Fortran 90/95

+ sopii erityisesti numeriikkaan:
  + optimoivat kääntäjät ⇒ tehokas koodi
  + mukana valmiiksi paljon varusfunktioita
  + kompleksiluvut
  + taulukko-operaatiot
  + operaattorit laajennettavissa myös omille tietotyypeille
+ standardoitu ⇒ hyvä siirrettävyys
+ saatavana hyviä aliohjelmakirjastoja (NaG yms.)
+ kääntäjät hyväksyvät myös vanhat ohjelmat, joita on liikkeellä paljon
- useimmat vanhoista ohjelmista hirveitä sekasotkuja
- kivikautisia sudenkuoppia, joita opittava varomaan (aina f77:ään saakka; f90:stä lähtien ongelmat voidaan välttää)

Esimerkki:

Lähdekielinen ohjelma tiedostossa `addition.f90`:

```
program addition
real x,y,z
write(*,*) 'anna kaksi lukua'
read (*,*) x,y
z=x+y
write(*,*) 'summa on',z
end program addition
```

Käännös ja suoritus (esimerkiksi):

```
>f95 -o add addition.f90
>add
anna kaksi lukua
1,3
summa on 4.00000000
>
```

Esimerkki: Yksinkertainen yhtälön ratkaisija

Kirjoitetaan yhtälö muotoon $x = f(x)$. Esimerkiksi $x^5 - x - 1 = 0 \Rightarrow$

$$x = x^5 - 1$$

tai

$$x = (1 + x)^{0.2}$$

```
program solve
! etsitaan yhtalon x**5-x-1=0
! reaalijuuri
real x0, x1

x0 = 0.5       ! arvataan alkuarvo
x1=(1.0+x0)**0.2
! iteroidaan, kunnes tulos ei muutu
do while (x1.ne.x0)
  x0 = x1
  x1 = (1.0+x0)**0.2
end do
write (6, *) x1, x1**5-x1-1
end program solve
```

```
>f95 -o solve solve.f90
>./solve
1.16730404 5.03132583E-07
```

Ohjelman ulkoasu

Fortran 77:ssa on vain kiinteä sarakesidonnainen kirjoitusasu. F90:stä lähtien käytännössä myös vapaa srakkeista riippumaton muoto. Eri muotoja ei saa käyttää sekaisin. Seuraavassa käsitellään vain vapaata muotoa.

Pienillä ja isoilla kirjaimilla ei ole eroa (paitsi mahdollisesti tiedostojen nimissä käyttöjärjestelmästä riippuen). Kannattaa kuitenkin olla johdonmukainen ja kirjoittaa sama nimi aina samalla tavalla. Typografisesti isojen kirjainten runsas käyttö voi tehdä tekstistä vaikeasti luettavaa.

Lause päättyy automaattisesti rivin loppuun, joten erityistä erotinta ei tarvita.

Jos lause jatkuu useammalle riville, se on erikseen osoitettava. Rivin lopussa oleva &-merkki ilmoittaa, että lause jatkuu seuraavalla rivillä:

```
y = 1.0 + x + 0.5 * x**3 &
  + 1.0/6 * x**4
```

Lauseen maksimipituus on 40 riviä.

Kommentit: Rivin loppu huutomerkistä (!) eteenpäin on kommentti, jota kääntäjä ei käsittele. Kommentti pättyy rivin loppuun, joten erityistä loppumerkkiä ei tarvita.

Yksinkertaiset muuttujat

Perinteisesti yksinkertaisilla muuttujilla on implisiittinen tyyppi:

- integer, jos ensimmäinen kirjain `I–N`
- real muuten

Vaarallista! Väärin kirjoitettu muuttujan nimi tulkitaan vain eri muuttujaksi.

```
x0=1.0
if (xO.gt.O) ...
```

Tämä voi johtaa hyvin hankalasti jäljitettäviin virheisiin. Määrittele siis kaikki muuttujat! Estä implisiittisten määrittelyjen käyttö:

```
implicit none
```

Yksinkertaisten muuttujien tyypit ovat:

```
integer
real
logical  (arvo .true. tai .false. )
complex
double precision
character
```

Muuttujan määrittelyn täydellinen muoto

```
tyyppi (parametrit), attribuutit :: nimi
```

Parametrit määrittelevät muuttujan esitystavan, attribuutit taulukon koon ja muuta tilanvaraukseen liittyvää tietoa.

F90:ssä haluttu esitystarkkuus määriteltävissä laitteistosta riippumattomalla tavalla. Mutta: toteutetaan vain muutaman erilaisen muuttujatyypin avulla, mielivaltaisen suurta tarkkuutta ei ole käytettävissä.

Muuttujan esitystarkkuus ilmoitetaan kind-attribuutilla:

```
integer count
integer (kind=selected_int_kind(5)) :: count
integer (selected_int_kind(5)) :: count
```

Kaksi viimeistä määrittelevät muuttujan, jonka esittämiseen tarvitaan korkeintaan 5 numeroa.

```
integer, parameter :: maxn=1000
real, parameter :: pi=3.141592654
```

`maxn` ja `pi` vakioita, joiden arvoja ohjelma ei saa muuttaa

```
real (kind=selected_real_kind(5)) :: a, b
real (kind=selected_real_kind(5,20)) :: c
```

tarkkuus viisi desimaalia; `c`:n lukualue 10^{-20} – 10^{20}.

```
integer, parameter :: short=selected_int_kind(4), &
                      long=selected_int_kind(7), &
                      longreal=selected_int_kind(10, 100)
integer (kind=short) :: i,j
integer (kind=long) :: isoluku
real (kind=longreal) :: big
```

Vakiot

Kokonaisluvut

```
123
123_short
1234567_long
```

Reaaliluvut

```
1.5
-1.5
1.5E10
1.5E-10
1.5_longreal
1.5E-10_longreal
```

Sijoitusoperaattori `=`

```
i=100
x=1.5
```

Lausekkeet

```
1.0+2.0*y/z**2-3.5*(y+x)
```

Normaali assosiatiivisuus:

- ensin `**` (potenssiinkorotus)
- sitten `*` ja `/` vasemmalta oikealle
- lopuksi `+` ja `-` vasemmalta oikealle
- järjestystä voidaan muuttaa suluilla

Ensin lasketaan sijoitusoperaattorin `=` oikealla puolella oleva lauseke, se muunnetaan vasemmalla olevan muuttujan tyyppiseksi ja talletetaan muuttujaan

```
real x
x = 1/2  ! x=?
```

Kokonaislukujen jakolaskun tulos on kokonaisluku (osamäärän kokonaisosa)

```
x=1.0/2
```

Varusfunktiot (intrinsic functions)

Trigonometriset funktiot (kulmat aina radiaaneina!):

```
sin(x), cos(x), tan(x)
asin(x), acos(x), atan(x), atan2(y,x)
```

Hyperboliset funktiot:

```
sinh(x), cosh(x), tanh(x)
```

Eksponentti, logaritmi ym.

```
exp(x), log(x), log10(x), sqrt(x)
```

Minimi ja maksimi; argumenttien määrä mielivaltainen:

```
min(x, y, ...), max(x, y, ...)
```

Itseisarvo

```
abs(x)
```

Esim. muunnos pallokoordinaatistoon

```
real x,y,z,r,phi,theta
real, parameter :: pi=3.141592654
x=-1.0 ; y=3.0; z=2.0
r=sqrt(x**2+y**2+z**2)
phi=atan2(y,x)*180.0/pi
theta=asin(z/r)*180.0/pi
```

Vertailuoperaattorit

```
==   .eq.
/=   .ne.
<    .lt.
<=   .le.
>    .gt.
>=   .ge.
```

Huom: `=` on sijoitusoperaattori; vertailu on `==`.

```
integer n
logical d
d = n == 100*(n/100)    ! tosi, jos n jaollinen 100:lla
d = n .eq. 100*(n/100)
```

Loogiset vakiot

```
.true. .false.
```

Loogiset operaattorit

```
.and.
.or.
.not.
```

`X.and.Y` on tosi, jos ja vain jos `X==.true.` ja `Y==.true.`.

`X.or.Y` on tosi, jos `X==.true.` tai `Y==.true.` tai molemmat ovat tosia.

`.not.X` on tosi, jos `X==.false.`.

```
logical leap, d4, d100, d400
integer y
 ...
d4 = y==4*(y/4)
d100 = y==100*(y/100)
d400 = y==400*(y/400)
leap = d4.and.(.not.d100 .or. d400)
```

Peruskontrollirakenteet: Peräkkäisyys

Tavallisesti kukin lause kirjoitetaan omalle rivilleen, jolloin ei tarvita mitään erotinta:

```
x=1.0
y=exp(-x**2/2)
z=1-y
```

Samalla rivillä voi olla useita lauseita, jotka erotetaan puolipisteellä:

```
x = 1.0 ; y = 2.0 ; z = 0.1
```

Peruskontrollirakenteet: Valinta

Yksi vaihtoehto:

```
if (x > 0.0) y = 1/x

if (x > 0.0 .and. x < 100.0) y=exp(x)

if (x > 0.0) then
  y=1/x
  z=log(x)
end if
```

Lause(et) suoritetaan vain, jos ehto on voimassa, muulloin ei tehdä mitään

Kaksi vaihtoehtoa:

```
if (x > 0.0) then
  y=1/x
else
  y=0.0
end if
```

Useampia vaihtoehtoja:

```
if (x > 0.0) then
  y=log(x)
else if (x < 0.0) then
  y=-log(abs(x))
else
  y=0.0
end if
```

Peruskontrollirakenteet: Toisto

Kiinteä kierrosmäärä:

```
sum=0.0
do i=1,100
  sum=sum+i
end do
```

Lause `sum=sum+i` suoritetaan toistomuuttujan `i` arvoilla 1, 2, ... 100.

Toistomuuttujan askel voi olla myös jokin muu kokonaisluku:

```
sum=0.0
do i=0,100,2  ! parillisten lukujen summa
  sum=sum+i
end do
```

Toisto, kunnes lopetusehto toteutuu:

```
x=0.2
sum=0.0
term=1.0
do while (term > 0.0001)
  sum = sum+term
  term = term*x
end do
```

Missä tahansa toistolauseessa iteraatiokierros voidaan keskeyttää ja aloittaa seuraava kierros lauseella `cycle` (kuten C:n continue):

```
s=0.0
do i=1,100
  read (5, *) x
  if (x <= 0.0) cycle
  s=s+log(x)
end do
```

$n + 1/2$ kierroksen silmukka: Muodollisesti ikuinen silmukka, jonka lopetusehtoa tutkitaan jossakin silmukan sisällä. Silmukasta voidaan poistua lopulisesti `exit`-lauseella (kuten C:n break):

```
x0 = 0.5
do
 x1=(1.0+x0)**0.2
 ! lopetetaan, jos tarkkuus saavutettu
 if (abs(x1-x0) < 0.0001) exit
 x0 = x1
end do
```

Lopetusehtoja voi olla useita eri paikoissa silmukkaa:

```
x0 = 0.5
n=0
do
 x1=(1.0+x0)**0.2
 if (abs(x1-x0) < 0.0001) exit
 n=n+1
 if (n > 100) exit
 x0=x1
end do
```

Syöttö ja tulostus

```
read (laitenumero, formaatti) muuttujaluettelo
write (laitenumero, formaatti) muuttujaluettelo

open (laitenumero, tiedoston attribuutit)
close (laitenumero)
```

Kullakin tiedostolla on yksikäsitteinen laitenumero (LUN, logical unit number).

Perinteisesti 5=kortinlukija (nykyisin yleensä pääte), 6=rivikirjoitin (se sama pääte).

Formaatti on merkkijono, joka määrittelee, miten tulostus muotoillaan. Vapaan formaatin merkki on `*`.

```
read(5,*) x,y
z=x+y
write (6,*) x,y,z
```