Merkkijonot

```
program koe
  character a='A'
  character (len=3) :: b='xyz'
  character (len=6) :: c,d,e
  c = b//a//'qq'  ! katenaatio
  d = c
  d(1:2)='--'
  e='X'
  write(6,*) a,b,c,d,e
end program
```

Ohjelman tulostus

```
A
xyz
xyzAqq
--zAqq
X
```

Merkkijonot tulostuslauseen muotoimessa:

```
write(*,'("x=",F10.4)') x
write(*,(''x='',F10.4)') x
```

Merkkijonon jatkaminen useammalle riville:

```
character (len=100)  :: s
s="alkuosa&
 &loppuosa"

write(*,'("x=",F10.4/&
      &"y=",F10.4)') x, y
```

Sekalaisia täydennyksiä

Kontrollirakenteet: case-lause

```
select case (n)
case (0)
  x=0.0
case (1:5)
  x=1.0
end case

character code

select case (code)
case ('a')
  x=0.0
case ('b':'z')
```

```
  x=1.0
case default
  x=2.0
end case
```

Valinnan oltava yksikäsitteinen.

Jos mikään ehdoista ei ole voimassa, ei tehdä mitään.

Valitsin (n) voi olla vain tyyppiä `integer`, `character` tai `logical`.

Hyppykäsky `goto` tai `go to`:

```
      if (x > 0) goto 100
      ...

   100 continue
```

Osoite on 1–5 numeron merkkijono.

Suositeltavaa, että osoitteellisena lauseena käytetään vain tyhjää lausetta `continue`.

Yleensä hyppykäskyjä ei tarvita.

Nimetyt lauseet:

```
      liikaiso: if (x > 1000) then
              ...
              end if liikaiso
```

Nimi on lähinnä kommentti. Se ei ole osoite, johon voitaisiin hypätä muualta.

```
      iteroi: do while ( .. )

            if (..) exit iteroi

            end do iteroi


      ulompi: do i=1,100
      sisempi: do j=1,100

            if (..) exit ulompi

            end do sisempi
            ...
            end do ulompi
```

Omat muuttujatyypit

Vrt. Pascalin `record` ja C:n `struct`.

Muuttujatyypin äärittely:

```
type star
 real :: ra, dec, magnitude
 character (len=20) :: name
end type
```

Tyyppiä `star` olevien muuttujien määrittely:

```
type (star) :: stella, s1, s2
type (star), dimension(10000) :: catalogue
```

Komponentteihin voi viitata operaattorilla `%`:

```
stella%magnitude = 15.2
r= catalogue(i)%ra
```

Omia tyyppejä voi käyttää sijoituslauseissa:

```
s1 = s2
s2 = catalogue(i)
stella=(0.45199, -29.299, 8.80, 'HD2347')
```

Myös muita operaattoreita voi määritellä (käsitellään myöhemmin).

Dynaaminen muuttujien varaus

Taulukon koko voi riippua syöttötiedoista.

Aliohjelmassa tarvittavan aputaulukon koko tiedetään vasta aliohjelmaa kutsuttaessa.

Fortran 77:
- varataan iso taulukko, joka riittää pahimmassakin tapauksessa, tai
- kutsuva ohjelma välittää aliohjelmille tarvittavat työtilat parametreina

Fortran90: taulukot voidaan varata dynaamisesti suoritusaikana.

Aliohjelman taulukon tila varataan aliohjelmaan tultaessa. Koko voi olla muuttuja:

```
subroutine zz(x,n)
integer, intent(in) :: n
real, dimension (n) :: x
  ...
real, dimension(n,n) :: matrix
   ...
```

Jos taulukon koko saadaan selville vasta suoritusaikana (pääohjelmassa, aliohjelman keskellä), se voidaan varata dynaamisesti:

```
real, allocatable, dimension (:,:) :: matrix
  ...
read(*,*) n
allocate(matrix(n,n))
  ...
```

Funktiolla `allocated` voidan tutkia, onko taulukolle varattu tilaa:

```
if (.not. allocated(matrix)) &
  allocate(matrix(1:10, -10:10))
```

Aliohjelman paikalliset muuttujat katoavat aliohjelman päättyessä. Tila voidaan vapauttaa myös eksplisiittisesti:

```
deallocate(matrix)
```