FORTRAN

osnove

Osnove programskega jezika Fortran (19. 10. 2010)

Uporabljali bomo prosti slog (kodo začnemo pisati v prvi stolpec, vrstica je lahko daljša od 72 znakov)

Skladnja (sintaksa) v skladu s Fortran 90 standardom

program ime_programa

Telo programa
- deklaracije spremenljivk
- Izvršilni stavki
- vhodno/izhodni ukazi ...

end program ime_programa

ime_programa ne sme biti hkrati ime katere izmed v programu uporabljenih spremenljivk.

Ime spremenljivke lahko vsebuje črke angleške abecede, cifre 0-9 in _ (podčrtaj).

Posebni znaki so = : + presledek - * / ( ) , . $ ' ! " % & ; < > ?

1) = enačaj; v prireditvenih stavkih
2) : dvopičje; pri določanju območja tabel
3) + seštevanje
4) - odštevanje
5) * množenje
6) / deljenje
7) presledek ; pri pisanju kode, izpisih
8) ( ) oklepaji; pri deklaraciji tabel, pri rač. operacijah
9) , vejica; pri deklaraciji večdimentionalnih polj, pri funkcijah etc.
10) . pika; pri zapisu decimalnih števil
11) $ dolarski simbol; pri direktivah
12) ‘ akcent; pri izpisih
13) ! klicaj; znaki desno od njega so komentar
14) “ narekovaj; pri izpisih
15) % procent; pri sestavljenih spremenljivkah
16) & in (ampersand); nadaljevanje vrstice , namelist
17) ; podpičje; če je več stavkov v eni vrstici
18) < manjše od
19) > večje od
20) ? vprašaj; pri poimenskem branju spremenljivk
21) \ poševnica nazaj; je t.i. escape character

Spremenljivke v Fortranu

1. INTEGER cela števila
2. REAL realna števila
3. COMPLEX kompleksna števila
4. CHARACTER znakovni niz
5. LOGICAL dve vrednosti: .TRUE. (DA) ali .FALSE. (NE)
6. Sestavljeni tipi n.pr. oseba (ime, priimek, leto rojstva)

Primer:

```
integer(4) :: i,j,n
real(8) :: pi, r2,ploscina
complex(4) :: z
character(len=5) :: molekula
logical :: ali_dezuje
```

Pomemben je stavek:
implicit none
Od nas zahteva dosledno deklaracijo vseh uporabljenih spremenljivk

Besede, zapisane **poudarjeno**, so rezervirane (deklaracije, ukazi); nobena spremenljivka se ne more tako imenovati.

Nekaj vgrajenih (intrinsic) ukazov oziroma funkcij:

Funkcija	Excel	Fortran
eksponent	^	**
kvadratni koren	sqrt( )	sqrt( )
sinus	sin()	sin()
kosinus	cos()	cos()
tangens	tan()	tan ()
naravni logaritem	ln()	log()
absolutna vrednost	abs()	abs()

http://office.microsoft.com/en-in/excel-help/list-of-worksheet-functions-alphabetical-HA010277524.aspx?CTT=3#BM13

Ukaza **print** in **read**

Uporabljamo ju za izpis na zaslon ter za branje podatkov s tipkovnice.

Primer:

print *, ‘vnesi stevilo a’

read *, a

Vedno sledi ***** in **,**

IF stavek (tudi Block IF) (26. 10. 2010)

Sintaksa:

if(logični pogoj 1)**then**
stavek 1
stavek 2
....
elseif(logični pogoj 2)**then** ta del se lahko poljubnokrat ponovi
stavek 3
stavek 4
...
else tu preostane le še ena možnost
...
endif

Logični pogoj je običajno rezultat primerjave dveh vrednosti oziroma izrazov.

Če je izpolnjevanje logičnega pogoja dovolj za izvršitev nekega ukaza, lahko zapišemo tudi takole:

if(logični pogoj) *stavek* (brez **then** in **endif**)

Primerjalni operatorji

Pomen	F90	F77	primer
enako	==	.eq.	a==b
večje	>	.gt.	D>0
manjše	<	.lt.	D<0
večje ali enako	>=	.ge.	a>=b
manjše ali enako	<=	.le.	a<=b
ni enako	/=	.ne.	x/=0

Operatorji nad logičnimi izrazi

Pomen	ukaz	primer
logični *ali*	**.or.**	(logični izraz 1 **.or.** logični izraz 2)
logični *ne*	**.not.**	(**.not.** logični izraz)
logični *in*	**.and.**	(logični izraz 1 **.and.** logični izraz 2)

.or.

logična vrednost 1	logična vrednost 2	operator	rezultat
.true.	.false.	.or.	.true.
.false.	.true.	.or.	.true.
.true.	.true.	.or.	.true.
.false.	.false.	.or.	.false.

.not.

Logična vrednost	operator	vrednost
.true.	.not.	.false.
.false.	.not.	.true.

.and.

logična vrednost 1	logična vrednost 2	operator	rezultat
.true.	.false.	.and.	.false.
.false.	.true.	.and.	.false.
.true.	.true.	.and.	.true.
.false.	.false.	.and.	.false.

Primer: **if**(x>0 .**and**. y>0) **print** *, ‘tocka je v prvem kvadrantu’

do while stavek

Sintaksa:

do while (logični izraz)
  stavek 1
  stavek 2
  ...
end do

Dokler logični izraz ne dobi vrednosti **.true.** , se stavki znotraj telesa do while zanke ponavljajo. Ta stavek je uporaben tam, kjer ne vemo vnaprej, kolikokrat bomo določene ukaze morali ponavljati do izpolnitve zahtevanega pogoja.

Ta stavek je nadomestil **if**(logični izraz)**go to** *n* in *n* **continue** stavka.

Funkcijski podprogram

Pogosto je potrebno izračunati vrednost dane funkcije (ene ali več spremenljivk) v neki točki. Da se izognemo večkratnemu pisanju formule v programu, uporabimo funkcijski podprogram.
Funkcijski podprogram (funkcija) ni del glavnega programa, temveč ga pokličemo od tam z ustreznimi argumenti.

Glavni program:

```
real(8) :: a,x,f
...
x=5.
a=f(x)  ! tule pokličemo izračun funkcije
...
end program

function f(x)
implicit none
real(8) :: x,f
f=x**2+5.d0      !  tu priredimo spremenljivki f vrednost po enačbi
return           !  spremenljivka f je ime funkcijskega podprograma
end              !  f mora biti deklariran z istim tipom v glavnem in
                 ! funkcijskem podprogramu
```

Oblikovanje izpisa (9. 11. 2010)

Izpis oblikujemo s stavkoma **write** in **format**. Ukaz **format** uporabljamo tudi pri branju podatkov iz datotek.

Skladnja

write(*,*) spremenljivka1,spremenljivka2,...,spremenljivkaN

Pri tem prva zvezdica v (*****,*****) pomeni izpis na standardni izhod, to je zaslon, druga pa pomeni izpis v privzeti (angl. *default*) obliki. S tem ni nič narobe, a so števila izpisana na preveliko število mest.
Ukaz ‘’ **write**(*,*) ‘’ je ekvivalenten ukazu ‘‘**print *,’’ .**

Oblika (format) izpisa spremenljivk je odvisen od njihovega tipa in s tem povezane deklaracije na začetku programa.

Celoštevilčne spremenljivke (tip integer) so običajno sestavljene iz 4 zlogov (bytov), to pomeni 32 bitov. V 32-bitnem registru lahko torej zapišemo cela števila v območju med $-2^{32-1} < i < 2^{32-1} - 1$
− 2 147 483 648 < i < 2 147 483 647 , za kar potrebujemo do 11 mest.

Izpis celega števila oblikujemo z določilom **I*n***, kjer je *n* število željenih izpsanih mest. *n* mora biti enak desetiškemu redu velikosti števila ali pa večji, sicer bo na izpisu *n* *.

Če je n večji, se pred število izpiše ustrezno število presledkov.

```
program izpis
implicit none
integer(4) :: i
read(*,*)i
write(*,10)i
10 format(i2)
end
```

➔

```
C:\Programi>izpis.exe
83
83
C:\Programi>izpis.exe
5
 5   (presledek pred 5!)
C:\Programi>izpis.exe
102
** (tromestnega števila ne  moremo zapisati na 2 mesti)
C:\Programi>izpis.exe
-83
**  (ni prostora za predznak)
```

Izpis realnih števil ima dve osnovni obliki, in sicer

1. oblika z decimalno piko (fortran ne pozna vejice)
2. eksponentna oblika

Ad 1) število izpišemo z določilom "F*a.b*'', kjer je *a* število vseh rezerviranih mest, vključno z decimalno piko in mestom za predznak; *b* pa je število decimalnih mest.

```
program izpis
implicit none
integer(4) :: i
real(4)    :: a
write(*,*)'vnesi a'  ➔
read(*,*)a
write(*,10)a
10 format(f5.3)
end
```

```
C:\Programi>izpis.exe
 vnesi a
12.3456
*****      (pri f5.3 ostane za celi del samo 1 mesto)
C:\Programi>izpis.exe
 vnesi a
1.23456
1.235      (višek mest 'odreže')
C:\Programi>izpis.exe
 vnesi a
-1.23456
*****   (-,1 ter . porabijo 3 mesta, na voljo sta pa 2)
C:\Programi>izpis.exe
 vnesi a
-0.123456
-.123   (tako gre, če izpustimo 0)
```

Pri tej obliki moramo vsaj približno vedeti, s kako velikimi števili imamo opravka, da lahko predvidimo dovolj mest za izpis.

V kolikor pa tega ne vemo vnaprej, je primernejši zapis s pomočjo eksponenta. Tukaj ločimo tri podzvrsti:

1. Eksponentni v obliki E*a.b*
2. Znanstveni v obliki ES*a.b*E*c*
3. Inženirski v obliki EN*a.b*E*c*

b je število mest v mantisi za decimalno piko.

c določa število mest eksponenta, n.pr. E+01 (c=2) ali E-007 (c=3).

Pri eksponentni obliki se število podaja z vodilno 0, ki ji sledi decimalna pika in eksponent: 0.123456E+03 = 123.456

Pri znanstveni obliki je vodilna cifra različna od 0: 1.234560E+00

Pri inženirski obliki se eksponent spreminja v korakih po 3, sicer je enaka znanstveni.

```
program izpis
implicit none
integer(4) :: i
real(4)    :: a
write(*,*)'vnesi a'
read(*,*)a
write(*,*)a
write(*,*)'12345678901234567890'
write(*,'(a20)')'12345678901234567890'
write(*,10)a
10 format(e15.6)
write(*,20)a
20 format(es15.6)
write(*,30)a
30 format(en15.6)
end
```

```
 vnesi a
123.456
   123.45600
 12345678901234567890
12345678901234567890
   0.123456E+03
   1.234560E+02
 123.456001E+00
```

Izpis znakovnih nizov

Te izpišemo z določilom A*n,* kjer je *n* število rezerviranih mest.

```
program izpis
implicit none
character(len=11) :: beseda
write(*,*)'vnesi a'
read(*,*)a
beseda='abrakadabra'
write(*,40)beseda
40 format(A5)
write(*,50)beseda
50 format(A11)
end
```

Izpis:

```
abrak
abrakadabra
```

Izpis logičnih spremenljivk

Izpišemo jih z določilom Ln, kjer je *n* število rezerviranih mest v izpisu.

```
program izpis
implicit none
logical    :: ena_je_dva
ena_je_dva=.true.
write(*,60)ena_je_dva
60 format(L1)
write(*,70)ena_je_dva
70 format(L5)
end
```

Izpis:

```
12345678901234567890
   0.123456E+03
   1.234560E+02
 123.456001E+00
abrak
abrakadabra
T
    T
```

Izpis kompleksnih števil

Kompleksna števila nastopajo kot par *(a,b)* , kjer je *a* vrednost realne komponente in *b* vrednost imaginarne.

```
program izpis
implicit none
complex(8) :: z
z=(1.d0,2.d0)
write(*,80)z
80 format(2f8.2)
write(*,*)z
end
```

```
    1.00     2.00
(   1.0000000000000000     ,   2.0000000000000000     )
```

Presledek in tabulator

Pri izpisih več spremenljivk v isti vrstici pogosto mednje vrinemo enega ali več presledkov z določilom *n*X, kjer je n število presledkov.

Če pa želimo, da se posamezni izpisi začnejo na ustreznih mestih, se poslužimo t.i tabulatorja z določilom T*n*, kjer je *n* številka znaka v vrstici, kjer se izpis začne.

```
write(*,'(a60)')'123456789012345678901234567890123456789012345678901234567890'
write(*,90)a,a,a
90 format(t15,f8.4,t30,f8.4,t45,f8.4)
```

```
123456789012345678901234567890123456789012345678901234567890
               123.4560       123.4560       123.4560
```

Uporaba datotek za pisanje in branje podatkov

Pogosteje kot na zaslon zapisujemo podatke v datoteke, pa tudi beremo iz njih.

Datoteko odpremo za branje ali pisanje z ukazom **open**. Skladnja:

open(iunit,file='ime_datoteke.dat')

V datoteko pišemo z ukazom **write**(*iunit,ifmt*), kjer sta *iunit* številka 'enote' oz. datoteke, *ifmt* pa številka ukaza format.

Primer:
write(1,10)a
10 format(f10.3)

Po končanem pisanju/branju datoteko zapremo z ukazom **close**(iunit), npr. **close**(1).

Če je odprtih več datotek hkrati, morajo imeti različne *iunit* številke.

Ukaz za branje **read**(iunit,ifmt) ima enako skladnjo kot ukaz **write**; lahko uporabljata tudi *isti* formatni stavek, če gre za branje enako formatiranih podatkov.

Druge možnosti ukaza **open** si bomo ogledali kasneje.

''**DO**'' zanka

Pri računanju je eden izmed najpogosteje uporabljanih programerskih elementov zanka z vnaprej znanim številom ponavljanj. Skladnja:

do stevec = spodnja_meja, zgornja_meja, korak
  stavek1
  ...
  stavekN
enddo

Spremenljivke *stevec*, *spodnja_meja*, *zgornja_meja* in *korak* so tipa **integer**.

Če je spodnja_meja > zgornja_meja in korak > 0, se zanka nikoli ne izvede.

Če je spodnja_meja = zgornja_meja, se zanka izvede 1-krat, neodvisno od koraka.

Tabele (polja)

Pogosto imamo opravka s tabelami v 1 ali 2 dimenzijah. Enodimenzionalni tabeli pravimo tudi vektor.

V tabele spravljamo spremenljivke *istega* tipa, kar povemo pri deklaraciji.

real(4) :: tocka(3)

Tu smo povedali, da so elementi tabele ‘’tocka’’ trije in da so vsi tipa real.

Na posamezne elemente tabele se sklicujemo z imenom tabele ter indeksom elementa v tabeli, n.pr. write(*,*)tocka(2) bo izpisal y koordinato.