講　　座

ALGOL N について

(VI) Standard declarations (つづき)

筧　　捷　　彦*

はじめに

〖前回にひきつづき ALGOL N の報告第2版の 6. Standard declarations を，その日本語訳を中心に解説する．〗

6.2 Real Number

以下，M_b を **then, else,,, ←, :=, step,, while,**

M_c を $\equiv, \not\equiv, =, \neq, <, \leq, \geq, >$

とする．

(SD 2.0.1)

let π, e **operate before all left after right**

(SD 2.0.2)

let entier, round, sign, float, abs, $\surd$, **exp, log,** $\log_{10}$, **sin, cos,** $\tan^{-1}$

operate before all left after right

(SD 2.0.3)

let $^{-1}$, $^{-}$ **operate before** M_b **left after right**

(SD 2.0.4)

let mod operate

before M_b, M_c **left after** $^{-1}$, $^{-}$ **right**

(SD 2.0.5)

let $+, -$ **operate**

before M_b, M_c **mod left after** $^{-1}$, $^{-}$ **right**

(SD 2.0.6)

let $\times, /, \div$ **operate**

before M_b, M_c, **mod,** $+, -$ **left after** $^{-1}$, $^{-}$

right

(SD 2.0.7)

let $\uparrow$ **operate**

before M_b, M_c, **mod,** $+, -, \times, /, \div$ **left after** $^{-1}$, $^{-}$ **right**

(SD 2.1.1) $op \in \{e, \pi\}$ とする．*value* (*op*) で，

$op = e$ のとき，自然対数の底　を

$op = \pi$ のとき，円周率　をあらわす．

let *op* **represent procedure** ( ) **real**:

code ( ) **real**:

core let H←H1 [**real**];

let W←*value* (*op*);

p(H, W) if W'→next, $L \Rightarrow L$;

q(**Q**)⇒Q;

t(Q)←**real**;

h(Q)←H;

w(Q)←W;

⇒Q

end of core

(SD 2.2.1) $op \in \{+, -, \times, /, \uparrow\}$ とする．*arith* (*op*) で，*op* が $+, -, \times, /, \uparrow$ に対してそれぞれ，加算，減算，乗算，除算，巾乗をあらわす．

let ( ) *op* ( ) **represent**

procedure(**real** *a*, **real** *b*)**real**:

code(*p1*: *a*, *p2*: *b*)**real**:

core let Q←parameter;

let W_a←w(Q[*p1*:]);

let W_b←w(Q[*p2*:]);

let H←h(Q[*p1*:]);

let W←*arith* (*op*) (W_a, W_b);

p(H, W) if W'→next, $L \Rightarrow L$;

q(**Q**)⇒Q';

t(Q')←**real**;

h(Q')←H;

w(Q')←W';

⇒Q'

end of core

(SD 2.2.2)

let +( ) **represent**

procedure(**real** *a*)**real**: (**copy** *a*)

* 東京大学工学部計数工学科

(SD 2.2.3)
let −( ) represent
procedure(real *a*)real : (real [mode *a*] *0*−*a*)
(SD 2.2.4)
let ( )$^{-1}$ represent
procedure(real *a*)real : (real[mod *a*] *1*/*a*)
(SD 2.2.3)
let ( ) mod ( ) represent
procedure(real *a*, real *b*)real :
(*a* − (*a*÷*b*)×*b*)
{ +, −, ×, /, ↑, mod の結果は，二項演算であれば左の *parameter* と同じ *mode* となる. }
(SD 2.3.1) ⌊*x*⌋ で，*x* を越えない最大の整数をあらわす.
let entier ( ) represent
procedure(real *a*)integer :
code(*p1*: *a*)integer :
core let *Q*←parameter ;
let *W*←⌊w(*Q*[*p1*:])⌋;
let *H*←H 2 [real];
p(*H*, *W*) if *W′*→next, *L*⇒*L* ;
q(**Q**)⇒*Q′* ;
t(*Q′*)←real ;
h(*Q′*)←*H* ;
w(*Q′*)←*W′* ;
⇒*Q′*
end of core
(SD 2.3.2)
let sign( ) represent
procedure(real *a*)integer :
(if *a*<*0* then −*1*
else if *a*>*0* then *1* else *0*)
(SD 2.3.3)
let round( ) represent
procedure(real *a*)integer :
(sign *a* × entier(*0.5*+abs *a*))
(SD 2.3.4)
let ( )÷( ) represent
procedure(real *a*, real *b*)integer :
(entier(*a*/*b*))
{ entier, sign, round, ÷ の結果は integer-*mode* である. }
(SD 2.4.1)
let float ( ) represent
procedure(real *a*)real :
begin real *x* ; *x* : =*a* ; *x* end
(SD 2.4.2)
let abs( ) represent
procedure(real *a*)real :
begin let *x* be real ;
x : =if *a*≥*0* then *a* else −*a* ;
x end
(SD 2.4.3) *op*∈{√, exp, log, log$_{10}$, sin, cos, tan^{-1}} とする. *func* (*op*) で，*op* が √, exp, log, log$_{10}$, sin, cos, tan^{-1} のとき，それぞれ，正の平方根，指数関数，自然対数関数，常用対数関数，正弦関数，余弦関数，逆正接関数（主値 $(-\pi/2, \pi/2)$ をとる）をあらわす.
let *op*( ) represent
procedure(real *a*)real :
code(*p1*: *a*)real :
core let *Q*←parameter ;
let *W*←w(*Q*[*p1*:]);
let *H*←H 0 [real];
W←*func* (*op*) (*W*);
p(*H*, *W*) if *W′*→next, *L*⇒*L* ;
q(**Q**)⇒*Q′* ;
t(*Q′*)←real ;
h(*Q′*)←*H* ;
w(*Q′*)←*W′*
⇒*Q′*
end of core
{ float, abs, √, exp, log, log$_{10}$, sin, cos, tan^{-1} の結果は real-*mode* である. √ 等に対して，数学的に値の定義されない *parameter* が与えられたときの結果は不定とする. }
(SD 2.5.1) *op*∈{>, <} とする. *relation* (*op*) で，*op*=">" のとき，関係 > を，*op*="<" のとき，関係 < を，あらわす.
let ( ) *op* ( ) represent
procedure(real *a*, real *b*)Boolean :
code(*p1*: *a*−*b*)Boolean :
core let *Q*←parameter ;
let *w*←1 ;
if *relation* (*op*) (w(*Q*[*p1*:]), *0*)
then→*L1*, else→next ;
w←0 ;

L1: q(**Q**)⇒*Q′*;
t(*Q′*)←**bits**;
h(*Q′*)←H 2 [**bits**];
w(*Q′*)←*W*;
⇒*Q′*
end of core

(SD 2.5.2)
let ( )≥( ) **represent**
procedure(**real** *a*, **real** *b*)**Boolean**:
(¬(*a*<*b*))

(SD 2.5.3)
let ( )≤( ) **represent**
procedure(**real** *a*, **real** *b*)**Boolean**:
(¬(*a*>*b*))

{ <, ≤, ≥, > の結果は, **true** または **false** である. これらの関係は, 左の *parameter* の *mode* における *value* で調べられる.

6.3 Bits, String and Array

以下, M_b を **then, else,** , , ←, : =, **while,**
M_a を +, −, ×, /, ↑, **mod**, ÷, $^{-1}$, $^{-}$ とする.

(SD 3.0.1)
let λ, **true, false operate before all left after right**

(SD 3.0.2)
let Λ, **filler operate before all left after right**

(SD 3.0.3)
let size, lower bound, upper bound,¬ **operate before all left after right**

(SD 3.0.4)
let find operate before all left

(SD 3.0.5)
let replacing operate before M_b **left**

(SD 3.0.6)
let first operate

(SD 3.0.7)
let with, in operate after M_a **right**

(SD 3.0.8)
let conc operate before M_b, **with, in left after** M_a **right**

(SD 3.0.9)
let * **operate**
before M_b, **with, in, conc left**
after M_a **right**

(SD 3.0.10)
let set, up to, from, at operate
before M_b, **with, in, conc, set, up to, from, at left**
after M_a **right**

(SD 3.0.11)
let ∧, ∨, ⊕ **operate**
before M_b, **with, in, conc, * left**
after M_a **right**

(SD 3.0.12)
let ≡, ≠, =, ≠, <, ≤, ≥, > **operate**
before M_b, **with, in, conc,** *, ∧, ∨, ⊕ **left**
after $^{-1}$, $^{-}$ **right**

(SD 3.1.1)
let λ **represent procedure**( )**bits**:
code( )**bits**:
core q(**Q**)⇒*Q*;
t(*Q*)←**bits**;
h(*Q*)→H 1 [**bits**];
w(*Q*)←BO;
⇒*Q*
end of core

(SD 3.1.2)
let true represent
procedure( )**Boolean**: **bits** [**exact** 1] **1**;

(SD 3.1.3)
let false represent
procedure( )**Boolean**: **bits** [**exact** 1] **0**;

(SD 3.2.1)
let Λ **represent**
procedure( )**string**: **string** [ ]‘ ’

(SD 3.2.1)
let filler represent
procedure( )**string**: **string** [**exact** ‘1’] ‘ ’

(SD 3.3.1) *T*∈{**bits, string**}∪**T** [**array**] とする.
f(*T*, *W*) を,
T∈{**bits, string**} のとき, l(*W*)
T∈**T** [**array**] のとき, u(*W*)−v(*W*)+*1*
とする.
let size( ) **represent**
procedure((*T*)*a*)**integer**:
code(*p1*: *a*)**integer**:
core let *Q*←parameter;
let *W*←w(*Q*[*p1*:]);

```
        W←f(T, W);
        let H←H2[real];
        p(H, W) if W'→next, L⇒L;
        q(Q)⇒Q';
        t(Q')←real;
        h(Q')←H;
        w(Q')←W';
        ⇒Q'
    end of core
```

(SD 3.4.1) *op*∈{upper bound, lower bound}, *T*∈T[array] とする.
func(*op*) で, *op*=lower bound のとき, v を, *op*=upper bound のとき, u をあらわす.

```
let op( ) represent
    procedure( ( T ) a ) integer:
    code( p1: a )integer:
      core let Q←parameter;
           let W←w(Q[p1: ]);
           W←func (op)(W);
           let H←H2[real];
           p(H, W) if W'→next, L⇒L;
           q(Q)⇒Q';
           t(Q')←real;
           h(Q')←H;
           w(Q')←W';
           ⇒Q'
      end of core
```

(SD 3.5.1) *T*∈{bits, string} とする.

```
let ( )conc( ) represent
    procedure( (T) a, (T) b)(T):
    code( p1: a, p2: b )(T):
      core let Q←parameter;
           let Wa←w(Q[p1: ]);
           let Wb←w(Q[p2: ]);
           let n←l(Wa);
           (let W[i]←Wa[i]; )
                for i=1, 2, ···, l(Wa)
           (let W[n+i]←Wb[i]; )
                for i=1, 2, ···, l(Wb)
           let H←H1[T];
           p(H, W) if W'→next, L⇒L;
           q(Q)⇒Q';
           t(Q')←T;
           h(Q')←H;
           w(Q')←W';
           ⇒Q'
      end of core
```

(SD 3.5.2) *T*∈T[array] とする.

```
let ( )conc( ) represent
    procedure( (T) a, (T) b )(T):
    begin let va be lower bound a;
          let ua be upper bound a;
          let vb be lower bound b;
          let ub be upper bound b;
          let i be va−1;
          array [va: ua+ub−vb+1]
            (begin i:=i+1; if i≤ua then
                    (copy a[i])else
                    copy b[i−ua+ub−1]
             end)
    end
```

(SD 3.6.1) *T*∈{bits, string} とする.
null(*T*) で *T*=bits のとき λ を, *T*=string のとき Λ をあらわす.

```
let ( )*( ) represent
    procedure( real a, (T) b )(T):
    begin let w be null(T);
          let n be entier a;
          for i:= 1 step 1 until n do
            w:=w conc b;
          b
    end
```

(SD 3.7.1) *T*∈{bits, string} とする.

```
let ( )from( ) represent
  procedure( T a, integer b )T:
    begin let k be entier b;
          let s be a at k;
          let i be integer;
          for i:=k+1 step 1 until size a do
            s:=s conc a at i;
          s
    end
```

(SD 3.7.2) *T*∈T とする.

```
let ( )from( ) represent
  procedure( array T a, integer b )array T:
    begin let k be entier b;
          if k<lower bound a then
```

```
        (new a)
    else if  k>upper bound a  then  array
    [1:0]T
    else begin  let  j  be  k-1;
                array [k: upper bound a]
                (begin j:=j+1; new a[j]
                end)
  end
```

{ *a* **from** *b* は，*a* の *b* 番目以降の要素からなる部分列である．*b* が *a* の要素の数をこえる時は，結果は空となる． }

(SD 3.8.1) $T\in$ {**bits, string**} とする．

```
let  ( )up to( )  represent
  procedure( T a, integer b )T:
    begin let  k  be  entier b;
          let  s  be  a at k;
          let  i  be  integer;
          for i:=k-1 step -1 until 1 do
              s:=  a at i  conc  s;
          s
    end
```

(SD 3.8.2) $T\in$**T** とする．

```
let  ( )up to( )  represent
  procedure( array T a, integer b )array T:
    begin let  k  be entier b;
          if  k>upper bound a  then
             (new a)
      else if  k<lower bound a  then  array
      [1:0]T
      else begin  let  j  be  lower bound a;
                  array [lower bound a: k]
                  (begin j:=j+1; new a[j]
                        end)
    end
```

{ *a* **up to** *b* は，*a* の *b* 番目以前の要素からなる部分列がその結果である． }

(SD 3.9.1) $T\in$ {**bits, string**} とする．*null*(T) を T の **bits** または **string** に応じて λ または Λ とする．

```
let  ( )at( )  represent
  procedure( T a, integer b )T:
  begin  let  k  be  entier b;
         if  1≤k  ∧  k≤size a
  then code( p1: a, p2: k )T:
    core let Q←parameter;
         let W←w(Q[p1: ]);
         let i←w(Q[p2: ]);
         let W'←W[i];
         g(Q)⇒Q';
         t(Q')←T;
         h(Q')←H 0[T];
         w(Q')←W';
         ⇒Q'
    end of core
  else null (T)
  end
```

(SD 3.9.2) $T\in$**T** とする．

```
let  ( )at( )  represent
  procedure( array T a, integer b )array T:
  (if  lower bound a ≤ b  ∧  b≤upper
  bound a
  then array[b: 1](a[b])else array [1:0]T )
```

{ *a* **at** *b* は，*b* 番目の要素からだけでなる部分列である．特に *b* の値が *a* の要素の添字の範囲内にないときは，空となる． }

(SD 3.10.1) $T\in$**T** とする．

```
let  ( )set( )  represent
  procedure( array T a, integer b )array T:
  begin let  v  be  lower bound a;
        let  u  be  upper bound a;
        let  i  be  v-1;
        array [b: b+u-v]T
        (begin i:=i+1; copy a[i] end)
  end
```

{ *a* **set** *b* は，添字の範囲を $b, b+1, b+2, \cdots$ に平行移動して *a* から作られた **array** である． }

(SD 3.11.1) $T\in$ {**bits, string**} とする．

```
let  find first( )in( )  represent
  procedure( T a, T b )integer:
  begin let  i  be  integer;
        for  i:=1 step 1 until  size b  do
        if  a =  b from i up to size a
          then  go to l                ;
        i:=0;
      l: i
  end
```

{ *b* の中で *a* と同じ部分を探し出し，その最初の位置を結果とする．同じ部分が無いときは，結果は 0 と

なる. }

(SD 3.11.1) $T\in${bits, string} とする.

```
let ( )replacing first( )with( ) represent
  procedure( T a, T b, T c )T:
  begin let k be find first b in a;
        if k=0 then (copy a)
    else a up to k-1 conc c
         conc a from k+size b
  end
```

{ *a* **replacing first** *b* **with** *c* は, *a* の中の最初の *b* の部分を見い出し, その部分を *c* でおきかえたものとなる. }

```
(SD 3.12.1) let ¬( ) represent
  procedure( bits a )bits:
  begin let b be copy a;
        let i be integer;
        b:=λ;
        for i:= 1 step 1 until size a do
          b:=b conc if a at i =1 then 0
          else 1;
        b
  end
(SD 3.12.2) let ( )∧( ) represent
  procedure( bits a, bits b )bits:
  begin let c be copy a;
        let sa be size a;
        let sb be size b;
        let u be if sa≤sb then sa else sb ;
        let i be integer;
        c:=λ;
        for i:= 1 step 1 until u do
        c:=c conc
          if a at i =1 then b at i else 0;
        c:=c conc a from u+1 conc b from
        u+1;
        c
  end
(SD 3.12.3) let ( )∨( ) represent
  procedure( bits a, bits b )bits: (¬(¬a∧¬b))
(SD 3.12.4) let ( )⊕( ) represent
  procedure( bits a, bits b ) bits:
    begin let c be copy a;
          let sa be size a;
          let sb be size b;
          let u be if sa≤sb then sa else sb;
          let i be integer;
          c:=λ
          for i:= 1 step 1 until u do
          c:=c conc
            if a at i = b at i then 0 else 1;
          c:=c conc a from u+1 conc b
          from u+1;
          c
    end
```

{ $\neg a, a\wedge b, a\vee b, a\oplus b$ の結果は, *a* の *mode* と同じ *mode* である. $\neg a$ は *a* が λ のとき, λ であり, $a\wedge b, a\vee b, a\oplus b$ については, 両 *parameter* の共通部分についてのみ演算が施され, 残りについては元と同じ値をとる. }

(昭和47年7月13日受付)