TECHNISCHE HOGESCHOOL EINDHOVEN
Afdeling Algemene Wetenschappen
Onderafdeling der Wiskunde

Elementaire cursus
PROGRAMMEREN in ALGOL 60

Prof. Dr. E.W. Dijkstra

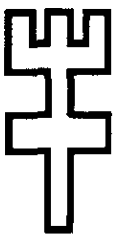
Voorjaarssemester 1967

Onderafdeling der Wiskunde

Elementaire cursus programmeren in ALGOL 60

BEHORENDE BIJ HET COLLEGE
WISKUNDE IV b
VAN PROF. DR. E.W. DIJKSTRA

SAMENGESTELD DOOR
DRS. A.J. GEURTS EN
DRS. A.N. HABERMANN



VOORJAARSSEMESTER 1967

TECHNISCHE HOGESCHOOL EINDHOVEN

DICT. NR. 212
PRIJS f 1,--

Inhoudsbeschrijving

Elementaire cursus programmeren in ALGOL 60

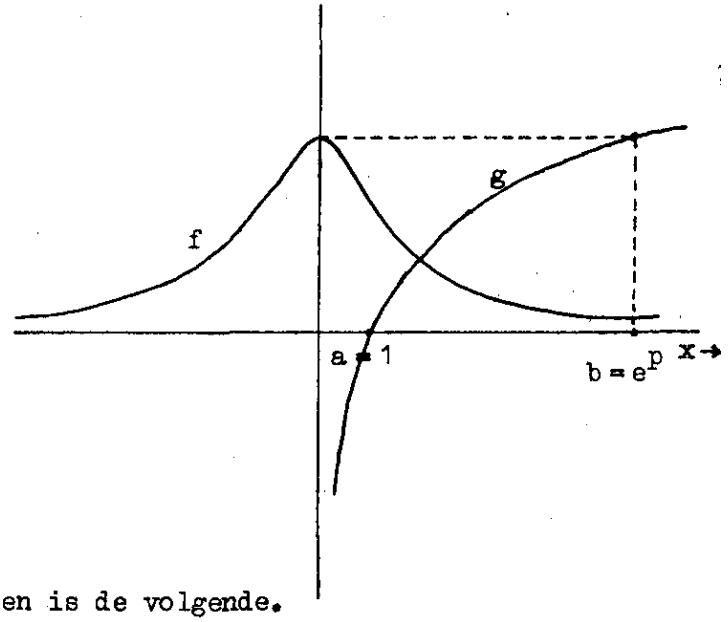
E.W. Dijkstra

Voorjaar 1967

JdG, 5 December 2005

INLEIDING

ALGOL (een samentrekking van ALGOrithmic Language) is de naam van een taal, die ontworpen is met het doel rekenprocessen te beschrijven. Deze taal is zo precies gedefinieerd dat een rekenmachine een in ALGOL geformuleerd reken-proces kan uitvoeren.



Voorbeeld.
Beschouw het volgende probleem:
Bepaal in drie decimalen nauwkeurig het snijpunt (x_s, y_s) van de krommen

$$f(x) = \frac{p}{1 + x^2} \quad \text{en} \quad g(x) = \log x$$

bij gegeven positieve waarde van de parameter p.

Een methode om dit probleem op te lossen is de volgende.
Het is zeker dat $f(1) > g(1)$ en $f(e^p) < g(e^p)$.
Zij $a = 1$ en $b = e^p$ dan ligt dus x_s in $[a, b]$.
Zij $c = \frac{1}{2}(a + b)$. Als $f(c) < g(c)$ dan ligt x_s in $[a, c]$.
In dit geval nemen we nu $a = 1$ en $b = c$.
Als echter $f(c) \geq g(c)$ dan ligt x_s in $[c, b]$. Dan nemen we $a = c$ en $b = e^p$.
In beide gevallen geldt weer x_s ligt in $[a, b]$. De lengte van het nieuwe interval $[a, b]$ is de helft van de lengte van het oude interval $[a, b]$. Dit proces wordt herhaald tot de lengte van $[a, b]$ kleiner dan 10^{-3} is geworden. Dan is $x_s = \frac{1}{2}(a + b)$ en $y_s = \frac{p}{1 + x_s^2}$ het gevraagde snijpunt met de gewenste nauwkeurigheid. De formulering van dit rekenproces in ALGOL luidt (voor het geval p = 2):

```
begin   real a, b, c, xs, ys, p;
        p := 2; a := 1; b := exp(p);
OPNIEUW: c := (a + b) / 2;
        if ln(c) < p / (1 + c × c) then a := c else b := c;
        if (b - a) ≥ 10^-3 then goto OPNIEUW;
        xs := (a + b) / 2; ys := p / (1 + xs × xs);
        print (p, xs, ys)
end
```

§1. ASSIGNMENT STATEMENT

In het voorbeeld hebben we voor p de waarde 2 genomen. In de ALGOL tekst staat niet p = 2 maar

```
p := 2
```

Het hier gebruikte symbool := (uit te spreken als "wordt") is een toekenningsteken en niet een gelijkteken. Het betekent dat aan de variabele in het linkerlid de waarde 2 wordt toegekend. Een dergelijke formule heet dan ook een assignment statement. Door middel van een assignment statement wordt aan een variabele een waarde toegekend.
Het gebruikelijk symbool = is door het symbool := vervangen om de nadruk te leggen op de asymmetrie. De variabele in het linkerlid krijgt z'n waarde van het rechterlid. Bovendien wordt door het symbool := tot uitdrukking gebracht dat het hier gaat om een handeling die uitgevoerd moet worden en niet, zoals bij het symbool =, om een relatie waaraan al of niet voldaan kan zijn. Zo wordt door de assignment statement

```
i := i + 1
```

de waarde van i met 1 opgehoogd, terwijl de relatie

```
i = i + 1
```

voor geen enkele waarde van i juist is.
Een ander voorbeeld van een assignment statement is

```
xs := (a + b) / 2
```

Deze assignment statement geeft aan xs de waarde die verkregen wordt door het rechterlid uit te rekenen. Deze assignment statement is alleen uitvoerbaar als a en b inmiddels (in vorige assignment statements) een waarde hebben gekregen.

§ 2. IDENTIFIERS EN NUMBERS

In het voorbeeld hebben we variabelen gebruikt welke we namen hebben gegeven (a, b, xs, enz.).
In ALGOL worden identifiers[1] (namen) o.a. gebruikt om variables (variabelen) aan te duiden In het vervolg zullen we zien dat ook bepaalde plaatsen in het programma of soms zelfs hele processen door identifiers worden aangeduid.

Een identifier is een opeenvolging van letters (hoofdletters en kleine letters) en cijfers, te beginnen met een letter.

Voorbeelden van identifiers:	Geen identifiers zijn:
Wiskunde 4b	4b
lste keer	1ste keer
auto	auto's
H2SO4	PV = RT
Xn	Xn - 1
a23	a23.2
Den Haag	's-Gravenhage

In ALGOL geldt de afspraak dat spaties en overgang op een nieuwe regel geen betekenis hebben. Zo mag de identifier "ALGOL 60" ook geschreven worden als "ALG OL60" of "A L G O L 6 0'.

In het rechterlid van een assignment statement kunnen behalve variabelen ook getallen voorkomen. De getallen worden normaal in het tientallig stelsel genoteerd, maar de schrijfwijze is ook hier gereglementeerd.

Numbers (getallen) worden onderscheiden in twee typen aangegeven door de symbolen integer en real.

Naast letters, cijfers en symbolen als + , - , (, zijn bepaalde onderstreepte woorden ook ALGOL symbolen. Door de onderstreping verandert het woord in een nieuw ALGOL symbool. Bijvoorbeeld, real moet men niet lezen als een identifier, welke bestaat uit 4 onderstreepte letters maar als één zelfstandig symbool.

1) We gebruiken in de tekst de vaktermen zoals die gebezigd worden in het Revised Report on the Algorithmic Language ALGOL 60.

Een integer is een opeenvolging van cijfers al of niet voorafgegaan door een + of - teken. Een integer is van het type integer.
Ieder ander number is van het type real ook al is z'n waarde geheel (bv. 42 is een integer maar 42.0 is van het type real). Om de notatieregels voor een number van het type real te beschrijven voeren we eerst de begrippen decimal fraction en exponent part in.

Een decimal fraction is een decimale punt gevolgd door één of meer cijfers.
Een exponent part is het symbool $_{10}$ gevolgd door een integer.

Mogelijke vormen van een number van het type real zijn:
1) integer, gevolgd door decimal fraction, gevolgd door exponent part;
2) integer, gevolgd door decimal fraction;
3) integer, gevolgd door exponent part;
4) decimal fraction, eventueel voorafgegaan door een + of - teken, gevolgd door exponent part;
5) decimal fraction, eventueel voorafgegaan door een + of - teken;
6) exponent part, eventueel voorafgegaan door een + of - teken.

Voorbeelden van numbers:	Geen numbers zijn:
- 17	17a
+ 23.2_{10} + 3	23. + 2_{10} + 3
- 14.231	- 14.23.1
0.137	$12_{-10}2$
$12_{10}2$	$a_{10}3$
- $.451_{10}$ - 1	$2_{10}a$
.137	$15.21_{10}3.5$
+ .137	
$_{10}$ - 3	- $._{10}3$
- $_{10}$ - 3	$12._{10}$ - 2
2.0	2.

N.B. Let er op dat $12_{10}2$ van het type real is, al is z'n waarde geheel.

§ 3. SIMPLE ARITHMETIC EXPRESSIONS

Een simple arithmetic expression kan worden opgebouwd uit variables, numbers, arithmetic operators en ronde haakjes.

Arithmetic operators zijn:

+	(optelling)
-	(aftrekking)
×	(vermenigvuldiging)
/	(deling)
↑	(machtsverheffing)
÷	(gehele deling)

Een vermenigvuldiging wordt aangegeven door het teken ×. Dit teken mag niet worden weggelaten. Het is dus niet toegestaan het product a × b als ab te schrijven; ab is immers een identifier. Ook a . b is niet toegestaan, de punt wordt alleen als decimale punt gebruikt. In handgeschreven tekst verdient het aanbeveling om in plaats van × te schrijven *, om verwarring met de letter x te voorkomen.
De gebruikelijke prioriteitsregels gelden. Machtsverheffing heeft de hoogste prioriteit, dan volgen vermenigvuldiging en deling en tenslotte optelling en aftrekking. Verder worden de operaties van links naar rechts uitgevoerd.
In expressies wordt zo nodig gebruik gemaakt van ronde haakjes, maar niet van accolades of rechte haken.

De exponent van een machtsverheffing wordt niet hoger geschreven dan het grondtal. Bijvoorbeeld a^2 wordt in ALGOL a ↑ 2.
Komt in de exponent een arithmetic operator voor dan moeten (op grond van de prioriteitsregels) haakjes gebruikt worden. Bijvoorbeeld a^{2p} wordt in ALGOL a ↑(2 × p). Twee arithmetic operators mogen niet naast elkaar staan. Dus voor a^{-2} schrijft men a ↑(-2) en niet a ↑ -2.

Het gebruik van, strikt genomen, overbodige haakjes kan de duidelijkheid van de ALGOL tekst verhogen. Bijvoorbeeld

a × b / c × d is equivalent met ((a × b) / c) × d .

Deze laatste schrijfwijze geeft duidelijk aan dat niet bedoeld wordt

(a × b) / (c × d).

Voor gehele getallen is er naast de gewone deling (operator /) ook nog de moge-

lijkheid van de gehele deling (operator $\div$). Als voor twee gehele getallen m en n geldt

$$|m| = q \times |n| + r \quad \text{waarbij } 0 \leqslant r < |n|$$

dan is $m \div n = \text{sign}(m \times n) \times q$

Voorbeelden.

Wiskundige formule	incorrecte expressie	correcte ALGOL expressie
$(a + b)(c + d)$	(a + b)(c + d)	(a + b) × (c + d)
$2n - 1$	2n - 1	2 × n - 1
$\frac{1}{1 + x}$	1 / 1 + x	1 /(1 + x)
$(2^3)^5$	2 ↑(3 ↑ 5)	(2 ↑ 3)↑ 5 of 2 ↑ 3 ↑ 5
2^{3^5}	2 ↑ 3 ↑ 5	2 ↑(3 ↑ 5)
$x \cdot \frac{-1}{(a + b)c}$	x × - 1 /(a + b) × c x ×(- 1) /(a + b) × c	x ×(- 1) /((a + b) × c)
$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$	ad - bc	a × d - b × c

Opmerking: in de middelste kolom zijn de 3^e, 4^e, 5^e, 7^e en 8^e expressie wel correct ALGOL, maar geen correcte weergave van de wiskundige formule.

De eenvoudigste vorm van een arithmetic expression is een number of een variable. Er zijn, behalve de behandelde simple arithmetic expressions, nog andere vormen van arithmetic expressions.
We gaan hier verder niet op in.

In § 1 is het begrip assignment statement genoemd.
We zijn nu in staat nader te formuleren wat een assignment statement is. Een assignment statement is
een variable, gevolgd door het symbool := , gevolgd door een arithmetic expression.
Er bestaat een tweede vorm van de assignment statement (zie blz. 11 onderaan).

§ 4. HET TYPE VAN VARIABLES EN SIMPLE ARITHMETIC EXPRESSIONS

We hebben gezien dat numbers door hun representatie van een bepaald type zijn, nl. real of integer.
De variables zullen van een bepaald type worden verklaard door middel van een zogenaamde type declaration.
Een dergelijk type declaration bestaat uit het symbool real of integer gevolgd door een of meer identifiers, welke onderling gescheiden zijn door een komma.

Voorbeelden.

```
real a, b, c, xs
integer n, m
```

Declarations worden onderling en van statements gescheiden door een puntkomma. Een puntkomma wordt ook gebruikt om statements te scheiden. Een scheidingsteken is noodzakelijk omdat spatie's en overgang op een nieuwe regel geen betekenis hebben in ALGOL.

De integer declaration houdt in dat de zo gedeclareerde variables slechts gehele waarden kunnen krijgen.
Een real gedeclareerde variable kan een willekeurige reële waarde krijgen. We moeten ons bij real gedeclareerde variables getallen denken, die slechts in eindige precisie gegeven zijn. De waarde van een integer gedeclareerde variable daarentegen wordt exact weergegeven.

De waarde van een **simple arithmetic** expression is een getal van een bepaald type, dat afhangt van de operators en van het type van de numbers en variables in de expressie.

- a + b, a - b, a × b — is van het type integer als zowel a als b van het type integer zijn, in alle andere gevallen van het type real.
- a / b — is van het type real.
- a ÷ b — is alleen gedefinieerd als a en b beide van het type integer zijn, en is dan van het type integer.
- a ↑ b — is van het type integer als a en b beide van het type integer zijn en bovendien b ≥ 0, in alle andere gevallen van het type real.

Bij de assignment statement

variable := arithmetic expression

of v := AE

is de toekenning duidelijk wanneer v en AE van hetzelfde type zijn.
Het is toegestaan dat het type van het linkerlid en van het rechterlid verschillend zijn.
Is v van het type real en AE van het type integer, dan wordt de waarde van AE als getal van het type real aan v toegekend.
Is v van het type integer en AE van het type real, dan wordt de waarde van AE afgerond op het dichtstbijzijnde gehele getal en als integer aan v toegekend.
Is de waarde van AE binnen de rekennauwkeurigheid gelijk aan $n + \frac{1}{2}$, dan is op grond van de eindige precisie van de arithmetiek de waarde van v niet gedefinieerd. Deze kan nl. n zijn maar ook n+1.

§ 5. GOTO STATEMENT

In het algemeen zullen de statements van een ALGOL programma in de geschreven volgorde worden uitgevoerd.
Een goto statement wijzigt deze volgorde en wijst de statement aan, die als volgende wordt uitgevoerd.
Om een statement te kunnen aangeven wordt deze voorzien van een label. Een label is een identifier. Een statement wordt gelabeld door vóór de statement deze identifier, gevolgd door een dubbele punt te plaatsen. Een label wordt als zodanig niet gedeclareerd.

Een goto statement bestaat uit het symbool goto, gevolgd door een identifier. Deze identifier moet elders in het programma als label voorkomen.

Voorbeeld. We willen de som van de reeks $\sum_{n=1}^{\infty} \frac{1}{n^2}$ berekenen.

```
begin  integer n; real som;
       som := 0; n := 1;
AGAIN: som := som + 1 / n ↑ 2;
       n := n + 1;
       goto AGAIN
end
```

Elk ALGOL programma begint met het symbool begin en eindigt met het symbool end. De declarations moeten staan ná begin en vóór de eerste statement. In deze declarations mag iedere identifier slechts één keer voorkomen.

§ 6. CONDITIONAL STATEMENT

Het laatste voorbeeld heeft het practische bezwaar dat het rekenproces nooit eindigt. Dit komt omdat de goto statement de volgorde van uitvoering onvoorwaardelijk wijzigt.
Wil men bijvoorbeeld berekenen

$$\sum_{n=1}^{1000} \frac{1}{n^2}$$

dan moet de goto statement 999 keer wèl, maar de duizendste keer niet worden uitgevoerd.
De mogelijkheid daartoe geeft de if statement.

Voorbeeld.[1)]

```
begin  integer n; real som;
       som := 0; n := 1;
AGAIN: som := som + 1 / n ↑ 2;
       n := n + 1;
       if n ≤ 1000 then goto AGAIN;
       print (som)
end
```

Een if statement bestaat uit een if clause gevolgd door een statement. De if clause bestaat uit het symbool if gevolgd door de voorwaarde op grond waarvan de statement al of niet moet worden uitgevoerd, gevolgd door het symbool then.
De voorwaarde heet een Boolean expression.

Een vorm van een Boolean expression is een relation.

Een relation is een simple arithmetic expression, gevolgd door een relational operator, gevolgd door een simple arithmetic expression.

1) De betekenis van de statement print (E), waarin E een willekeurige arithmetic expression mag zijn, is "typ de waarde van E".
Om aan een variabele x een waarde toe te kennen, die op ponsband staat, zullen we gebruik maken van de assignment statement x := read. Deze assignment statement betekent "de waarde van x wordt het volgende getal op de band".
Deze assignment statements worden behandeld in § 10.

Relational operators zijn

$<$	(kleiner dan)
$\leqslant$	(kleiner dan of gelijk aan)
$=$	(gelijk aan)
$\geqslant$	(groter dan of gelijk aan)
$>$	(groter dan)
$\neq$	(niet gelijk aan).

Voorbeelden.

$a = b$

$x + 1 \leqslant y \uparrow 2$

$x \times (y \uparrow 2 + z \uparrow 2) > -1$

maar niet $0 \leqslant x \leqslant 1$

De waarde van een Boolean expression kan zijn true of false.
Blijkt na de berekening van de simple arithmetic expressions dat de relatie juist is, dan krijgt de Boolean expression de waarde true, in het andere geval de waarde false.

Voorbeelden.

Boolean expression	waarde	
$1 \geqslant 2$	false	
$a = 2$	true	als $a = 2$
	false	als $a \neq 2$
$x \geqslant x + 10$	false	voor iedere waarde van x
$(x + 3) \times (x - 2) \geqslant 0$	true	als $x \geqslant 2$ of $x \leqslant -3$
	false	als $-3 < x < 2$

Naast variables van het type integer of real zijn er in ALGOL ook variables van het type Boolean.
Een variable van het type Boolean moet gedeclareerd worden door een type declaration. Een dergelijke type declaration bestaat uit het symbool Boolean, gevolgd door één of meer identifiers, onderling gescheiden door een komma. De waarde van een variable van het type Boolean is true of false. Zo'n variable krijgt zijn waarde door middel van een assignment statement, waarbij rechts een Boolean expression staat. Dit is de tweede vorm van een assignment statement.

De eenvoudigste vorm van een Boolean expression is true, false of een variable van het type Boolean. Deze, en de relations zijn de enige vormen welke wij behandelen.

Voorbeeld.

```
real x, y; Boolean b, c, d;
x := 2; y := 3.14;
b := true;
c := x > y;
d := b
```

De Boolean expression in de if clause van een if statement heeft het effect dat de statement volgend op then slechts wordt uitgevoerd, indien de waarde van de Boolean expression true is. Is de waarde false dan wordt de statement volgend op then overgeslagen.

Voorbeeld.
De waarde van de variable x wordt door de if statement

```
if x < 0 then x := - x
```

gelijk aan de absolute waarde van x.

Behalve een goto- of assignment statement kan de statement volgend op de if clause ook een compound statement zijn.

Een compound statement bestaat uit een aantal statements, voorafgegaan door het symbool begin en gevolgd door het symbool end. De symbolen begin en end heten statement haken.

Voorbeeld.

```
S0; if B then begin S1; S2 end; S3; S4
```

Heeft B de waarde true, dan worden achtereenvolgens uitgevoerd S0, S1, S2, S3, S4. Heeft B de waarde false, dan worden achtereenvolgens uitgevoerd S0, S3, S4.

Een if statement is één van de twee vormen van een conditional statement die

wij behandelen. De andere vorm van een conditional statement is een if statement, gevolgd door het symbool else, gevolgd door een statement.
De conditional statement

if B then S1 else S2

is gelijk aan de statement S1, als B de waarde true heeft, en gelijk aan de statement S2, als B de waarde false heeft.

Voorbeelden.

1. Voor gegeven waarden van x kan men de berekening van $y = |x + 2|$ beschrijven door middel van de statement

   if x ⩾ - 2 then y := x + 2 else y := - x - 2

2. S0; if B then begin S1; S2 end else S3; S4

   Heeft B de waarde true, dan worden achtereenvolgens uitgevoerd S0, S1, S2, S4.
   Heeft B de waarde false, dan worden achtereenvolgens uitgevoerd S0, S3, S4.

3. Van de vierkantsvergelijking $x^2 + 2bx + c = 0$ zijn de wortels z_1 en z_2.
   Gevraagd wordt te berekenen $\max(|z_1|, |z_2|)$.
   We veronderstellen dat de variabelen b en c een bepaalde waarde hebben gekregen.
   Het volgende stuk programma geeft dan aan de variabele z abs de gevraagde absolute waarde.

```
D := b ↑ 2 - c;
if D < 0 then begin x := - b; y := (-D) ↑ 0.5 end
         else begin if b < 0 then x := - b + D ↑ 0.5
                             else x := - b - D ↑ 0.5;
                   y := 0
              end;
z abs := (x ↑ 2 + y ↑ 2) ↑ 0.5
```

De statement volgend op then mag geen conditional statement zijn. Dus op het symbool then mag nooit volgen het symbool if. Door middel van de statementhaken begin en end kan men van iedere statement een compound statement maken. Op else mag iedere willekeurige statement volgen.

Voorbeeld.
De volgende statement geeft aan max de waarde van het maximum van a, b en c.

```
if a < b then begin if b < c then max := c else max := b end
         else if a < c then max := c else max := a
```

§ 7. FOR STATEMENT

In het voorbeeld op blz. 10 moet de statement

```
som := som + 1 / n ↑ 2
```

1000 keer worden uitgevoerd, waarbij n achtereenvolgens de waarde 1,2,3,...,1000 heeft.
Een dergelijk herhaald uitvoeren van een statement kan in ALGOL ook beschreven worden met behulp van een zogenaamde for statement.
Het stukje programma

```
        n := 1;
AGAIN:  som := som + 1 / n ↑ 2;
        n := n + 1;
        if n ≤ 1000 then goto AGAIN
```

is equivalent met de for statement

```
for n := 1 step 1 until 1000 do som := som + 1 / n ↑ 2
```

Het stukje "for .... do" heet for clause. Het effect van deze for clause is dat de statement volgend op het symbool do 1000 keer wordt uitgevoerd, waarbij n achtereenvolgens de waarde 1,2,3,...,1000 heeft.

Een for statement bestaat uit een for clause gevolgd door een statement.

De statement volgend op een for clause mag iedere willekeurige statement zijn, dus ook bijvoorbeeld een for statement.

Van de verschillende vormen van een for clause behandelen we alleen de volgende vorm

```
for k := ℓ step m until n do
```

waarin k een variable van het type integer is en ℓ, m en n arithmetic expressions (dus bijvoorbeeld variables of numbers) van het type integer zijn.

De uitvoering van de for statement

```
for k := ℓ step m until n do S
```

kunnen we als volgt beschrijven

```
   k := ℓ ;
L: if (n - k) × m ≥ 0 then begin S; k := k + m; goto L end
```

Het is dus ook toegestaan dat m negatief is. In dat geval wordt de for statement beëindigd als $k < n$ geworden is.
Verder blijkt: als $(n - \ell) \times m < 0$ dan wordt de statement S geen enkele keer uitgevoerd.

Voorbeeld.
Bepaal van de reeks

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots \qquad |x| < 1$$

de som van de eerste n termen bij gegeven waarden van x en n.
Het berekenen van de som wordt in het volgende stuk ALGOL programma beschreven.

```
som := 0; term := - 1;
for k := 1 step 1 until n do
    begin term := term × (- x);
          som := som + term / k
    end
```

Officieel is het in sommige gevallen toegestaan dat op een if clause een for statement volgt. Om moeilijkheden te voorkomen laten we na een if clause alleen toe een assignment statement, een goto statement of een compound statement.

§ 8. ARRAYS

Een array dient om een rij geïndiceerde variabelen aan te duiden, bijvoorbeeld een vector of een matrix. De verzameling van geïndiceerde variabelen wordt in ALGOL aangeduid door een identifier, genaamd array identifier. Een element van deze verzameling wordt aangeduid door een subscripted variable.

Een subscripted variable bestaat uit een array identifier, gevolgd door het symbool [, gevolgd door één of meer subscripts (indices) onderling gescheiden door een komma, gevolgd door het symbool].

Als subscripts zijn toegestaan willekeurige arithmetic expressions. De waarde van een subscript is een integer. Hierbij geldt de regel: a[AE] is het element a[n], waarbij de waarde van de integer n wordt verkregen door de assignment n := AE (zie § 4).

Voorbeelden.

x[1]	x[2 × n + 1]
A[1, 2]	A[i, j + 1]
B[5, 3, -2]	B[1.2, 12 - 1, x + y + z]

Een subscripted variable is een variable van een bepaald type (real, integer of Boolean) en kan als zodanig in een arithmetic, resp. Boolean expression voorkomen. Alle subscripted variables van één array zijn van hetzelfde type. Voordat we een array identifier als zodanig mogen gebruiken moet deze identifier gedeclareerd zijn door een array declaration. In een array declaration worden opgegeven het type van de subscripted variables van het array, de array identifier en de grenzen waartussen de subscripts moeten liggen. De onder- en bovengrens voor een subscript worden gescheiden door een dubbele punt; de grenzen voor verschillende subscripts door een komma.

Voorbeelden.

real array	x[0 : 4]
integer array	A[-1 : 7, 0 : 13]
Boolean array	B[4 : 5, 3 : 3, -7 : -1]
real array	x[1 : 10], y[1 : 14], z[0 : 5]

We beperken ons voorlopig tot het geval dat de grenzen integers zijn.
De declaration is alleen correct als de ondergrens niet groter is dan de bovengrens. Incorrect is bijvoorbeeld de declaration

```
real array A[10 : 3, 4 : 7]
```

Een subscripted variable is alleen gedefinieerd als het aantal subscripts overeenstemt met het aantal grenzenparen in de array declaration an als bovendien de waarde van de subscript niet buiten de opgegeven grenzen ligt.

Voorbeeld.
Het volgende stukje programma bepaalt van een gegeven rij geïndiceerde variabelen $x_1, x_2, \ldots, x_{379}$ het maximum en de kleinste index waarvoor dit maximum wordt aangenomen.

```
max := x[1]; k := 1;
for i := 2 step 1 until 379 do
    if x[i] > max then begin k := i; max := x[k] end
```

Voorbeeld.
De oplossing van het volgende stelsel lineair onafhankelijke vergelijkingen

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ & \ddots & \\ & a_{kk} \cdots a_{kn} & \\ & \ddots & \\ & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_k \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_k \\ \vdots \\ b_n \end{pmatrix}$$

wordt gegeven door

$$x_k = \left(b_k - \sum_{i=k+1}^{n} a_{ki} x_i\right) / a_{kk} \qquad k = n, n-1, \ldots, 1$$

Het volgende stuk ALGOL programma berekent bij gegeven waarden van a_{ki} en b_k bovenstaande oplossing.

```
for k := n step - 1 until 1 do
    begin W := b[k];
          for i := k + 1 step 1 until n do W := W - a[k,i] × x[i];
          x[k] := W / a[k,k]
    end
```

Voorbeeld.

Het product van een m × n matrix A en een n × p matrix B is een m × p matrix C waarbij

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

De vermenigvuldiging kan in ALGOL als volgt beschreven worden

```
for i := 1 step 1 until m do
    for j := 1 step 1 until p do
        begin c[i,j] := 0;
              for k := 1 step 1 until n do
                  c[i,j] := c[i,j] + a[i,k] × b[k,j]
        end
```

§ 9. BLOCK STRUCTUUR

Een compound statement bestaat uit een aantal statements, onderling gescheiden door een puntkomma, voorafgegaan door het symbool begin en gevolgd door het symbool end. Wanneer behalve statements ook declarations tussen begin en end staan, dan noemen we dit geheel een block. De declarations moeten staan tussen het symbool begin en de eerste statement.

Een block bestaat uit het symbool begin, gevolgd door één of meer declarations, gevolgd door één of meer statements, gevolgd door het symbool end.

Een block is een statement, welke in het programma mag staan op iedere plaats waar een compound statement mag staan. Een block is echter essentieel iets anders dan een compound statement. De identifiers, welke in een block gedeclareerd zijn heten lokaal (t.o.v. het block). Dit betekent:

a. dat het object, aangeduid door een dergelijke identifier, buiten het block niet bestaat;
b. dat een object, dat buiten het block door dezelfde identifier wordt aangeduid, binnen het block niet bestaat.

Uit a. volgt dat de lokale identifiers van een block buiten dit block vrij ter beschikking staan om gebruikt te worden.
Uit b. volgt dat door een declaration een identifier vrij gemaakt wordt van zijn vroegere betekenis.
Dit heeft grote voordelen. Bijvoorbeeld, als twee personen samen een programma schrijven, dan hoeft de een van de ander niet te weten welke lokale identifiers hij gebruikt.

Bij het verlaten van een block verliezen de lokale variables hun betekenis en ook hun waarde. Dit impliceert dat bij het opnieuw binnenkomen van het block de lokale variables geen waarde hebben.
Identifiers, die niet in een (binnen-)block opnieuw gedeclareerd worden, behouden hun betekenis en eventuele waarde zowel bij het binnenkomen als bij het verlaten van het block. Deze identifiers noemen we niet-lokaal ten opzichte van dit (binnen-)block.

Een label is lokaal ten opzichte van het kleinste block, waarin hij voorkomt. De consequentie hiervan is dat het niet mogelijk is, door middel van een goto statement, over de declarations van een block naar een statement in het block

te springen. Een block heeft dus maar één ingang, nl. via het symbool begin. Het is wel mogelijk door middel van een goto statement een block te verlaten.

Voorbeeld.

```
begin real x, y; Boolean b;
      x := 4; y := 1.25;
  L:  b := x > y;
      if b then begin real a, b, x;
                      x := y; b := 0;
                  L:  b := b + 1;
                      a := x + y + b;
                      if x ≥ b then goto L;
                      y := b;
                      goto M
                end;
  M:  if x > y then goto L
end
```

Een for statement gedraagt zich als een block. Dit betekent dat het niet is toegestaan door middel van een goto statement over de for clause in de for statement te springen.
Het is wel toegestaan door middel van een goto statement in een conditional statement te springen. In het geval dat men in de statement tussen then en else springt is het gedeelte "else statement;" gelijkwaardig met " ; " .

In §8 hebben we ons bij de array grenzen beperkt tot integers. Als array grenzen zijn toegestaan willekeurige arithmetic expressions. De waarde van een array grens is integer (evenals bij de subscripts, zie blz. 16).
Als in het begin van een block een array declaration voorkomt dan worden de waarden van de grenzen bij de binnenkomst in het block uitgerekend.
Dit impliceert dat de eventueel gebruikte variables in de grenzen reeds een waarde moeten hebben, dus niet lokaal ten opzichte van dit block kunnen zijn.

§ 10. PROCEDURE

Een block is een min of meer afgerond stuk programma. Het kan gebeuren dat zo'n block op verschillende plaatsen in het programma voorkomt. Om niet iedere keer de tekst van dit block te moeten uitschrijven geven we het block aan door een identifier, genaamd procedure identifier. Deze procedure identifier zetten we dan op de plaatsen in het programma waar het block moet worden uitgevoerd. Deze procedure identifier is een statement (en wordt ook van andere statements gescheiden door een puntkomma). Een statement bestaande uit een procedure identifier heet een procedure statement. Een procedure statement is toegestaan op elke plaats waar een compound statement is toegestaan.

Een procedure identifier moet gedeclareerd worden door middel van een procedure declaration. De eenvoudigste vorm van een procedure declaration bestaat uit het symbool procedure, gevolgd door de procedure identifier, gevolgd door een puntkomma, gevolgd door een zogenaamde procedure body.

De procedure body is het block, dat op de plaats van de procedure statement moet worden uitgevoerd.

Voorbeeld.
Op verschillende plaatsen in het programma moeten de waarde van de twee variables a en b verwisseld worden.
De procedure declaration is

```
procedure wissel;
        begin real s;
              s := a; a := b; b := s
        end
```

Een stuk programma waarin de procedure statement voorkomt is

```
a := 1; b := 5; wissel; x := b - a
```

of als we aan a de grootste en aan b de kleinste van de waarden van a en b willen toekennen

```
if a < b then wissel
```

Naast het gebruik van een procedure identifier als procedure statement kan men een procedure identifier ook gebruiken om de waarde van een functie aan

te duiden. We spreken is dit geval van een function designator.

Voorbeeld.
In de assignment statement

```
y := sin (x) + 1
```

is sin (x) een function designator, waarvan de waarde gelijk is aan sin x bij gegeven waarde van x.

Een function designator is toegestaan op elke plaats waar een Boolean of arithmetic expression is toegestaan. Een function designator moet worden gedeclareerd door middel van een procedure declaration. Bij deze declaration wordt vóór het symbool procedure het type van de function designator aangegeven door een van de symbolen real, integer of Boolean. Het bepalen van de waarde van een function designator geschiedt door het uitvoeren van de procedure body. Daartoe moet in de procedure body een assignment statement voorkomen waarin aan de procedure identifier een waarde wordt toegekend. Dit is de waarde die de function designator krijgt.

Voorbeeld.
In een programma moet de waarde van een polynoom

$$P(x) = a_0x^{10} + a_1x^9 + \ldots + a_{10}$$

bij verschillende waarden van x worden berekend.
De hiervoor benodigde declarations zijn

```
real x, y; real array a[0 : 10];
real procedure P;
    begin integer i; real s;
        s := a[0];
        for i := 1 step 1 until 10 do s := s × x + a[i];
        P := s
    end
```

Een stukje programma, waarin P als function designator gebruikt wordt is

```
x := 0; y := P; x := 1; y := y + P
```

De variable x is het argument van het polynoom P(x). Aan ieder gebruik van P als function designator (dit noemt men procedure call) moet een assigment statement vooraf gaan die aan x de gewenste waarde geeft.
In plaats hiervan is het ook mogelijk om bij de declaration aan de procedure identifier het argument toe te voegen in de vorm van een zogenaamde formal parameter, die bij de procedure call vervangen wordt door de gewenste waarde.

Een formal parameter is een identifier, die in de procedure declaration voorkomt tussen ronde haken, direct achter de procedure identifier.

Het vorige voorbeeld wordt nu

```
real y; real array a[0 : 10];
real procedure P(x); value x; real x;
    begin integer i; real s;
          s := a[0];
          for i := 1 step 1 until 10 do s := s x x + a[i];
          P := s
    end
```

Het stukje programma wordt nu de volgende statement

```
y := P(0) + P(1)
```

Bij de procedure call staat op de plaats van de formal parameter de zogenaamde actual parameter. Een actual parameter mag een willekeurige expressie zijn. Andere vormen van actual parameters zullen we niet beschouwen.

Voorbeeld.

```
y := 0.70; y := P(y ↑ 2 + 1)
```

Een formal parameter wordt niet buiten de procedure declaration gedeclareerd. Als bij de procedure declaration aan een procedure identifier een formal parameter is toegevoegd, dan wordt vóór de procedure body een nadere omschrijving van deze formal parameter gegeven. Daarbij betekent "value x" dat de waarde van de actual parameter aan x wordt toegekend. Het gedeelte"value x" mag ook worden weggelaten, maar op de consequenties hiervan gaan we verder niet in. Het gedeelte "real x" legt het type van de waarde van de actual parameter vast.

In de procedure body moet een assignment statement voorkomen met de procedure identifier aan de linkerkant (zie blz. 21). Dit geldt ook als de procedure declaration een formal parameter bevat. In deze assignment statement wordt de parameter dus weggelaten.
Dit is het enige geval waarin de procedure identifier in het linkerlid van een assignment statement voorkomt. Het is in dit geval geen function designator. In alle andere gevallen (uitgezonderd bij een declaration) wordt de procedure identifier gebruikt voor een procedure call. Dan mag een eventuele actual parameter niet worden weggelaten.

In de procedure body van het eerste voorbeeld op blz. 22 moeten we de hulpgrootheid s invoeren. We kunnen namelijk geen gebruik maken van P in de statement

```
s := s × x + a[i]
```

omdat de procedure identifier in het rechterlid van een assignment statement een procedure call betekent.

In een procedure declaration mogen meerdere formal parameters tussen de ronde haken voorkomen. Ze worden onderling gescheiden door een komma. Bij de procedure call moet elke formal parameter vervangen worden door een actual parameter.

Voorbeeld.
In een programma moet op verschillende plaatsen het maximum van twee waarden bepaald worden. We kunnen dit doen met behulp van de procedure max.

```
real procedure max (x, y); value x, y; real x, y;
    begin real s;
        if x > y then s := x else s := y;
        max := s
    end
```

Het gebruik van de function designator als actual parameter is ook toegestaan. Zo wordt door de statement

```
M := max (max (a, b), c)
```

aan M de waarde max (a, b, c) toegekend.

Een aantal function designators behoeft niet te worden gedeclareerd. Dit zijn de zogenaamde standard functions.

abs (E)	voor de absolute waarde van de expressie E
sign (E)	voor het teken van de waarde van E (+ 1 voor E > 0, 0 voor E = 0, - 1 voor E < 0)
sqrt (E)	voor de vierkantswortel van de waarde van E
sin (E)	voor de sinus van de waarde van E
cos (E)	voor de cosinus van de waarde van E
arctan (E)	voor de arctangens van de waarde van E
ln (E)	voor de natuurlijke logarithme van de waarde van E
exp (E)	voor de waarde e^E.

In het voorbeeld van de inleiding hebben we reeds gebruik gemaakt van standard functions.

De statements

print (x) en print (x, y)

die zijn ingevoerd op blz. 10, zijn op te vatten als special procedure statements, welke niet in het programma gedeclareerd behoeven te worden. De op dezelfde bladzijde ingevoerde statement

x := read

bevat de function designator read, waarvan de waarde gelijk is aan "het volgende getal op de band". Deze function designator behoeft eveneens niet gedeclareerd te worden.

ALPHABETISCHE INDEX bladzijde