 □37777777765 1.5 O11 □7760000000 81 O0 121 Dacă *w*=0, la ieșire se vor utiliza atâtea poziții, câte sunt necesare afișării valorii (la intrare nu se admite *w*=0).
[*n*]Z*w*[.*m*]	Date numerice hexa întregi (cu baza în 16)	Intrare: / Format: / Valoare (decimală): A2F Z3 2607 −A2F□ Z5 -2607 3.A2F Z5 Eroare! (punct zecimal invalid) Valoare (decimală): / Format: / Ieșire: 3033 Z5 □□BD9 16 Z5.4 □0010 -10 Z8 FFFFFFF6 1.1 Z8 3F8CCCCD 2.5 Z0 40200000 Dacă *w*=0, la ieșire se vor utiliza atâtea poziții, câte sunt necesare afișării valorii (la intrare nu se admite *w*=0).
'*c*[*c*...]' sau "*c*[*c*...]"	Constante alfanumerice citate (CHARACTER)	Intrare: nu se aplică (nu se pot folosi pentru intrări). Format: / Ieșire: 'aBc''DD:' (apostrof dublat în interior) aBc'DD: 'aBc"DD:' (ghilimele în conținut) aBc"DD: "abcD'#" abcD'# "ab'cD'#" (citat în citat!) Eroare!

Tabel cu descriptorii de control:

Sintaxa:	Semnificația:	Explicații și exemple:
BN BZ	BLANK NULL BLANK ZERO	BN va avea ca efect ignorarea spațiilor din câmpurile numerice. BZ va avea ca efect "înlocuirea" spațiilor din câmpurile numerice cu cifre 0. Intrare: / Format: / Valoare: □1□□ BN,I4 1 □1□□ BZ,I4 100 1□23 BZ,I4 1023
*k*P	Power *k* este un factor de scalare, cu valoare din intervalul [-128, +127]	Permite interpretarea valorilor numerice cu zecimale, cu factorul de scară *k* în cazul descriptorilor D, E, F și G, dacă aceste valori nu conțin în mod explicit exponent. La intrări, o valoare pozitivă *k* va avea efectul mutării separatorului zecimal spre stânga, iar o valoare negativă spre dreapta (la ieșiri efectul va fi invers). Descriptorul P nu trebuie separat neapărat cu virgulă de descriptorul la care se referă, însă trebuie să-l preceadă.

		De exemplu, următoarele specificații vor avea același efect, factorul de scară fiind asociat cu primul descriptor pentru numere reale care îl urmează în listă (`E10.3`):
		`(2P,I4,E10.3,F8.2)`
		`(I4,2P,E10.3,F8.2)`
		`(I4,2PE10.3,F8.2)`
		Intrare: Format: Valoare:
		□□□37.614□ 3PE10.5 0.037614
		□□□37.614□ -3PE10.5 37614.0
		123.45 2PF8.3 1.2345
		123.45 -2P,F8.3 12345.0
		Valoare: Format: Ieșire:
		-170.139 1P,E10.3 -1.701E+02
		-170.139 -1PE10.3 □-0.02E+04

Q	Quantity (cantitate)	Returnează numărul de caractere (poziții) rămase netratate din intrare (deși este acceptat de regulă din motive de compatibilitate, nu face parte din variantele noi ale limbajului Fortran și compilatorul G95 îl va semnala ca eroare).
		Cod sursă:
		`READ(*,'(Q)')nr`
		`PRINT 2,nr`
		`READ(*,'(Q)')nr`
		`PRINT 2,nr`
		`2 FORMAT('Buc: ',I2)`
		`END`
		Input:
		abcdefg
		Afișaj:
		Buc: 7
		Input:
		55aa
		Afișaj:
		Buc: 4

S SP SS	Sign Sign Positive Sign Suppress	SP va determina afișarea semnului + în fața valorilor pozitive, iar SS va inhiba afișarea acestuia. S funcționează ca un comutator între SP și SS.

T*n*
TLn
TR*n*
	Tab
	Tab Left
	Tab Right
	n – poziția de tabulare
		Prin descriptorul T se precizează poziția *n* dintr-un rând, de la care se dorește citirea (sau la care se dorește scrierea).
		Considerând că de la tastatură se va introduce șirul:
		123456789ABC
		care se va citi cu secvența:
		`CHARACTER(3) C1,C2`
		`READ(*,5) NR,C1,C2`
		`5 FORMAT(T7,I3,T1,A3,T10,A3)`
		vor rezulta valorile: NR=789; C1="123" și C2="ABC".
		TR*n* permite specificarea poziției a *n*-a spre dreapta, de la poziția curentă, iar TL*n* spre stânga (*n* fiind un număr pozitiv). În cazul utilizării TL, dacă *n* este mai mare sau egal cu poziția curentă, atunci poziționarea se va face pe primul caracter din rând.

[*n*]X	Determină saltul peste *n* poziții din rândul curent	La intrare va determina ignorarea a *n* poziții, iar la ieșire va avea ca efect tipărirea de *n* spații (dacă apare la sfârșitul listei de descriptori, atunci nu are efect). În exemplu efectul este evidențiat prin marcarea cu □ la afișare):
		Cod sursă:
		`PRINT 4`
		`READ 3,nr`
		`PRINT 4,nr`
		`3 FORMAT(2X,I2)`
		`4 FORMAT("numar:",1X,I2)`
		Afișaj:
		numar:
		Input:
		1234
		Afișaj:
		numar:□34

$ \	Suprimă saltul la rând nou (suprimă <*LF*>).	Va determina rămânerea cursorului pe ultima poziție curentă (<*LF*> este prescurtarea de la *Line_Feed*).

		Varianta `$` este mai nou introdusă, însă nu face parte din standard, iar varianta `\` este cea ”clasică” (compilatorul G95 acceptă ambele). Oricare variantă s-ar utiliza, descriptorul trebuie să fie ultima din lista din care face parte.
		Cod sursă:
		`PRINT 5,”nr:”`
		`READ *,nr`
		`PRINT 4,nr`
		`5 FORMAT(A,$)`
		`4 FORMAT(”numar:”,1X,I2)`
		Afișaj+Input (12):
		`nr:12`
		Afișaj:
		`numar:□12`

[*n*]/	Induce *n* salturi la rând nou (induce *n* bucăți <*LF*>)	Se poate utiliza și fără *n*, de exemplu (`3/`) este echivalent cu (`///`), fără a necesita virgulă de separare. În următorul exemplu va introduce un salt la rând nou înainte de a afișa ”numar”, după care se vor introduce încă 2 salturi la rând nou:
		Cod sursă:
		`PRINT 5,”nr:”`
		`READ *,nr`
		`PRINT 4,nr`
		`5 FORMAT(A,$)`
		`4 FORMAT(/”numar:”,2/,3X,I2)`
		Afișaj+Input (12):
		`nr:12`
		Afișaj:
		`□`
		`numar:`
		`□`
		`□□□12`

:	Termină controlul descriptorilor în lipsa elementelor din lista de intrare / ieșire	În următorul exemplu, în lipsa elementelor de afișat, descriptorul va determina ignorarea părții ”j2”:
		Cod sursă:
		`PRINT 1,3`
		`PRINT 2,14`
		`1 FORMAT(”i”,I2,1X,”i2”,I2)`
		`2 FORMAT(”j”,I2,:,1X,”j2”,I2)`
		Afișaj:
		`i□3i2□□`
		`j14`

Specificația de format poate să fie compusă și din expresii de șir (caracter). În următorul exemplu se arată cum s-ar putea aplica pentru N perechi de descriptori de forma `(I2,1X)`, considerând 1< N <9:

Exemplu:	Explicații:

`CHARACTER fm(10)`
 	
`INTEGER j(9)`
 	
`PRINT *,”nr. (1-9): ”`
`READ(*,*)n`
`k=48+n`
 	
`fm=”(”//ACHAR(k)//”(i2,1x))”`
 	
`PRINT *,”cele ”,n,” valori: ”`
`READ *,(j(i),i=1,n)`
 	
`WRITE(*,fm)(j(i),i=1,n)`
 	
`END`
	Se declară șirul FM cu 10 poziții (se va utiliza ca specificație de format).
	Variabila J va avea 9 poziții (va fi un vector) și va conține valorile care se vor afișa cu descriptori de tipul `I2`.
	Se afișează șirul citat și în urma citirii se memorează numărul introdus (de bucăți dorite) în variabila N.
	Se compune numărul poziției din tabela de coduri a cifrei corespunzătoare numărului de bucăți (din variabila N), obținând caracterul cifrei care reprezintă acea valoare.
	Se construiește prin concatenare și folosind funcția intrinsecă `ACHAR` (ce returnează caracterul de pe poziția K din tabela de coduri) un șir alfanumeric, ce se atribuie variabilei FM, care va fi descriptorul de format cu N perechi de câmpuri de tip `I2` (întreg pe două poziții) și `1X` (un spațiu) pentru cele N valori.
	Se afișează șirul de caractere citat (inclusiv valoarea lui N), după care se citesc valorile corespunzătoare celor N poziții ale vectorului J (prin ciclu implicit).
	Se afișează cele N poziții din vectorul J (tot prin ciclu implicit) folosind specificația de format memorată în variabila FM ca șir alfanumeric.