Fortran 77

eli tuulahdus reikäkorttikauden romantiikkaa

Ulkoasu

Ohjelmateksti on sarakesidonnaista, ja sen muoto on suunniteltu 80 sarakkeen reikäkorteille.

Jos sarakkeella 1 on `C` tai `*`, loppuosa rivistä tulkitaan kommentiksi

Sarakkeet 1–5: osoitekenttä

Sarake 6: jos jotakin muuta kuin välilyönti, kyseessä on jatkorivi

Sarakkeet 7–72: ohjelman lauseet

Sarakkeet 73–: kommentti (käytettiin joskus muinoin esim. reikäkorttien numerointiin). Uudemmat kääntäjät hyväksyvät pitemmät rivit; mahdollisesti ilmoitettava sopivalla kääntäjän optiolla.

Välilyönneillä ei ole mitään merkitystä missään kohdassa; niitä voi olla jopa tunnusten keskellä (!!)

```
      REAL X1 Y(10)
      DIMENSIONU(100)
      DO100I=1,10
      Z=X1
     1Y(I)
      IF(Z.GT.0.0.AND.Z.LT.1.0)T H E N
      U(I)=S
     SQRT(Z)+1 . 23
     11E-1-Z
      EN
     +DIF
100   CONTINUE
```

Muuttujat

Muuttujan nimi saa olla korkeintaan kuuden merkin mittainen. Jotkin kääntäjät sallivat pitemmät nimet, mutta ne eivät kuulu standardiin.

Standardin mukaiset tyypit:

```
integer
real
double precision
complex
logical
character
```

Useimmissa toteutuksissa laajennuksia, kuten

```
integer*4
real*8
logical*1
```

Numero kertoo, kuinka monta tavua muuttujalle varataan. Tämä ei ole standardin mukainen määrittely! Sitä käytetään kuitenkin hyvin yleisesti.

Vanhoissa ohjelmissa luotetaan usein implisiittiseen määrittelyyn. Tästä johtuvat joskus omituiset muuttujien nimet, kuten `amass`, `icount`.

Muuttujat voidaan alustaa `data`-lauseella:

```
real x(3)
data x /1.0, 2*0.5/
```

Useat muuttujat (jopa eri tyyppiset) voivat viitata samaan muistipaikkaan:

```
real z(100),x(3),y
integer nn
equivalence (x,z(10))
equivalence (y,z(100))
equivalence (nn,z)
```

Käyttökohteita:

- Aliohjelmien työtilat (kutsuva ohjelma välittää yhden pitkän työvektorin)
- Vanhoissa Fortraneissa merkkijonojen ym. puuttuvien tyyppien käsittely

Käyttöä ei suositella. Seuraavissa versioissa `equivalence` ei ole enää käytössä.

Ainoa keino globaalin ympäristön välittämiseen on `common`-alue. Toteutustapa vaarallinen: sama muistialue voidaan jakaa eri muuttujien kesken eri tavoin eri aliohjelmissa.

Kontrollirakenteet

Samalla rivillä ei voi olla useita lauseita.

Toistolause on aina osoitteellinen:

```
      do 100 i=1,10
100   x(i)=0.0
```

tai mieluummin:

```
      do 100 i=1,10
      x(i)=0.0
100   continue
```

Kaikkein vanhimmissa versioissa ei ole `if .. then .. end if` -rakennetta; valinta on toteutettava `goto`-lauseilla.

Aritmeettinen `if`:

```
      if (i-j) 100,200,300
100   k=i-j
      goto 400
200   k=0
      goto 400
300   k=j
400   ...
```

Taulukkosyntaksi on F90:n laajennus, vaikkakin mukana monissa vektorikoneiden (kuten Cray) Fortraneissa.

Matemaattiset varusfunktiot ovat olleet jo kauan käytettävissä. Muista varusfunktioista suurin osa vasta F90:ssä.

Lausefunktio (oikeastaan makro):

```
f(x)=sqrt(1-x^2)
...
z=f(0.1)
```

Notaation ongelma (myös F90:ssä):

```
real f
...
y=f(x)
```

Tämä voi tarkoittaa:

- haetaan taulukon `f` alkio
- lasketaan funktio (tai lausefunktio) `f`

Aliohjelmien ja funktioiden parametrit aina viiteparametreja. Ei mitään keinoa varmistaa, että aliohjelma ei muuta niitä.

Aliohjelmat käännetään erikseen. Kääntäjä ei voi varmistaa, että parametrien tyypit ovat oikein, tai että niitä on oikea määrä. Virhe johtaa virheellisiin tuloksiin tai parhaassa tapauksessa ajon kaatumiseen.