Voorbeeld

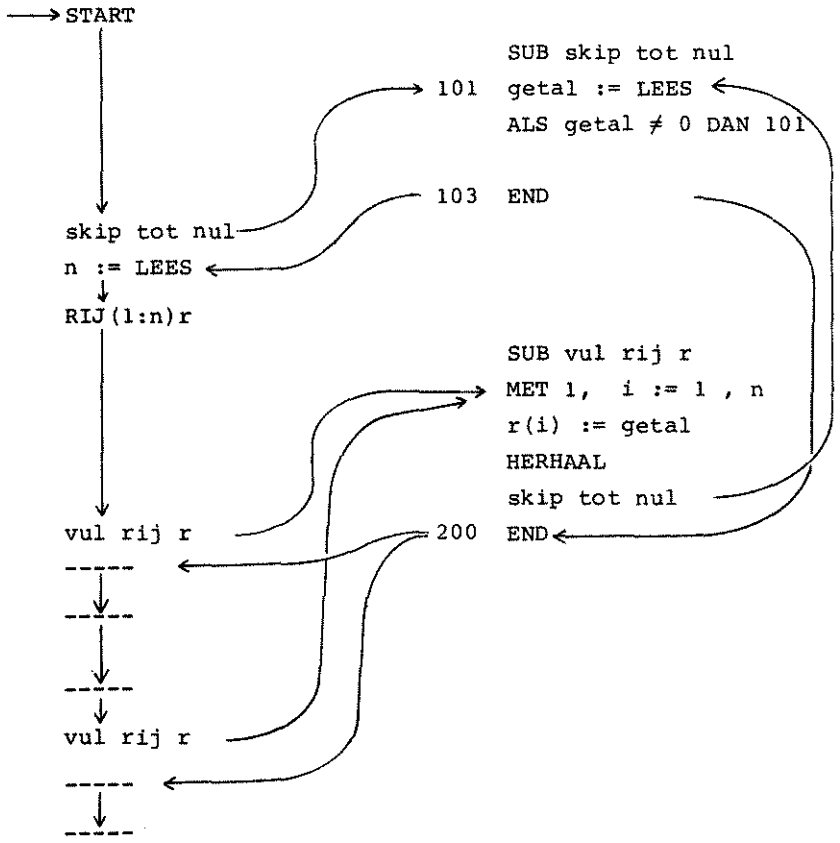
```
SUB vul rij r          ; deze subroutine vult de rij "r" met gelezen getallen
MET 1, i := 1, n
r(i) := LEES
HERHAAL
skip tot nul           ; als de rij gevuld is, worden alle getallen tot en met 0 geskipt.
END
```

Deze SUB-beschrijving moet tekstueel volgen op de reservering van de rij "r" (zie C7.3).

21. SUB-roep

De executie van een SUB-roep betekent, dat het programma van de aangeroepen SUB-beschrijving wordt doorlopen. Zodra daarbij "END" wordt bereikt, wordt teruggekeerd naar de aktie die tekstueel volgt op de SUB-roep. In dit verband wordt ook END als aktie beschouwd (zie in onderstaand voorbeeld de pijl van 103 naar 200).

Een en ander wordt geadstrueerd aan onderstaand stuk programma, waarin de voorbeelden van 20 worden aangeroepen:



N.B. Een SUB-roep lijkt wel wat op een sprong. Men mag echter nooit met een sprongopdracht naar een SUB-beschrijving springen (zie 8). Ook worden rijen, die gereserveerd worden tussen de plaats van SUB-roep en SUB-beschrijving, door de SUB-roep niet ongereserveerd gemaakt (vgl. 13).

22. Informele inleiding tot het FUN mechanisme

De SUB voorziet in de behoefte wanneer men op verschillende plaatsen van het programma precies dezelfde opdrachtenreeks nodig heeft. Heel vaak echter heeft men op verschillende plaatsen van het programma slechts ongeveer dezelfde opdrachten nodig, opdrachten die wel hetzelfde rekenschema representeren, maar dat met telkens andere variabelen uitvoeren. Hierin voorziet de FUN.

In de FUN-beschrijving geeft men het rekenschema weer voor fictieve grootheden (hier "formelen" genoemd), terwijl men dan bij de aanroep zegt wat de werkelijke grootheden zijn waarop geopereerd moet worden (hier "aktuelen" genoemd). Als aktuelen kan men meegeven uitdrukkingen (dus o.a.

variabelen) maar ook RIJ-namen. In de FUN-beschrijving moet men aangeven wat voor soort aktuelen men wenst mee te geven.

Voorbeeld van FUN-beschrijving

```
FUN som(h,k,RIJ a)
    hulp := 0
    MET 1,  i:=h,  k
    hulp := hulp + a(i)
    HERHAAL
    FUN := hulp
```

Deze FUN bepaalt voor een meegegeven RIJ, die we hier maar zolang a noemen ("formele" naam) de som van het h-de tot en met het k-de element (h en k zijn ook formelen). De aanduiding RIJ vóór a geeft aan dat we bij de aanroep als derde parameter de naam van een rij mee moeten geven. Het ontbreken van deze aanduiding bij h en k geeft aan dat men bij de aanroep op de eerste en tweede plaats een uitdrukking moet zetten. Nu een

Voorbeeld van aanroep van deze FUN vanuit het hoofdprogramma

```
-----
-----
RIJ(1:500)m
-----
-----
p := som(6, 34, m)
q := som(3, m(6), m)
```

Met de eerste aanroep worden de elementen 6 t/m 34 van de rij m opgeteld, en de som wordt aan p als waarde toegekend. Bij de tweede aanroep wordt de som van de elementen 3 t/m m(6) aan q toegekend. Dat m, een rijnaam, ook nog eens op de plaats van de tweede parameter voorkomt, terwijl de tweede parameterplaats in de FUN-beschrijving niet met RIJ is gemerkt, is niet onjuist: immers de tweede aktuele luidt m(6), dat is dus gewoon een variabele, dus een uitdrukking.

Een heel belangrijke, en ook doorgaans prettige, eigenschap van de FUN is dat alle er in optredende labels en namen (behalve de namen van andere FUN's) lokaal zijn, d.w.z. niets te maken hebben met identieke labels en namen die mogelijk in andere FUN's of in het hoofdprogramma gebruikt worden. En in een bepaalde FUN-beschrijving hoeft men met de namen van andere FUN's alleen nog maar rekening te houden voor zover men die andere FUN's vanuit deze FUN-beschrijving wenst aan te roepen. De namen van FUN's die men niet wenst aan te roepen mag men rustig als (lokale) GP's of formelen gebruiken.

Bij het schrijven van een FUN kan men dus de te gebruiken labels en namen (behalve namen van aan te roepen FUN's) vrij kiezen, ongeacht welke namen in andere FUN-beschrijvingen of in het hoofdprogramma gebruikt worden. Dit maakt het o.a. mogelijk voor vaak voorkomende ingewikkelde processen (bijv. het oplossen van stelsels vergelijkingen) FUN-beschrijvingen te maken die men aan elk willekeurig programma kan toevoegen.

23. FUN-beschrijving

Volgens [86] t/m [89] begint een FUN-beschrijving met FUN gevolgd door de naam die men aan de FUN wenst te geven, gevolgd door een rijtje formelen, minstens één, van elkaar gescheiden door komma's dit rijtje gevat in haakjes. Elke formele heeft de gedaante [GP] (d.w.z. [naam]) of RIJ [GP-rij] (d.w.z. het woordje RIJ gevolgd door een [naam]).
Na deze "kop" van de FUN-beschrijving komen [opdrachten], d.w.z. dat RIJ-reserveringen en SUB-beschrijvingen niet zijn toegestaan. Maar ook zijn SUB-roepen niet toegestaan, zoals aanstonds zal blijken.
De FUN-beschrijving eindigt tenslotte met FUN := [uitdr] (en niet met END, zoals de SUB-beschrijving).

Een zelfgeschreven FUN moet dus, in afwijking van de standaardfunkties, minstens één parameter hebben, ook al heeft men die eigenlijk helemaal niet nodig. Men is evenwel niet verplicht deze

parameter in de rest van de FUN-beschrijving ook te gebruiken. Overigens zou men met parameterloze, procedures ook niet erg veel kunnen aanvangen, enkele bijzondere gevallen daargelaten.
Evenzo moet een FUN-beschrijving eindigen met FUN :=---, ook al is men helemaal niet van plan van de aldus toegekende waarde gebruik te maken, bijv. doordat de FUN-roepen van deze FUN nooit in een [uitdr] maar altijd als [opdracht] plaats hebben (zie [23] en [37]).

De verschillende [FUN-beschrijvingen] zijn dermate onafhankelijk van elkaar dat men voor elke [FUN-beschrijving] apart bij het in tekstuele volgorde lezen weer zekerheid heeft over de status van een [naam] bij het eerste optreden ervan.

Context-regels voor het funktiedeel

Voor elke [FUN-beschrijving] afzonderlijk geldt:

C23.1 Een en dezelfde [naam] heeft overal in deze [FUN-beschrijving] dezelfde betekenis.

C23.2 Een [naam] is een [GP-rij] als hij in deze [FUN-beschrijving] optreedt op de plaats van [formele] voorafgegaan door de aanduiding RIJ (zie [86] en [89]).

C23.3 Een [naam] is een [FUN-naam] als hij in deze [FUN-beschrijving] voorkomt gevolgd door een openingshaakje (spaties spelen geen rol), en niet valt onder C23.2.

C23.4 Alle overige namen zijn [GP]'s voor deze [FUN-beschrijving].

C23.5 Alle [labels] in een [FUN-beschrijving] moeten van elkaar verschillen.

C23.6 Alle [namen] in deze [FUN-beschrijving] die geen [FUN-naam] zijn, en alle [labels] hebben niets te maken met eventuele identieke [namen] en [labels] elders in het programma.

Voor het funktiedeel als geheel geldt:

C23.7 Elke [naam] die in een of meer der [FUN-beschrijvingen] als [FUN-naam] optreedt moet ook precies éénmaal optreden op de plaats van [FUN-naam] in een [FUN-beschrijving].

Uit deze context-regels volgt nog dat SUB-roepen vanuit een FUN-beschrijving onmogelijk zijn, want SUB-namen zijn namen in het hoofdprogramma.

Voorbeeld

```
        FUN f(x)      ; Hiermee is "f" voor het gehele programma, inklusief het funktiedeel, een
                      ; [FUN-naam], "x" is, alleen voor deze FUN-beschrijving, een formele [GP]
        y := x        ; Hiermee is "y", echter alleen voor deze FUN-beschrijving, een zg. "lokale"
                      ; [GP]
1       x := r(y)     ; "r" kan onmogelijk de naam zijn van een [GP-rij] want dan zou "r" als zo-
                      ; danig formeel moeten zijn. De compiler vertaalt "r" nu als de naam van een
                      ; FUN met 1 formele [GP], waarvan de beschrijving nog moet volgen
                      ; De label "1" is ook "lokaal", dat wil zeggen: geldt alleen voor deze FUN-
                      ; beschrijving
        FUN := x+y    ; Hiermee wordt deze FUN-beschrijving afgesloten
                      ; De betekenis van de namen "x" en "y" en van de label "1" gaan nu verloren,
                      ; n i e t  echter de betekenis van de namen "f" en "r" als FUN-namen
                      ; De compiler verwacht nog de beschrijving van "r".
        FUN r(x,RIJp); FOUT! "r" mocht alleen maar 1 formele [GP] hebben
        -----
        -----
        FUN := ----
        FUN r(x)      ; Dat verwacht de compiler (zie overigens C23.7)
        -----
        -----
        FUN :=----
```

24. FUN-roep

De executie van een FUN-roep betekent, dat het programma van de aangeroepen FUN-beschrijving wordt doorlopen. Zodra daarbij "FUN :=[uitdr]" wordt bereikt, wordt teruggekeerd naar het punt vanwaar de funktie werd aangeroepen. Wanneer dit plaats vond in een [uitdr] , is de waarde van de FUN-roep, de aan "FUN" toegekende waarde; wanneer de funktie als [opdracht] werd geroepen, speelt deze waarde verder geen rol (zie voorbeeld 24.2).

In een FUN-roep moet op de plaats van een formele [GP] als [aktuele] een [uitdr] worden meegegeven en op de plaats van een formele [GP-rij] een (aktuele) [GP-rij] (niet voorafgegaan door de aanduiding RIJ). Deze rij dient ten tijde van de aanroep gereserveerd te zijn.
Als de aktuele [uitdr] een [variabele] is, wordt bij executie het adres van die [variabele] aan de funktie doorgegeven, anders wordt die [uitdr] uitgerekend en in een voor de programmeur onbereikbare hulp-GP gestopt waarvan bij executie ook het adres aan de funktie wordt meegegeven.
De waarden der bedoelde [uitdr]ukkingen en de adressen der [variabelen] worden weer bepaald in de tekstuele volgorde der parameters (vgl. 10.6).

Omdat bij executie het adres van de aktuele wordt meegegeven opereert de FUN ook echt op die aktuele. Wordt in een FUN-beschrijving een waarde toegekend aan een formele GP, dan wordt bij aanroep aan de aktuele GP een waarde toegekend. Zo kan de FUN, ondanks context-regel C23.6 toch via de parameters waarden van GP's in het hoofdprogramma veranderen.
Evenzo wordt bij een aktuele [GP-rij] de descriptor van de rij meegegeven, zodat weer op de echte rij-elementen wordt geopereerd.

Voorbeeld 24.1

```
FUN f(x,y)
x := 3
FUN := y
```

Met een aanroep f(a,b) krijgt a de waarde 3. Met een aanroep f(a+b,c) of f(17,q) krijgt de hulp-GP, waar de waarde van a+b resp 17 ingestopt was, de waarde 3, maar met a, b of overige getallen 17 in het programma gebeurt niets. Interessant is nog de aanroep f(i,d(i)). Als bij aanroep i bijv. de waarde 5 heeft, krijgt FUN de waarde d(5), en niet d(3), zoals men misschien zou denken (omdat i door FUN de waarde 3 krijgt). Immers bij aanroep wordt het adres van de parameters meegegeven, dus het adres van d(5).
Als de FUN-beschrijving daarentegen geluid had

```
FUN g(x,RIJ y)
x := 3
FUN := y(x)
```

dan zou bij aanroep met g(i,d) (d een op dat moment gereserveerde rij) FUN de waarde d(3) hebben gekregen.

Voorbeeld 24.2

```
    FUN max(index,RIJ r)        ; deze funktie levert het grootste getal van de rij  r en kent
                                ; aan "index" de waarde toe van de index waarvoor dat maximum
                                ; wordt bereikt
    h := ONDIN(r)               ; h := onderindex van r, zie 18
    k := BOVIN(r)               ; k := bovenindex van r, zie 18
    index := h
    probeer := r(h)
    MET l, i := h+l, k
    volgende := r(i)
    ALS volgende ≤ probeer DAN l
    index := i
    probeer := r(i)
  l HERHAAL
    FUN := probeer
```