FORTRAN

- ## Uvod

FORTRAN (**FOR**mula **TRAN**slating system) je viši simbolički programski jezik, prvenstveno namijenjen za rješavanje numeričkih problema. Programski jezik FORTRAN ima točno određenu leksičku i sintaktičku strukturu te semantiku.

- ## Alfabet

Alfabet programskog jezika FORTRAN čine slijedeći skupovi znakova:

  - mala i velika slova engleskog alfabeta: **a, b, ..., z, A, B, ..., Z,**
  - znamenke: **0, 1, ..., 9,**
  - znakovi: **+ - * / = . , ( ) ' : $,**
  - praznine: **tab, blank**.

- ## Pravila pisanja programa

Program napisan u programskom jeziku FORTRAN sastoji se iz niza "rečenica", koje nazivamo naredbe ili instrukcije, pisanih prema određenim leksičkim pravilima. Naredbe se dijele u dvije osnovne klase: organizacijske i izvršne. Organizacijske naredbe definiraju neophodne parametre za izvršavanje programa (vrsta i veličina korištenih varijabli, organizacija podataka, ...). Izvršne naredbe određuju način izvršavanja programa. Naredbe programa u programskom jeziku FORTRAN pišu se svaka u posebnom retku koji se može nastavljati. Svaki je redak podijeljen u četiri polja:

  - polje komentara i obilježja (labele): kolone 1 - 5
  - polje nastavka naredbe: kolona 6
  - polje naredbe: kolone 7 - 72
  - polje za identifikaciju: kolone 73 -80

- ## Komentari

Unutar programa u programskom jeziku FORTRAN dozvoljeni su komentari. Oni ne predstavljaju naredbe, ne prevode se u strojni kod, a služe za opis programa, varijabli ili naredbi. Komentar se upisuje između 2 i 72 kolone i unutar komentara se može upisati bilo kakav tekst. Svaki komentar započinje znakom "**C**" ili "*****" u prvoj koloni. U novijim verzijama programskog jezika FORTRAN dozvoljeni su i kometari iza naredbe. Kometar koje se upisuje iza naredbe mora početi znakom "**!**".

Primjeri:

```
C Ovo je tekst komentara

* Tekst komentara može započeti znakom *

C ------------------------------------

real D1 ! Unatarnji promjer
```

- ## Obilježja (labele)

Polje obilježja (labele) služi za upisivanje obilježja instrukcije u kolone 1 do 5. Obilježje (labela) je cijeli broj, može se upisivati bilo gdje unutar polja obilježja, a služi za referenciranje na naredbu iz drugih dijelova programa.

Primjeri:

```
DO 100 i = 1, N

. . . . .

. . . . .

100 continue

write (6, 10) I

10 format(' Element polja A(', I2, ') = ', $)
```

- ## Nastavak naredbe

Polje nastavka naredbe ukazuje da naredba nije završena već se radi o nastavku naredbe iz prethodnog retka. Ukoliko se naredba nastavlja u više redova, polje nastavka naredbi svih redova kroz koje se naredba proteže mora sadržavati neki znak različit od znaka **0** ili **praznine**.

Primjer:

```
DATA ALFAKT / 2.10, 1.55, 1.38, 1.26,

B 2.40, 1.85, 1.55, 1.35,

3 2.60, 1.95, 1.70, 1.45,

> 2.64, 1.90, 1.58, 1.40 /
```

- ## Polje naredbe

Svaka se naredba u programskom jeziku FORTRAN upisuje u polje naredbe (kolone 7 do 72), a ukoliko je dulja od 66 znakova može se produžiti u slijedeći redak.

Primjeri:

```
A = A + B

x = FLOAT(i) * Pi / 180.0

y = SIN(x)
```

- ## Polje za identifikaciju

Ovo je polje (kolone 73 - 80) rezervirano za internu identifikaciju retka programa. Prilikom prevođenja simboličkog koda, ovaj se dio retka ne prevodi u strojni kod.

1. Vrste podataka

U FORTRANU se međusobno razlikuju i deklariraju slijedeće vrste podataka: cjelobrojne (INTEGER), realne (REAL), dvostruke preciznosti (DOUBLE PRECISION), kompleksne (COMPLEX), znakovne (CHARACTER) i logičke (LOGICAL).

1. *Konstante*

Konstante su eksplicitno zadani podaci koji ne mijenjaju svoju vrijednost tijekom izvođenja programa. Mogu se svrstati u tri grupe: numeričke konstante, logičke konstante i znakovne konstante. Numeričke konstante mogu biti cjelobrojne, realne i kompleksne. Logičke konstante imaju samo dvije vrijednosti .TRUE. ili .FALSE., dok su znakovne konstante nizovi alfanumeričkih znakova.

1. Cjelobrojne konstante (INTEGER)

Cjelobrojne konstante su cijeli pozitivni i negativni brojevi. Kod pozitivnih brojeva predznak nije obavezano navoditi, dok je kod negativnih obavezan. Opći oblik cjelobrojne konstante definiran je kao:

```
[ predznak] cijeli_broj
```

pri čemu je:

`predznak` Pozitivan i neobavezan označen znakom "**+**" ili negativan i obavezan označen znakom "**-**".

`cijeli_broj` Niz decimalnih znamenki čija veličina ne smije premašiti dozvoljene granice (zapis u 2 byte-a: -32768 do +32767, zapis u 4 byte-a: -2147483648 do +2147483647).

Primjeri:

Ispravno	*Neispravno*	
`0`	**`0.`**	nije dozvoljena decimalna točka
`-91`	**`-257,3`**	nije dozvoljen decimalni zarez
356	**`22345678976`**	prevelik broj
`-32768`	**`33.44`**	nije dozvoljen decimalni ostatak

2. Realne konstante (REAL)

Realne konstante su pozitivni ili negativni brojevi koji mogu (ali ne moraju) imati decimalni dio i obavezno se zapisuju s decimalnom točkom "**.**". Mogu se zapisati i u eksponencijalnom obliku. Pozitivne realne konstante ne moraju imati predznak "**+**", dok je kod negativnih predznak "**-**" obavezan. Opći oblik realne konstante definiran je kao:

```
[ predznak] cjelobrojni_dio . [ decimalni_dio] [ E|e] [ predznak_eksponenta] [ eksponent]
```

pri čemu je:

`predznak` Pozitivan i neobavezan označen znakom "**+"** ili negativan i obavezan označen znakom "**-**".

`cjelobrojni_dio` Cjelobrojni dio realne konstante (nije obavezan ukoliko postoji decimalni dio.

`.` Decimalna točka (obavezna).

`decimalni_dio` Decimalni dio realne konstante (nije obavezan ukoliko postoji cjelobrojni dio). Broj značajnih znamenaka realne konstante ovisi o zapisu (2 ili 4 byte-a). Za realne konstante zapisane u 4 byte-a ima smisla koristiti 7 značajnih znamenaka.

E | e Oznaka da je realna konstanta zapisana u eksponencijalnom obliku. Obavezno za eksponencijalni zapis realne konstante.

predznak_eksponenta Pozitivan i neobavezan označen znakom "**+**" ili negativan i obavezan označen znakom "**-**".

eksponent Cijeli broj koji predstavlja eksponent. Za realne konstante zapisane u 4 byte-a eksponent je ± 38, odnosno veličine realnih konstanti se kreću u granicama ± $0.29 \cdot 10^{-38}$ do ± $1.7 \cdot 10^{39}$.

Primjeri:

Ispravno	*Neispravno*	
0.0	**0**	nema decimalne točke
.0	**6.22E777**	prevelik eksponent
1.23	**1,23.45**	zarez nije dozvoljen
-.356E+7	**3.E5E4**	višestruki eksponent
246.357e22	**-1,2345**	zarez nije dozvoljen
273.51E-13	**256,33e28**	zarez nije dozvoljen

3. Konstante dvostruke preciznosti (DOUBLE PRECISION)

Ove se konstante razlikuju od realnih u broju značajnih znamenaka i mogu se zadavati samo u eksponencijalnom obliku. Broj značajnih znamenaka ovisi o zapisu i za konstante dvostruke preciznosti zapisane u 8 byte-a mogu imati do 17 značajnih znamenaka, dok su im granice iste kao i kod realnih konstanti tj. ± $0.29 \cdot 10^{-38}$ do ± $1.7 \cdot 10^{39}$. Opći je oblik zapisa:

[predznak] cijelobrojni_dio . [decimalni_ostatak] D|d [predznak_eksponenta] [eksponent]

pri čemu je:

D | d Oznaka da se radi o konstanti dvostruke preciznosti. Ove konstante obavezno moraju sadržavati oznaku **D** ili **d**.

Primjeri:

Ispravno	*Neispravno*	
1.33D+5	**1,256D39**	zarez nije dozvoljen
-0.3d-02	**22.3D39**	prevelik eksponent
12.234568901234567D32	**22.3E31**	obilježje eksponenta je **D**
.256D37	**22d33**	nedostaje decimalna točka

1.D0	**0.2D3.5**	eksponent mora biti cijeli broj

4. Kompleksne konstante (COMPLEX)

Kompleksna konstanta je uređeni par dviju realnih konstanti, odvojenih zarezom "**,**" i unutar okruglih zagrada "**(**" i "**)**". Prva realna konstanta predstavlja realni dio kompleksne konstante, dok druga predstavlja imaginarni dio kompleksne konstante, bez imaginarne jedinice ***i***. Pravila i ograničenja koja vrijede za realne konstante vrijede i za realni i imaginarni dio kompleksne konstante. Opći oblik kompleksne konstante je:

`( realni_dio, imaginarni_dio )`

pri čemu je:

`realni_dio` Realna konstanta.

`imaginarni_dio` Realna konstanta.

Primjeri:

Ispravno	*Neispravno*	
`(-5.2E3, 16.)`	**`(-5,3, +16.)`**	zarez nije dozvoljen
`(.33e-2, 22.e3)`	**`(22.3, 16)`**	nije dozvoljen cijeli broj
`(22., -1.567E-2)`	**`(2.3, -5.D2)`**	nije dozvoljena konstanta dvostruke preciznosti

5. Logičke konstante (LOGICAL)

Logičke konstante određuju vrijednost logičkih varijabli. Vrijednost logičke konstante može biti samo **.TRUE.** (istina) i **.FALSE.** (neistina). Točke su obavezne. Opći oblik zapisa je:

`logička_konstanta = .TRUE.` ili **`.FALSE.`**

6. Znakovne konstante (CHARACTER)

Znakovne konstante su niz alfanumeričkih znakova zatvorenih unutar para znakova " **'** ". Niz znakova može biti bilo koji skup alfanumeričkih znakova uključujući i praznine. Ukoliko je potrebno unutar znakovne konstante upisati znak " **'** ", to je moguće učiniti na način da se upiše " **''** ". Opći oblik zapisa znakovne konstante je:

`znakovna_konstanta = 'niz_znakova'`

Primjeri:

`'123!@#$%^/()abcABC?-_+='`

```
'Ovo je tekst broj 123'

'Ovako se zadaje '' (apostrof)'
```

2. *Varijable*

U programskom jeziku FORTRAN varijable predstavljaju simbolička imena memorijskih lokacija u koje se zapisuju podaci prilikom izvođenja programa. Varijable kao i konstante mogu biti cjelobrojne, realne, dvostruke preciznosti, kompleksne, logičke i znakovne. Ovisno o izvedbi FORTRAN prevodioca, ime varijable je određeno s minimalno 1 i maksimalno 6 (u starijim izvedbama) ili do 31 (u novijim izvedbama) alfanumeričkih znakova (velikih i malih slova engleskog alfabeta A - Z, a - z, znamenaka 0 - 9), uz ograničenje da **prvi znak mora biti slovo**. Minimalne i maksimalne veličine varijabli ovise o izvedbi FORTRAN prevodioca i granice su identične kao i kod odgovarajućih konstanti. FORTRAN ne razlikuje velika i mala slova ($\mathrm{A} \equiv \mathrm{a}$). Ukoliko se posebno ne odredi (ili implicitno ili eksplicitno), varijable koje počinju slovima **I**, **J**, **K**, **L**, **M**, **N** su cjelobrojne, dok su varijable koje počinju slovima **A** - **H** i **O** - **Z** realne.

3. Polja

Sekvencijalni niz memorijskih lokacija sa zajedničkim simboličkim imenom naziva se polje. Polja mogu biti istih tipova kao i konstante ili varijable (cjelobrojna, realna, dvostruke preciznosti itd.) i sva pravila koja vrijede za zadavanje konstanti i varijabli, vrijede i za polja. Svaki se element polja unutar programa promatra kao pojedinačna varijabla, a određivanjem indeksa polja određuje se element polja. Indeks polja je cjelobrojna konstanta ili varijabla. Programski jezik FORTRAN poznaje višedimenzionalna polja, a maksimalni broj dimenzija ovisi o izvedbi FORTRAN prevodioca. Za svaku dimenziju polja potreban je indeks, te tako jednodimenzionalna polja imaju 1 indeks, dvodimenzionalna 2, a *n*-dimenzionalna *n* indeksa.

2. Organizacijske naredbe

Svaki program ili potprogram napisan u programskom jeziku FORTRAN sastoji se iz niza naredbi i obavezno završava naredbom **END**. Organizacijske naredbe definiraju tipove konstanti i varijabli, početne vrijednosti, dimenzije polja, zauzeće memorije i način prihvaćanja argumenata u potprogramima. Ove se naredbe izvode samo prilikom prevođenja simboličkog u strojni kod i služe za definiranje i razvrstavanje varijabli koje će biti korištene u izvršnim naredbama. Prilikom kodiranja programa organizacijske naredbe moraju prethoditi izvršnim naredbama. Organizacijske naredbe ne smiju imati obiljažja (labele).

1. *Deklaracija polja*

Deklaracija polja je zadavanje imena polja, definiranje broja dimenzija polja (broja indeksa) i određivanje maksimalne veličine svake dimenzije polja. Opći oblik naredbe za deklaraciju polja je:

```
ime_polja (max_veličina [ ,max_veličina] )
```

pri čemu je:

`ime_polja` Ime za koje vrijede sva pravila za davanje imena varijabli.

`max_veličina` Cjelobrojna konstanta koja definira maksimalnu veličinu odgovarajućeg indeksa polja. Može biti i cjelobrojna varijabla, ali samo unutar potprograma u slučajevima kada se polje i dimenzija polja prenose u potprogram kao argumenti.

2. *Naredba DIMENSION*

Organizacijska naredba koja služi za definiranje dimenzija polja. Opći oblik naredbe je:

```
DIMENSION deklaracija_polja [ ,deklaracija_polja]
```

pri čemu je:

deklaracija_polja Deklaracija polja opisana u prethodnom odlomku.

Primjeri:

```
DIMENSION A(5), B(3,7,9,8), I1(2,5)

DIMENSION K(8)
```

pri čemu je:

A Jednodimenzionalno realno polje sa 5 elemenata.

B Četverodimenzionalno realno polje sa 1512 elemenata.

I1 Dvodimenzionalno cjelobrojno polje sa 10 elemenata.

K Jednodimenzionalno cjelobrojno polje sa 8 elemenata.

- **Dinamičko dimenzioniranje polja**

Argumenti potprograma su ili formalni ("dummy") ili aktualni. Formalni se specificiraju u kodu programa (prilikom pisanja), a aktualni se specificiraju u trenutku poziva potprograma. Kad se kontrola "predaje" potprogramu, svaki formalni argument poprima vrijednost odgovarajućeg aktualnog argumenta. Vraćanjem kontrole pozivajućoj proceduri (programskoj jedinici-cjelini) zadnja pridružena vrijednost svakom formalnom argumentu pridružuje se odgovarajućem aktualnom argumentu.

Ako se želi više puta pozivati neki potprogram i pritom kod svakog poziva koristiti različite dimenzije polja koja su formalni argumenti, tada se može koristiti tzv. prilagodljiva polja (adjustable arrays). Taj postupak zove se i "dinamičko dimenzioniranje polja". Deklaracija (naredba za dimenzioniranje) prilagodljivog polja može sadržavati cjelobrojne varijable koje su ili formalni argumenti ili su u common bloku. Dimenzije prilagodljivog polja određuje se u trenutku poziva potprograma na temelju odgovarajućih aktualnih argumenata i ne smiju biti veće od dimenzija aktualnog polja. Ovakav način dimenzioniranja polja može se koristiti samo u potprogramima.

Primjer:

```
c glavni program:

DIMENSION A1(10, 35), A2(3, 56)

SUM1 = SUM (A1, 10, 35)
```

```
SUM2 = SUM (A2, 3, 56)

SUM3 = SUM (A1, 10, 10)

END

c funkcijski potprogram:

FUNCTION SUM (A, M, N)

DIMENSION A (M, N)


SUM = 0.0

DO 10 J = 1, N

DO 10 I = 1, M

10 SUM = SUM + A (I, J)

RETURN

END
```

1. ### *Naredba COMMON*

   COMMON naredba omogućuje zajedničko korištenje dijela memorije glavnog programa i njegovih potprograma. Unutar naredbe COMMON sadržan je popis konstanti, varijabli ili polja koja čine COMMON blok, odnosno definiraju zajednički dio memorije. COMMON blok može, ali i ne mora imati ime, te se tada govori o ***imenovanim*** ili ***neimenovanom*** COMMON bloku. Opći oblik naredbe COMMON je:

   `COMMON [ /ime/] element_liste [ ,element_liste ]`

   pri čemu je:

   `ime` Ime COMMON bloka koje se zadaje unutar nakova "**/**", a upisuje se po pravilima za zadavanje imena konstanti, varijabli ili polja.

   `element_liste` Ime konstante, varijable ili polja koja se koriste kao u zajedničkom dijelu memorije. Pravila zadavanja imena su ista kako su prethodno opisana

   Primjeri:

   ```
   COMMON A, B, KI2, A2(2,7)

   COMMON /COM1/ C, D, E

   COMMON /COM2/ i, j, k, l
   ```

2. ### *Naredba IMPLICIT*

   Ova organizacijska naredba služi za implicitno deklariranje tipova varijabli i polja u programu. Tip varijable ili polja se pridružuje imenu početnim slovom varijable. Opći je oblik naredbe:

IMPLICIT tip(znak)

pri čemu je:

tip Jedan od tipova varijabli ili polja (INTEGER, REAL, COMPLEX, CHARACTER, ...)

znak Slovo ili interval.

Primjeri:

IMPLICIT INTEGER(A - Z)

IMPLICIT REAL(L, M)

3. ## *Deklaracija tipa*

Deklaracijska naredba koja definira tip svake pojedine konstante, varijable ili polja i ima opći oblik:

tip [* veličina] ime [,ime]

pri čemu je:

tip Jedan od mogućih tipova konstanti, varijabli ili polja (INTEGER, REAL, COMPLEX, DOUBLE PRECISION, CHARACTER, LOGICAL).

veličina Neobavezan parametar koji definira veličinu zapisa (byte-a). Ukoliko se ne navede eksplicitno, uzima se veličina definirana za određenu izvedbu FORTRAN prevodioca.

ime Ime konstante, varijable ili polja prema pravilima zadavanja imena.

Primjeri:

INTEGER*4 d33, Duzina

REAL*8 L1

LOGICAL Ima

CHARACTER*10 A, B*5, C(5), D(10,10)*20, E*2 /'AB'/

A niz od 10 znakova

B niz od 5 znakova

C polje od 5 nizova od kojih svaki sadrži 10 znakova

D polje od 100 nizova od kojih svaki sadrži 20 znakova

E niz od dva već definirana znaka A i B

4. *Podniz (SUBSTRING)*

Kontinuirani segment znakovne konstante, varijable ili elementa znakovnog polja koji je definiran na način:

ime (indeks_početka : indeks_kraja)

pri čemu je:

ime Ime znakovne konstante, varijable ili polja. Ime se zadaje prema već opisanom pravilu.

indeks_početka Cjelobrojna konstanta ili numerički izraz koji definira položaj prvog lijevog znaka podniza

indeks_kraja Cjelobrojna konstanta ili numerički izraz koji definira položaj zadnjeg desnog znaka podniza.

5. *Naredba SUBROUTINE*

Naredba SUBROUTINE je organizacijska naredba koja služi za definiranje potprograma. Unutar jedne programske cjeline (programske cjeline su odijeljene naredbom END), naredba SUBROUTINE mora biti prva naredba i može se pojaviti samo jednom. Potprogrami predstavljaju nezavisne cjeline unutar kojih mora postojati barem jedna izvršna naredba RETURN koja osigurava povrat na izvršne naredbe iza poziva potprograma u pozivajućoj programskoj cjelini. Argumenti se između pozivajuće programske cjeline i potprograma prenose na dva načina: preko liste argumenata koja mora biti identična po redoslijedu i tipovima u naredbama SUBROUTINE i CALL, te preko naredbe COMMON. Opći oblik naredba je slijedeći:

SUBROUTINE ime [(argument [,argument])]

pri čemu je:

ime Ime potprograma koje se zadaje po istim pravilima za zadavanje imena konstanti ili varijabli.

argument Konstanta, varijabla ili polje koje se u potprogram prenosi iz pozivajuće programske cjeline (glavnog programa ili drugih potprograma). Argumenti moraju obavezno biti poredani po istom redoslijedu i moraju biti istih tipova u pozivu potprograma i u deklaraciji potprograma. Imena argumenata ne moraju biti identična u pozivu i definiciji potprograma, jer se prilikom poziva potprograma prenose samo adrese memorijskih lokacija, a ne i njihova simbolička imena.

Primjeri:

```
SUBROUTINE SUB1

. . .

. . .

RETURN

END

SUBROUTINE Test(i, j, k, l, A, B)
```

```
. . .

. . .

RETURN

END

SUBROUTINE SUB2(i, F, G)

DIMENSION F(i)

. . .

. . .

RETURN

END
```

6. *Naredba INCLUDE*

Naredba **INCLUDE** daje nalog prevodiocu (compiler) da uključi programske naredbe iz datoteke navedene iza naredbe **INCLUDE.** Opći je oblik naredbe:

```
INCLUDE 'ime_datoteke'
```

pri čemu je:

`ime_datoteke` Ime datoteke koja se uključuje

INCLUDE naredba može se koristiti bilo gdje u programu. Datoteka koja se uključuje također može imati **INCLUDE** naredbu. Naredbe koje se uključuju moraju zadovoljiti pravila o poretku organizacijskih i izvršnih naredbi kada se kombiniraju s programom u koji se uključuju.

7. *Naredba FUNCTION*

Naredba FUNCTION je organizacijska naredba koja služi za deklaraciju potprograma kao funkcije koja svojim imenom definira varijablu čija je vrijednost rezultat obrade podataka zadanih preko argumenata. Opći oblik je:

```
[ tip_varijable] FUNCTION ime [ (argument [ ,argument] )]
```

pri čemu je:

`tip_varijable` Tip varijable koju predstavlja ime funkcijskog potprograma prema pravilima definiranja tipa varijable.

`ime` Ime funkcijskog potprograma koje istovremeno predstavlja i rezultirajuću varijablu, a zadaje se prema već opisanim pravilima definiranja imena.

`argument` Konstanta, varijabla ili polje koje se u potprogram prenosi iz pozivajuće programske cjeline.

Primjeri:

```
REAL FUNCTION FUN(x)

. . .

FUN = 2.0 * SIN(x) + 3

RETURN

END


CHARACTER FUNCTION UPPER(Znak)

. . .

UPPER = CHAR (ICHAR(Znak) - 32)

RETURN

END
```

8. ## *Naredba EXTERNAL*

Ovo je organizacijska naredba i služi za deklariranje imena potprograma koji se prenose u listi argumenata prilikom poziva potprograma. Opći je oblik definiran kao:

```
EXTERNAL ime [ ,ime]
```

pri čemu je:

ime Ime potprograma prema pravilima zadavanja imena.

Primjer:

```
EXTERNAL FUN

. . .

CALL SUB1(..., FUN, ...)

. . .

END

SUBROUTINE SUB1(..., F, ...)

. . .

RETURN

END

FUNCTION FUN(x)
```

```
. . .

RETURN

END
```

9. *Naredba DATA*

Organizacijska naredba DATA služi za inicijalizaciju početnih vrijednosti varijabli prije početka izvršavanja programa. Ovaj se postupak izvodi u trenutku prevođenja simboličkog u strojni kod programa. Opći je oblik naredbe:

```
DATA ime [ ,ime] / konstanta [ ,konstanta] /
```

pri čemu je:

ime Ime varijable ili elementa polja prema pravilima zadavanja imena.

konstanta Konstanta koja inicijalizira navedene varijable ili elemente polja. Konstante mogu biti već opisanih tipova.

Primjeri:

```
DATA A, B, I, J, NIZ*3 / 2.3, 2.3E-7, 2, 3, 'ABC' /

DIMENSION A(4), B(100)

DATA A / 1., 2., 3., 4 /, B(1), B(2), B(3) / 15., 20., 25.0 /

REAL VECT(100)

DATA / 100 * 0.0 /
```

10. Naredba END

Organizacijska naredba **END** je obavezno posljednja naredba u svakoj programskoj cjelini (glavnom programu, i svakom potprogramu tipa **SUBROUTINE** ili **FUNCTION**). Ovo je jedina organizacijska naredba koja je obavezna iza svih izvršnih naredbi.

1. Ulazno-izlazne naredbe

Ulazno/izlazne instrukcije služe za učitavanje i ispis podataka. U FORTRAN-u postoje dva osnovna načina pristupa podacima u datotekama: sekvencijalni i direktni.

1. *Sekvencijalne ulazno/izlazne naredbe*

Naredba za čitanje je **READ** a pisanje **WRITE**.

1. Naredba READ

Instrukcija READ služi za učitavanje sloga podataka iz datoteke. Opća forma instrukcije READ je:

```
READ (a, b, ERR = c, END = d) lista
```

a broj periferne jedinice s koje će se učitati podaci (cjelobrojna konstanta ili varijabla)

b neobavezan parametar koji predstavlja obilježje (labelu) instrukcije FORMAT koja definira izgled, organizaciju podataka u učitanom slogu.

ERR=c parametar u kome C predstavlja obilježje (labelu) instrukcije na koju će se skočiti ukoliko su podaci pogrešni, odnosno, ako se javi greška kod učitavanja.

END=d neobavezan parametar u kome d predstavlja obilježje (labelu instrukcije na koju će se skočiti ukoliko nema više podataka za učitavanje.

`lista` popis varijabli, polja i elemenata polja u koja učitavamo podatke.

U FORTRAN-U je moguće učitavati podatke u memoriju na dva načina: neformatizirane (binarno) i formatizirane, kada se podaci zadaju znacima alfabeta kojim raspolaže računalo i prevode u binarni oblik, a koji od načina će se primjeniti ovisi od hardware-skih karakteristika perifernog uređaja i potreba. Izgled sloga formatiziranih podataka definiran je instrukcijom FORMAT.

Lista, odnosno popis varijabli, elemenata polja i polja definira adrese memorijskih lokacija u koje će podaci biti učitani. Varijable se učitavaju tako da se samo redom navedu i odvoje zarezom. Polja se mogu učitavati na tri načina:

a. navođenjem imena polja i indeksa polja, koji je ili cjelobrojna konstanta ili cjelobrojna varijabla koja definira o kom elementu polja se radi.

Primjer:

```
I = 1

READ(IN) A(1), A(2), B(I), C(2)
```

b. Navođenjem samo imena polja. U ovom slučaju učitava se cijelo polje, odnosno onoliko elemenata koliko je specificirano u deklaraciji polja. Ukoliko je polje više dimenzionalno elementi polja se u pravilu učitavaju tako da se najbrže mijenja prvi index, a najsporije posljednji.
c. Navođenjem imena polja, indeksa polje i granica kako indeks varira. Ukoliko je polje više dimenzionalno, za svaku dimenziju je potrebno definirati način variranja indeksa. Kod ovakvog načina učitavanja podataka moguće je kombinirati ga sa načinom opisanim pod a) za više dimezionalna polja.

Primjer:

```
DIMENSION I(10,20), J(100), B(5,6,7), C(100), d(20)

...

...

...

READ (5,20) ((I(K,L), K=1,10), L=1,20)

READ (5,20) (J(K), K=21,80)

READ (5,30) (((A(K,L,M), K=1,5), L=1,7), M=1,10)

M=5

READ (5,30) ((B(K,L,M), L=1,6), K=1,5)
```

Polje **I** učitat će se tako, da se prvo učitaju **I(1,1)** do **I(10,1)** pa **I(1,2)** do **I(10,2)** i tako redom, odnosno indeks prve dimenzije polja **I**, promijeni sve svoje vrijednosti od 1 do 10, dok indeks **L**, druge dimenzije ostaje isti. Tek promjenom indeksa **L** koji se povećava za 1 ponovo se varira indeks **K** od 1 do 10. Koji od indeksa višedimenzionalnog polja će se najbrže mijenjati zavisi o tome kako su složeni ulazni podaci. Kod učitavanja polja **B**, prvo se varira drugi indeks **L** i za cjelokupnu njegovu primjenu prvi indeks dimenzije **K** ostaje nepromijenjen.

Primjer:

```
DIMENSION A(100), B(2,3,4), i (20)

IN = ...

READ (in,20) A, (((B(J,K,L), J=1,2), K=1,3), L=1,4)

* ,(I(J), J=11,20), C, D
```

Ovom instrukcijom učitano je cijelo polje **A** i **B**, zadnjih 10 elemenata polje I i varijable **C** i **D**.

1. Naredba WRITE

Instrukcija WRITE služi za pisanje, bilo neformatizirano (binarno), ili formatizirano, odnosno kada se vrši konverzija iz binarnog koda u alfanumerički kod računala. Sva pravila i razmatranja koja se odnose na instrukciju READ odnose se i na instrukciju WRITE koja ima opću formu:

```
WRITE (a, b, ERR = c, END = d) LISTA
```

gdje su svi parametri identični kao u instrukciji **READ** s tom razlikom što se odnose na pisanje (ispis) podataka na perifernu jedinicu. Parametar **ERR=c** označava obilježje instrukcije na koju se skače ukoliko se otkrije greška prilikom pisanja, a parametar **END=d** označava obilježje instrukcije na kojoj program nastavlja rad ukoliko na perifernoj jedinici nema više mjesta za pisanje.

1. *Naredba FORMAT*

Instrukcija FORMAT služi za definiranje formata sloga podataka koji se učitavaju, odnosno ispisuju i ujedno osigurava konverziju podataka iz binarnog koda u alfanumerički i obrnuto. Opća forma instrukcije FORMAT je:

`n FORMAT (q1,t1,z1,t2,z2, ..., tn,zn,q2)`

`n` obilježje instrukcije FORMAT, obavezno se zadaje i služi za povezivanje ulaznih/izlaznih instrukcija (READ i WRITE) sa formatom sloga podataka koji se učitavaju, odnosno pišu,

`q1` kosa crta "/" završetak sloga. Može se izostaviti ili navesti onoliko koliko ih je potrebno,

`t1` specifikacija formata ili niz specifikacija formata između otvorene i zatvorene zagrade,

`z1` separator npr. (,)

Specifikacija formata definira izgled dijela sloga podataka koji se učitavaju, odnosno ispisuju. Opća forma specifikacije formata je:

`rcw.d, wc ili cn`

`r` cjelobrojna konstanta koja predstavlja broj ponavljanja specifikacije formata. Ukoliko je jedan, može se ispustiti,

`c` jedan od kodova specifikacije formata (I,L,F,E,D,G,A,H,X i T),

`w` cjelobrojna konstanta koja određuje broj pozicija u slogu, koje opisuje specifikacija formata,

`.d` cjelobrojna konstanta koja predstavlja broj decimalnih mjesta desno od decimalne točke,

`n` cjelobrojna konstanta koja definira poziciju u slogu.

Svaka vrsta varijable u FORTRAN-u ima svoju specifikaciju formata i to:

varijable u duploj preciznosti **`rDw.d`**

cjelobrojne varijable **`rIw`**

logičke varijable **`rLw`**

realne i kompleksne varijable **`rFw.d,rEw.d i rGw.d`**

Hollerith konstante, tekst i uređenje sloga **`rAw, wH,"...",wX i Tn`**

1. **l specifikacija formata**

   Služi za cjelobrojne varijable. Opća forma je:

rIw

pri čemu je:

r broj ponavljanja

w broj znamenki

2. ## F specifikacija formata

   Služi za učitavanje i ispisivanje realnih brojeva. Opća forma je:

   rFw.d

   pri čemu je:

   r broj ponavljanja

   w ukupni broj mjesta

   d broj decimalnih mjesta

3. ## E specifikacija formata

   Služi za učitavanje i ispis realnih veličina u eksponencijalnom obliku. Opća forma je:

   rEw.d

   pri čemu je:

   r broj ponavljanja

   w ukupni broj mjesta

   d broj decimalnih mjesta

4. ## G specifikacija formata

   Služi za realne veličine i veličine u duploj preciznosti. Opća forma:

   rGw.d

   pri čemu je:

   r broj ponavljanja

   w ukupni broj mjesta

   d broj decimalnih mjesta

5. ## D specifikacija formata

   Služi za veličine u duploj preciznosti. Opća forma je:

rDw.d

pri čemu je:

r broj ponavljanja

w ukupni broj mjesta

d broj decimalnih mjesta

6. A specifikacija formata

Služi za učitavanje i ispis alfanumeričkih znakova (byte-a informacije). Opća forma je:

rAw

pri čemu je:

r broj ponavljanja

w ukupni broj znakova

7. X specifikacija formata

Služi za preskakanje određenog broja pozicija u ulaznom, odnosno izlaznom slogu. Opća forma je:

wX

pri čemu je:

w broj preskočenih pozicija u slogu.

8. T specifikacija formata

Služi kao tabulator za pozicioniranje veličine u ulaznom, odnosno izlaznom slogu. Opća forma je:

T n

pri čemu je:

n cjelobrojna konstanta koja definira poziciju podatka u slogu

9. Upravljanje ispisom

Kod izlaznih perifernih jedinica, kao što su štampač i terminal, prvi znak svakog retka ispisa služi za upravljanje ispisom. Upravljanje se vrši na taj način da se u prvu poziciju izlaznog retka smješta znak operacije za upravljanje. Znalkovi za upravljanje su "(praznina), "0", "1" i "+", a njihovo je značenje:

kod operacija

praznina prelazak na slijedeću liniju prije štampanja

0 preskok 2 linije prije štampanja

1 skok na prvu liniju nove stranice prije štampanja

+ ispis u istom redu

1. Izvršne naredbe

Izvršne se naredbe mogu podijeliti u tri osnovne skupine:

- naredbe pridruživanja,
- naredbe kontrole,
- ulazno-izlazne naredbe.

1. *Naredbe pridruživanja*

Kao što im samo ime govori, to su naredbe kojima se varijabli ili elementu polja pridružuje vrjednost konstante, varijable ili izraza. Imena varijable ili elemenata polja kojima se pridružuje vrijednost obavezno mora biti s lijeve strane znaka za pridruživanje " **=** ". Opći je oblik slijedeći:

```
ime = vrijednost
```

pri čemu je:

ime Ime varijable ili elementa polja prema pravilima zadavanja imena

vrijednost Vrijednost konstante, varijable ili izraza koji se pridružuje.

1. Izrazi

U programskom jeziku FORTRAN definirane su operacije koje se mogu izvoditi nad konstantama, varijablama ili elementima polja. Tri su osnovna tipa operacija:

- aritmetičke,
- relacijske,
- logičke.

Unutar jednog izraza moguće su sve tri vrste operacija. Hijerarhijski redoslijed izvođenja operacija prikazan je tablicom. Unutar izraza mogu postojati i pozivi funkcijskih potprograma i u tom se slučaju oni prvo pozivaju, izvršavaju i njihov rezultat dalje obrađuje po opisanom redoslijedu.

Vrsta operacije	Operacija		Operandi
	`**`	potenciranje	
ARITMETIČKE	`-`	promjena predznaka	Aritmetičke konstante, varijable
	`* i /`	množenje i dijeljenje	ili elementi polja.
	`+ i -`	zbrajanje i oduzimanje	
	`.LT.`	manje	
	`.LE.`	manje ili jednako	Aritmetičke i logičke konstante,
RELACIJSKE	`.EQ.`	jednako	varijable i elementi polja.

	.NE.	različito	Sve ove operacije imaju isti
	.GE.	veće ili jednako	prioritet izvršavanja.
	.GT.	veće	
	.NOT.	negacija	
LOGIČKE	**.AND.**	konjunkcija (I)	Logičke konstante i varijable
	.OR.	disjunkcija (ILI)	

1. Kontrolne naredbe

1. *Naredba odluke (IF)*

Postoje tri osnovne vrste instrukcija odluka: aritmetičk IF, logički IF i IF Blok.

1. Aritmetički IF

Instrukcija aritmetičke odluke izračunava aritmetički izraz na osnovu kojega se donosi odluka. Moguće su tri akcije ovisno o tome da li je vrijednost izraza manja, veća ili jednaka nuli.

SIntaksa:

`IF (e) n1,n2,n3`

e aritmetički izraz

n_1 labela instrukcije koja se izvodi ako je rezultat manji od nule

n_2 labela instrukcije koja se izvodi ako je rezultat jednak nuli

n_3 labela instrukcije koja se izvodi ako je rezultat veći od nule

2. Logički IF

Ako je izraz **e** u zagradi istinit, izvršava se naredba **s**.

Sintaksa:

`IF (e) s`

`i` logički i/ili relacijski izraz

`s` bilo koja izvršna naredba, osim **DO** i **IF**

3. IF BLOK

IF blok se koristi ako je potrebno izvesti više instrukcija u slučaju istinitosti izraza. Unutar konstrukcije IF bloka smiju se pojaviti slijedeći elementi naredbe. Elementi **`IF()THEN`** i **`END IF`** su obavezni i smiju se koristiti samo jednom, **`ELSE`** je neobavezan, a ako se koristi smije se upotrijebiti samo jednom, **`ELSE IF () THEN`** je neobavezan, a ako se koristi može se upotrijebiti više puta.

```
IF () THEN

ELSE IF () THEN

ELSE

END IF
```

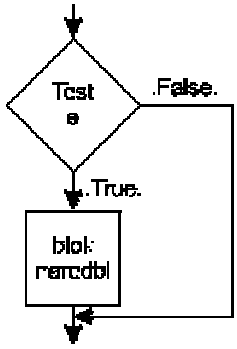
Opći oblik:

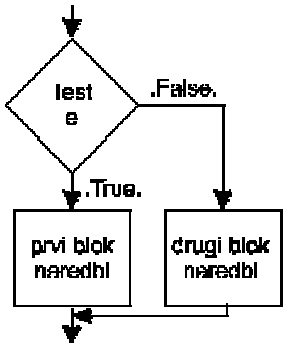
```
IF (e) THEN

block 1

ELSEN IF (e_l) then

block 2

.

.

.

ELSE

block n

END IF
```

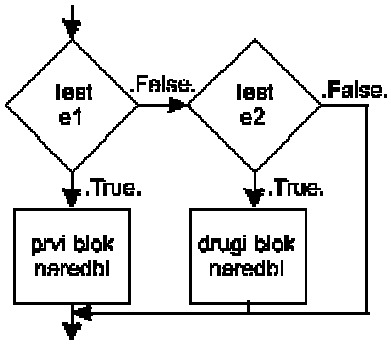
pri čemu su:

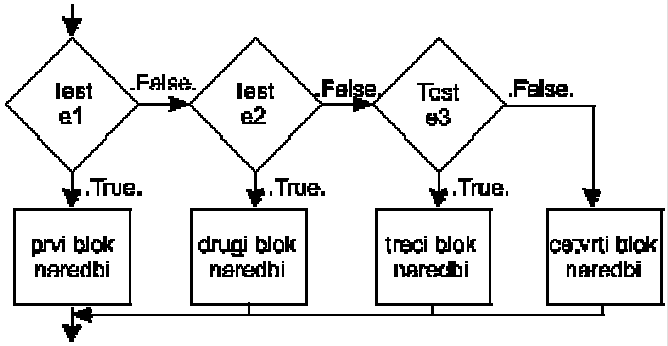
e, e_l logički izrazi

block (1,..., n) niz fortranskih naredbi

IF (e) THEN
 	
blok naredbi
 	
END IF
	

IF (e) THEN
 	
prvi blok naredbi
 	
ELSE
 	
drugi blok naredbi
 	
END IF
	

IF (e1) THEN
 	
prvi blok naredbi
 	
ELSE IF(e2)THEN
 	
drugi blok naredbi
 	
END IF
	

IF (e1) THEN
 	
prvi blok naredbi
 	
ELSE IF (e2) THEN
 	
drugi blok naredbi
 	
ELSE IF (e3)THEN
 	
treci blok naredbi
 	
ELSE
 	
cetvrti blok naredbi
 	
END IF
	

2. ***DO petlja***

Instrukcija **DO** kontrolira uzastupno izvođenje bloka naredbi. Broj ponavljanja izvođenja bloka ovisi o vrijednosti kontrolne varijable (tzv. indeks petlje).

Sintaksa:

DO [n] i = m_1, m_2[,m_3]

n labela zadnje instrukcije u petlji koja će se izvesti prije ponavljanja (to ne smiju biti instrukcije **DO, GO TO, IF, RETURN, PAUSE i STOP**). Ukoliko se labela ne navede petlja obavezno završava instrukcijom **END DO**

i indeks petlje, cjelobrojna varijabla koja služi kao kontrolna varijabla.

m_1 početna vrijednost kontrolne varijable

m_2 krajnja vrijednost kontrolne varijable

m_3 korak promjene kontrolne varijable, ukoliko se ne navede podrazumijeva se da je korak **1**.

Unutar **DO** petlje mogu se pojaviti sve izvršne instrukcije, ali postoje ograničenja kod skokova unutar programa (instrukcije **GO TO** i **IF**). Iz petlje je dozvoljeno u svakom trenutku iskočiti, ali nije dozvoljen skok u petlju. Unutar jedne petlje moguće je imati drugu petlju ili više drugih petlji, ali unutarnje petlje ne mogu završiti nakon vanjske petlje. Moguće je iskočiti iz unutarnje petlje u vanjsku, ali nije dozvoljen skok iz vanjske u unutarnju. Promjena vrijednosti kontrolne varijable unutar petlje nije dozvoljena.

3. *DO WHILE petlja*

Naredba **DO WHILE** radi slično naredbi **DO**, s tim da **DO** uvijek radi unaprijed zadani broj koraka, a **DO WHILE** se izvodi sve dok je zadovoljen logički izraz u zagradi.

Uvjet se provjerava svaki put prije ulaska u petlju, te ako je zadovoljen, program ulazi u petlju.

Iz **DO WHILE** petlje se može nasilno izaći (**GO TO** i sl.), ali se u nju ne smije uskočiti iz programa.

Sintaksa:

DO [s[,]] WHILE (e)

s labela izvršne instrukcije koja fizički mora slijediti **DO WHILE**

e logički izraz čija se istinitost provjerava

Labela je neobavezni parametar, ali ako se ne navede petlja mora završiti s **END DO**.

4. DATOTEKA (file)

- je cjelina u kojoj je pohranjen skup podataka na perifernoj jedinici računala
- zapis u datoteci može biti binarni zapis ili simbolički oblik (ASCII kod)
- dat. se sastoji od slogova (u kojima su zapisani podaci), slog je najmanja količina informacije unutar datoteke kojoj se može pristupiti u operacijama čitanja ili pisanja.

1. VRSTE DATOTEKA:

1. ovisno o duljini sloga:

- DATOTEKE S FIKSNOM DULJINOM SLOGA
- DAT. S VARIJABILNOM DULJINOM SLOGA (ovakve datoteke radi npr. editor)
-

2. ovisno o pristupu slogu datoteke:

- SEKVENCIJALNA - (dat. sa sekvencijalnim pristupom) - nekom slogu unutar datoteke može se pristupiti samo ako se pristupi svim slogovima koji mu prethode unutar datoteke
- DIREKTNA - (dat.sa direktnim pristupom) - svakom slogu se može pristupiti direktno

3. ovisno o načinu zapisa:

- NEFORMATIRANA - binarni zapis
- FORMATIRANA - simbolički oblik (ASCII kod)

POMOĆNE I/O (ULAZNO/IZLAZNE) NAREDBE

- OPEN
- CLOSE
- INQUIRE
- REWIND
- BACKSPACE
- ENDFILE
- DELETE

1. OPEN

Služi za povezivanje postojeće datoteke s logičkom jedinicom ili stvara novu datoteku i povezuje je sa logičkom jedinicom. Naredbom OPEN određuju se atributi datoteke potrebni za read/write naredbe.

Sintaksa naredbe:

```
OPEN (parametar [ ,parametar] ....)
```

parametar predstavlja ključnu riječ (keyword)

keyword = value

Ključne riječi se razvrstavaju u više kategorija ovisno o funkcijama. Redoslijed specifikacija ključnih riječi proizvoljan je.

1. **Ključne riječi koje određuju jedinicu i datoteku (koja se otvara)**

`UNIT=lun` određuje logičku jedinicu koja se povezuje s datotekom, mora biti navedena u listi parametara osim ako se lun ne nalazi na prvom mjestu u listi parametara OPEN naredbe

= `lun` (*logical unit number*) integer 0-99, koji se odnosi na određenu datoteku ili I/O uređaj

`FILE ili NAME=file` ime datoteke koja se povezuje na log. jedinicu

`STATUS` određuje status datoteke koja se otvara

= **'OLD'** datoteka mora postojati

= **'NEW'** stvara novu datoteku

= **'SCRATCH'** - briše datoteku nakon zatvaranja

= **'UNKNOWN'** - ako postoji datoteka onda će se otvoriti ako ne postoji stvara novu

2. **Ključne riječi koje opisuju organizaciju datoteke**

`ACCESS` određuje pristup datoteci

= **'SEQUENTIAL'** - sekvencijalni pristup (default)

= **'DIRECT'** - direktni pristup (po brojevima slogova)

= **'KEYED'** - pristup prema ključnoj riječi

= **'APPEND'** - sekvencijalni, nakon zadnjeg sloga u datoteci

3. **Ključne riječi koje opisuju slogove unutar datoteke**

CARRIAGECONTROL određuje način interpretiranja prve kolone kod printanja /pisanja

= **'FORTRAN'** - fortranska interpretacija prvog znaka (default za formatiranu dat.)

= **'LIST'** - standardna interpretacija ASCII znakova

= **'NONE'** - nema implicitne cc. (default za neformatiranu dat.)

FORM određuje da li se u datoteku koja se otvara upisuje ili iz nje čita koristeći formatirane ili neformatirane READ/WRITE naredbe

= **'FORMATTED'** - default za sekvencijalne datoteke

= **'UNFORMATTED'** - default za direktne i keyed datoteke

RECL određuje duljinu logičkog sloga u datoteci

= num. izraz izražen u *byte* jedinicama za slučaj formatiranog sloga odnosno u *longword* jedinicama (4byte) za slučaj neformatiranog sloga

4. Ključne riječi koje osiguravaju dodatne mogućnosti pri direktnom pristupu

ASSOCIATEVARIABLE predstavlja integer varijablu koja se mijenja nakon svake operacije direktnog pristupa tako da daje broj slijedećeg sloga u datoteci

5. Posebne ključne riječi

ERR (transfer-of-control specifier)

= integer varijabla koja označava labelu izvršne naredbe kojom se nastavlja izvođenje programa u slučaju greške

IOSTAT varijabla u koju se sprema status (0 ako je dobro, 1 ako je greška)

= integer veći ili jednak 0 (0 ako je dobro ili 1 ako je greška)

Primjer 1: (otvaranje postojeće sekvenc. datoteke - seqfo.dat):

```
OPEN (unit=1, status='OLD', file='seqfo.dat',
> form='formatted', access='SEQUENTIAL',
> err=100)
```

Primjer 2: (otvaranje direktne nove datoteke):

```
character *20 ime
5 write (6,10)
10 format ('Upisi ime direktne datoteke:',$)
read (5,15) ime
15 format (a20)
OPEN (unit=1, status='NEW', file=ime, acces='DIRECT'.
> FORM='FORMATTED', recl=80, err=5)
.
.
.
CLOSE (1)
END
```

1. ## CLOSE

   Služi za zatvaranje datoteke.

   Sintaksa naredbe:

   CLOSE ([unit=] u [,DISPOSE=p] [,ERR=s] [ISOSTAT=ios])

   UNIT=u lun (logical unit number) integer 0-99, koji se odnosi na određenu datoteku ili I/O uređaj

   DISPOSE znakovni izraz koji određuje dispoziciju datoteke, može poprimiti vrijednosti:

   = **'KEEP'/'SAVE'** - zadržava datoteku nakon zatvaranja

   = **'DELETE'** - briše datoteku nakon zatvaranja

   = **'PRINT'** - šalje datoteku na ispis

2. REWIND

Služi za pozicioniranje na prvi slog već otvorene sekvencijalne datoteke, bez obzira na trenutnu poziciju unutar datoteke.

REWIND naredba se ne koristi u datotekama otvorenim za direktni pristup ili pristup po ključnoj riječi.

3. BACKSPACE

Služi za postavljanje trenutno otvorene sekvencijalne datoteke na početak prethodnog sloga.

BACKSPACE naredba ne bi se smjela koristiti u datotekama otvorenim za direktni pristup, pristup po ključnoj riječi i append pristup.

Sintaksa REWIND i BACKSPACE naredbe:

```
REWIND ([ UNIT=] u [ ,ERR=s] [ ,IOSTAT=ios] )

BACKSPACE ([ UNIT=] u [ ,ERR=s] [ ,IOSTAT=ios] )
```

`UNIT=u` lun(logical unit number) integer 0-99, koji se odnosi na određenu datoteku ili I/O uređaj

`ERR=s` integer varijabla koja označuje labelu izvršne naredbe s kojom se nastavlja izvođenje programa u slučaju greške

`IOSTAT=ios` pozitivni integer ako je došlo do greške

= 0 ako nema greške