Hoved kopi

101 Naur

FIRST LIST OF SUGGESTED CHANGES IN DOCUMENT 5

Naur

101 - Page 1 - section 1

OK

Insert between ... provided with labels. and Statements ~~array~~ may ...

Sequences of statements may be combined into compound statements by insertion of statement brackets.

Wijngaarden

102 - Page 1 section 1 - alinea 6.

OK

Replace by

A program is a self-contained compound statement, that is a compound statement not contained within another compound statement and which makes no use of other compound statements not contained within it.

Bauer

103 - page 1 - 6th paragraph - Objection 3

Add :

" A block which is not contained in another block and which makes no use of objectswhich are not defined within itself. "

104 Naur

Wijngaarden

104 - page 1 - section 1

Add to the end of the section (i.e. just before 1.1) something of the following kind :

Whenever it is said that precision of precise arithmetic in general is not specified or that the outcome of a certain process is undefined this is to be interpreted in this way that a program only fully defines a computational process if in accompanying information is specified what precision is assumed or what arithmetic is assumed and what course of action is assumed to be taken in all such undefined cases that may occur in the execution of the computation.

Naur

105 - page 2 - At end of section 1.1.

OK

Insert

⟨empty⟩ ::= the null string of symbols.

And change the following heading :

2. Basic symbols, unsigned integers and identifiers.

Naur

106 - page 2 - section 2.1.

OK

Add :

This alphabet may arbitrarily be stricted, or extended with any other distinctive characters (i.e. characters not coinciding with any digits, logical values or delimiters)

1.07 Naur

Naur

107 - Page 2 - section 2.2

Change to read :

2.2.1. Digits

<digit> :: = 0|1|2|3|4|5|6|7|8|9

Digits are used for forming (numbers) identifiers and strings

2.2.2. Logical values

<logical value> :: = true | false

The logical values have a fixed obvious meaning

McCarthy (introduce $\div$) sheet 1 of 3

108

Page 2 - section 2.2.3.

<arithmetic operator> :: = | + | - | x | / | $\div$ | ↑

McCarthy (deletion of $\downarrow$) sheet 1 of 3

109 - page 2 - section 2.2.3.

<arithmetic operator> :: = + | - | x | / | ↑
<bracket> :: = (|) | [|] | begin | end

Wijngaarden

finally rejected NB

110 - page 2 - section 2.3.

Add to <declaration> :: = — | — | — | dummy | own |

Wijngaarden

111 - page 2 - section 2.3.

in <declaration> change local into real

112 Naur

Wijngaarden

112 - page 2 - section 2.3.

Add the delimiter

⟨specifier⟩ :: = global

and remove global from the list in ⟨declarator⟩

Perlis

113 - page 2 - section 2.3.

Insert after ⟨declarator⟩: = ...

a new line

⟨comment⟩: = comment

Naur

114 - page 2 - section 2.3.

ok

Replace Figures and delimiters by

Digits, logical values and delimiters ...

Naur

114.1 - page 3 - section 2.5.

Replace last sentence by :

ok

Identifiers have no inherent meaning, but serve for the idenfification of simples variables, arrays, labels, switches, ~~functions and procedures.~~ They may be chosen freely (cf. however section 3.3.4. standard functions).

115 Naur

Naur

115 - page 3, section 2.5.

Add at end :

Within the statements of one block the same identifier may be used to devote only one of these entities.

McCarthy (scope of names) sheet 1 of 1

116 - page 3 - section 2.5.

The same identifier cannot be used to denote two different objects except when these identifiers objects have disjoint scopes as determined by own or local

(This replaces the correction to 2.5.)

se 170 Scope, 174

Perlis

117 - page 3 - Insert after last line of 2.5.

Two objects, within a block in which

(i) They are declared local (cf. section) and are both

or (ii) variables
or (iii) procedures (cf. section)
or (IV) functions (cf. section)
or (V) arrays (cf. section)
or (VI) labels (cf. section)
or (VII) switches (cf. section)

may not be assigned the same identifier.

Wijngaarden

118 - page 3 - section 2.5.

Add at the end of the section :

The same identifier may not be used to identify different objects within a block.

It is felt

Standard footnote

118 Naur

some members of the committee felt that there might be ambiguity associated with this concept under certain circumstances of use

Wijngaarden

118.1 - page 3 - section 2.6.

Insert after 2.6. :

2.6. Values.

~~The value associated with an identifier is that identifier.~~

The value associated with a variable or an expression is the value of that variable or expression, respectively.

The value associated with an array is the set of values of its components. (ordered)

The value associated with ~~a function or~~ a procedure is ~~that function or procedure, respectively.~~ as specified in the semantics of procedure

The value associated with a procedure identifier is the process specified by that procedure

Perlis

119 - page 3 - section 3

Insert in 3. Expressions: after " and other operators called functions." The sentence "Associated with an expression is an identifier. This identifier need not be specified explicitly. In all cases the value associated with that identifier is the value of that expression" and then continues the text.

Footnote

Naur

120 - page 3 - section 3.1.

Change heading to

3.1. Numbers.

Naur

121 - page 3 - section 3.1.1.

Add at end :

<number> :: = <unsigned number> |
+ <unsigned number> |
- <unsigned number>

122 Naur

Naur

122 - page 3 - section 3.1.2.

Read as follows :

```
  + 0          200.084          .083 10 - 02
   1 7       +  07.43 10 8          - 10 7
 -.5384          9.34 10 10       + 10 - 4
 0.073           2 10 - 4           10 + 5
```

Naur

123 - page 4 - section 3.2.3.

Delete last line :

Identifiers .... freely.

Perlis

124 - page 4 - section 3.2.4.2.

Replace section 3.2.4.2. by :

Subscript expressions must be ~~integer valued~~ of type integer, and ~~this value~~ the value of the subscr. var. is defined only if within the subscript bounds of the array (cf. section Array declaration).

The value of the subscript expression is

Wijngaarden

125 - page 4 - section 3.2.4.3.

Should read :

3.2.4.3. The type of an array is identical to that of its components.

126 Naur

Naur

126 - page 4 - section 3.3.1.

Read :

⟨function identifier⟩ :: = ⟨identifier⟩
⟨procedure identifier⟩:: =⟨identifier⟩
⟨input parameter⟩:: = ⟨array identifier⟩ |
⟨procedure identifier⟩ | ⟨function identifier⟩ |
⟨general arithmetic expression⟩
⟨input list⟩ :: = ⟨input parameter⟩ |
⟨input list⟩, ⟨input parameter⟩
⟨input part⟩ :: = ⟨empty⟩ | (⟨input list⟩)
⟨function value⟩ :: = ⟨function identifier⟩⟨input part⟩

Perlis

127 - page 4 - section 3.3.4.

Ok

Delete " Identifiers designating .... variables. However "

Perlis

127.1 - page 5 - section 3.3.4.

Insert before (in the list of functions)

" entier (E) for the largest ..... value of E "
symbolic transfer function (E), e.g.,

128 Vaur

Wijngaarden

128 p : 5 section 3.3.4. line 7

Arctan (E) for the principal value of the arctangent of the value of E.

Mc Carthy (introduce $\frac{\cdot}{\cdot}$) sheet 2 of 3

129 p : 5 3.4.

<term> : : =<factor>|<term><factor>|<term>|<factor>|<term> $\frac{\cdot}{\cdot}$ <factor>

or

3.4.4.

"The operators + - x / $\frac{\cdot}{\cdot}$ have the meanings given in section 4.2.4. with"

etc,

130 Naur

Perlis

130 p: 5

Last sentence of 3. 3. 4.
Add after "explicit declarations."
(Cf. section on declarations).

Wijngaarden

131 p : 5 Section 3. 3. 4. line 7

replace by

Arctan (E, E')	for the arctangent of the value of E/E' , i.e. y satisfying $-\pi < y \leq \pi$, sin (y) = E , cos (y) = E'

132 Mc Carthy

(Deletion of ↓) sheet 2 of 3

p : 5 section 3. 4. 2.

$(A \times \arctan(y) + Z) \uparrow (7 + Q)$

p : 5 section 3. 4. 4.

The operator ↑ denotes exponentiation where the operand preceding the ↑ is the base and the operand following the ↑ is the exponent.

Thus

$2 \uparrow (2 \uparrow n)$ means $2^{(2^n)}$
$(2 \uparrow 2) \uparrow n$ means $(2^2)^n$

133 Naur

133 p : 5

Perlis

3. 4. 3. line 1
replace "one real number" by
"value"

Perlis

134 p : 5

Section 3. 4. 4.
replace
means
by ~~(((a x b^{-1}) x 7) x (p - q)$^{-1}$) x v) x~~ s^{-1}

$$((((a \times (b^{-1})) \times 7) \times ((p-q)^{-1})) \times v) \times (s^{-1})$$

135

Bauer

Concerns Document 5 p : 5 Section 3.4.4.

Replace

"The parentheses $\downarrow\uparrow$ denote ..... exponent"

by

"The operator $\uparrow$ denotes exponentiation"

also change example to

$2 \uparrow (2 \uparrow n)$ means $2^{(2n)}$

$(2 \uparrow 2) \uparrow n$ means $(2^2)^n$

136 Naur

136

Mc Carthy (Evaluation apparenthesized expressions) sheet 1 of 1

p : 5 3.4.5.1. replaced by

3.4.5.2. The expression between a left parenthesis and the matching right parenthesis is evaluated by itself and this value is used in subsequent calculations.

Mc Carthy (deletion of) sheet 3 of 3

137 p : 6 section 3.4.5.2.
first : ↑

Mc Carthy (introduce $\frac{\cdot}{\cdot}$) sheet 3 of 3

138 p : 6 3.4.5.2.
second x / $\frac{\cdot}{\cdot}$

Perlis

139 p : 6
sentence immediately preceding
3 - 5 Boolean expressions
is replaced by
Consequently. The desired order of execution of operations within an expression can always be arranged by appropriate positioning of parentheses.

140 Naur

Rutishauser

140 p : 6 section 3. 5. 1.

replace definition of boolean term by
<boolean term> : : = false | true | (<relation>) | and so on

Vauquois

141 p : 6 section 3.5.1.

replace from syntax up to 3.5.2. ~~8 lines~~

3.5.1 Syntax

general

<relation> : : = <arithmetic expression> <relational operator> <arithmetic expression>

<Boolean term> : : = true | false | <relation> | <variable> | <function value> | (<Boolean expression>) | (<Boolean equivalence>) | (<Boolean implication>) | (<Boolean disjunction>) | (<Boolean conjunction>) | ¬<Boolean term>

<Boolean conjunction> : : = <Boolean term> ∧ <Boolean term>

<Boolean term-conjunction> : : = <Boolean term> | <Boolean conjunction>

<Boolean disjunction> : : = <Boolean term-conjunction> ∨ <Boolean term-conjunction>

<Boolean term-disjunction> : : = <Boolean term-conjunction> | <Boolean disjunction>

<Boolean implication> : : = <Boolean term-disjunction> ⊃ <Boolean term-disjunction>

<Boolean term-implication> : : = <Boolean term-disjunction> | <Boolean implication>

141 could Naur

(141 cd) <Boolean equivalence> : : =<Boolean term-implication>≡<Boolean term-implication>

<Boolean expression> : : =<Boolean term-implication>|<Boolean equivalence>

Vauquois

142 p : 6 sections 3.5.3., 3.5.4. and 3.5.5.

replace from operational meaning up to the end of the page, ~~27 lines~~

3.5.3. Operational meaning

Boolean expressions are analogous to arithmetic expressions. However the values of the constituents are confined to the truth values true and false

OR

Variables and functions except those which appear in relation must be declared Boolean.

3.5.4. The operators

Relations take on the ~~current~~ value true whenever the corresponding relation is satisfied for the expressions involved otherwise false.

The meaning of the logical operators ¬ (not), ∧ (and), ∨ (or), ⊃ (implies), ≡ (equivalent) is given by the following table.

B_1	false	false	true	true
B_2	false	true	false	true
$\neg B_1$	true	true	false	false
$B_1 \wedge B_2$	false	false	false	true
$B_1 \vee B_2$	false	true	true	true
$B_1 \supset B_2$	true	true	false	true
$B_1 \equiv B_2$	true	false	false	true

(142 cd)

142 contd Naur

3.5.5. Precedence of operators. The seq...

3.5.5.1 According to the logicians use, and to the syntax given in section 3.5.1., the following rule of precedence hold:

For operators the conventional rule of precedence applies:

first	:	$\neg$
second	:	$\wedge$
third	:	$\vee$
fourth	:	$\supset$
fifth	:	$\equiv$
sixth	:	.

$< \leq = \geq > \neq$

or

The desired order of execution of operations can always be arranged by appropriate positionning of parentheses.

3.5.5.2. The use of parentheses will be interpreted in the sense given in section 3.4.5.2.

143 Naur

Perlis

143 - page 7 - section 3.6.1.

Replace

switch :: = ...

by

switch identifier :: = ...

McCarthy (kill integer as labels) sheet 1 of 1

144 - page 7 - section 3.6.1.

label :: = identifier

Bauer

145 - page 7 - section 3.1.2.

Shift the example, used in a go to statement,

to 4.3.2,

~~Kaudexx~~ Baudr

146 - page 7 - section 3.6.5.

Replace 3.6.5. by

Unsigned integers used as labels are not treated as numbers. Thus, label 00217 is different from label 217

Bauer

147 - page 10 - section 4.3.

Insert

The effect of the go to statement is not defined ~~ex~~ if the switch is not defined.

148 Naur

McCarthy (kill localization of labels) sheet 1 of 1

148 - page 10 - section 4.3.4.

4.3.4. Restriction. All labels used in the designational expression of a go to must be defined at the place in the program where the go to occurs.

SECOND LIST OF PROPOSED CHANGES OF DOCUMENT 5

Naur

149 - page 4 - section 3.3.3.

Delete sentence

A syntactic ..... 3 lines ... arithmetic expressions.

and in the following sentence delete for functions ... 1 line ...declarations.

Naur

150 - page 7 - section 3.6.1.

Read :

< label > :: = < identifier > | < unsigned integer >
< switch identifier > :: = < identifier >
< switch value > :: = < switch identifier > [< subscript expression >]
< designational expression > :: = < label > | < switch value >

Naur

151 - page 7 - Section 3.6.2.

Last exemple reads :

Town [if y < 0 then N else N + 1]

Naur

152 - page 7 - section 3.6.3.

Last line but one, read :

may again be a switch value

153 Naur

Naur

153 - page 9 and 10 - section 4.2.

The text of the section will read as follows :

4.2. ASSIGNMENT STATEMENTS

4.2.1. Syntax

⟨left part⟩ :: = ⟨variable⟩: =

⟨left part list⟩:: =⟨left part⟩|⟨left part list⟩⟨left part⟩

⟨assignment statement⟩:: = ⟨left part list⟩⟨expression⟩

4.2.2. Examples.

S := p[0] := n := n + 1 + S

~~A := B/C - v - q x s~~

n := n + 1

A := B/C - v - q x S

s[v, k+2] := 3 - arctan (s x zeta)

V := (Q>Y) ∧ Z

4.2.3. Semantics.

Assignment statements serve for assigning the value of an expression to variables. ~~Where arithmetic expressions are concerned this process must be understood as follows :~~

Numbers and variables must be interpreted in the sense of numerical analysis, i.e. as entities defined inherently with only a finite accuracy. Similarly, the possibility of the occurrence of a finite deviation from the mathematically defined result in any arithmetic expression is explicitly understood. No exact arithmetic will be specified, however, and it is indeed understood that different hardware representation may evaluate arithmetic expressions differently. The control of the possible consequences of such differences must be carried out by the methods of numerical analysis. This control must be considered a part of the process to be described, and will therefore be expressed in terms, of the language itself.

By means of a type declaration (section) a variable or function may be declared to belong to a certain class. An expression containing variables or functions of different types must in general be assumed to have the type which mathematically speaking will embrace it in all cases.

4.2.4. Evaluation.

The meaning of the multiple assignments is that the expression is evaluated once and then assigned to all the left part variables.

4.2.5. Types.

All variables of a left part list must be of the same declared type.

154 Naur

Naur

154 - pages 10-12 - sections 4.4 - 4.7.

Delete 4.4, 4.5, 4.6, 4.7

Naur

155 - page 12 - section 4.8.1.

For <blank> read <empty>

Naur

Page 12 - section 4.8.3.

Instead of :

It only serves ....

read :

It may serve ....

Naur

156 - page 16 - section 5.1.3.

The last lines will read :

... the values true and ~~fxxs~~ false.
In arithmetic expressions integer declared variables will form a subset of real declared ones.

157 Naur

Rutishauser

157 - concerning document 16

Replace from the beginning up to and excluding numbers by

semantic meaning of expressions

Let $V_1 \ldots V_n$ be variables of several classes $C_1 \ldots C_n$;

$V_k \in C_k \quad (k = 1 \ldots n)$

Then E(V1 ... Vn) (an expression containing the variables V1...Vn has the following semantic meaning :

a) if $C = \bigcup_{k=1}^{n} (Ck)$ does not exist in ALGOL, then E has no meaning
(programmes default; example,: C1= Boolean, C2 = integer)

b) if such C as above exists, then E has the meaning " as if "

$$T* \left\{ E \left(T1 \left\{ V1 \right\} T2 \left\{ V2 \right\} \ldots Tn \left\{ Vn \right\} \right) \right\}$$

where Tk is the transfer function from Ck to C, T* the transfer function from C* to C, where C* is the smallest class containing the expression E (T1 {V1}) ..... Tn (Vn)

c) It should be understood that if a function occurs within E, the actual value of this function must be entered the list of the V's

158 Naur

Naur

158 p : 3 Section 3.1.

Replace

< signed integer >

by

< integer >

in 2 places

Green

159 p : 5 line 2

Sign (E) for the sign of the value of E (+1 for E > 0,
0 for E = 0, -1 for E < 0)

160 Naur

5 - DECLARATIONS

Declarations serve to define the properties of the identifiers of the program. A declaration for an identifier is valid for one block. Outside this block, the particular identifier may be used for other purposes.

Dynamically this implies the following : at the time of a dynamical entrance into a block (through the begin , since the labels inside are local and therefore unaccessible from outside) all identifiers declared for the block assume the significance implied by the nature of the declarations given. If these identifiers had already been defined by other declarations outside they are given a new significance. Identifiers which are not declared for the block, on the other hand, retain their old meaning.

At the time of an exit from a block (through end, or by a go to statement) all identifiers which are local to the compound lose their significance again.

The declarations for a block are divided in two lists, one defining the so-called "local" identifiers, the other defining the so-called "own" identifiers. The difference is the following : upon a second entry into the block, the values of "own" quantities will be unchanged, while the values of "local" identifiers are undefined. All identifiers of a program, with the possible exception of those for standard functions, must be declared.

The syntax of declarations is as follows :

<declaration> : : = <type declaration> | <array declaration> | <switch declaration> | <process declaration> | <declaration list> : : = <declaration> | <declaration list> ; <declaration>

161 Naur

Naur

161 p : 2 Section 2.3.

Insert after .... for facilitating reading.
For the purpose of ~~entering~~ explaining including text among the symbols of a program the following "comment" convention holds : ~~any occurrence of the delimiter ; (semicolon) may be replaced by the following :~~ The string

comment < any string not containing ; > ;

is syntactically equivalent to a semicolon.

Perlis

162 127.1 p : 5

It is understood that transfer functions between any pair of recognized entities may be defined.

3.3.5. Transfer procedures. Among the standard procedures ~~is~~ one which "transfers" an expression E of real type to one of integer type, and assigns to it the value which is the largest integer not greater than the value of E.

Mc Carthy

163

it is recommended that there be one ~~denoted~~ entier(E), namely

5.3. OWN DECLARATION

5.3.1. Syntax
<identifier list> : : = <identifier> | < identifier>, <identifier list >
<own declaration> : : = own (< identifier list >)

5.3.2. Example
own (A, B, C)

5.3.3. Semantics
An own declaration in the declaration list of a ~~compound statement or~~ block causes the identifiers listed in the declaration to have the block ~~or compound statement~~ as their scope. The value of a quantity denoted by an identifier ~~limited to a block or compound statement~~ (such) is not available to the program outside ~~its scope~~ but is again available when the ~~scope~~ is re-entered.

the block

block

164 Naur

Mc Carthy

164

Real

5.5. LOCAL DECLARATION

5.5.1. Syntax

real

<local declaration> : : = local (<identifier list>)

5.5.2. Example

local (A, B, C)

5.5.3. Semantics

The effect of a local declaration is the same as that of an own declaration except that a quantity whose identifier is mentioned in a local statement loses its value when the scope of the identifier is left.

Katz

165 p : 10 Section 4.3.4.

Add the sentence "a GO TO statement is an empty statement if the value of the designational expression is undefined."

OK

Rutishauser

166 Correction of 140

Replace <Boolean term> in 3.5.1 by

<Boolean term> : : = false | true | (<relation>) |

Replace <if clause> in 4. . 1 by

<if clause> : : = if <relation> then | if <general Boolean expression> then

167 Naur

Perlis

167 p : 7 Section 3.6.5.

ORC

..... as labels have the property that leading zeroes do not affect their meaning, e.g. 00217 denotes the same label as 217

168 Naur

Wegelein

168 Insert in introduction

(was made out from the preliminary report and the recommendations of the preparatory meetings

Prior to this meeting a completely new draft of ~~ALGOL had been prepared~~ by Peter Naur and the conference adopted this new form as a basis for its report. The Conference then proceeded to work for agreement on each item of the report. The present report represents the union of the Committee ' s concepts and the intersection of its agreements.

Whenever the precision of arithmetic is stated as being in general not specified, or the outcome of a certain process is said to be undefined, this is to be interpreted in the sense that a program only fully defines a computational process if the accompanying information specifies the precision assumed, the kind of arithmetic assumed, and the course of action to be taken in all such undefined cases ~~that~~ may occur during the execution of the computation.

Naur

169 p : 5 Section 3.4.5.

3.4.5.1. and 3.4.5.2. to be replaced by :

3.4.5.1. According to the syntax given in section 3.4.1. the following rules of precedence hold :
first : ↑
second : × / ÷
third : + −

3.4.5.2. The expression between a left parenthesis and the matching right parenthesis is evaluated by itself and this value is used in subsequent calculations. Consequently the desired order of execution of operations within an expression can always be arranged by appropriate positioning of parentheses.

Perlis

cf. 174, 116, 118

170 2.8. Scope

The scope of a block is the set of statements comprising the block. The scope of a property of a quantity is the block in which that quantity is declared to have that property.

171 Naur

171 Section 5.3.
Switch declaration

Syntax

<Switch list> : : = <general designational expression>|(switch list >,
<general designational expression>
<switch declaration>: : = Switch <switch identifier>: =
(< switch list >)

Examples

Switch S : = (S1, S2, Q[m] , if v > 0 then S3 else S4)

Switch Q : = (p1, v)

Semantics.
The purpose of a switch declaration is to provide various successors for the associated GO TO statement. The designational expression of the switch list that is selected is determined by the actual numerical value of the subscript expression of the switch variable. Only integral values of the subscript expression that correspond to positions within the switch list are defined. If the value of the subscript expression is undefined, the refering GO TO statement is an empty statement.

Rutishauser

172 to document 25 part 1

Replace definition of <for prefix> , < for clause > by
<for clause> : : = for <variable> : = <expression list> do
For stacking of for clauses, see definition of syntax of statements.

173 Ad Doc 25 3 Semantics

To replace Recursive .....7 lines ..... expression list
A for clause controls the execution and the choice of the successor of the statement it precedes. This control permits, through convenient notation, certain variables to be assigned values of expressions, suitably selected from a set of expressions specified in the for clause.

Names. 104

The name of an identifier is that identifier.

⊗ It is felt that there might be ambiguity associated with this concept under certain circumstances of use.

174 Naur

174 - page 4 - Insert after 2.6.

2.7. Names

The name of an identifier is that identifier.

The name of a variable or expression is ~~the (name of the) identifier associated with that variable or expression, respectively.~~

~~The name of an array, function or procedure is that function identifier, procedure identifier or procedure~~ the identifier associated with that array, ~~function~~ or procedure, respectively.

footnote

cf. 170 Scope, 118 Value

2.8. Quantity

We distinguish the following kinds of quantities

~~A quantity~~ a variable, an expression
label switch
procedure
array

~~When kind is used it refers to quantity.~~

When type is used it refers to some of the properties of quantities in particular those properties which can be declared in ~~Algol 60~~ the language

175 Naur

FOR STATEMENT

1 - SYNTACTIC

<EXPRESSION LIST ELEMENT> :: = <GENERAL EXPRESSION>
| <GENERAL EXPRESSION> STEP
<GENERAL EXPRESSION> UNTIL
<GENERAL EXPRESSION> |
<GENERAL EXPRESSION> WHILE
<GENERAL BOOLEAN EXPRESSION>

<EXPRESSION LIST> :: = <EXPRESSION LIST ELEMENT> |
<EXPRESSION LIST>, <EXPRESSION LIST ELEMENT>

<FOR CLAUSE> :: = FOR <VARIABLE> : = <EXPRESSION LIST> DO *)

<FOR STATEMENT> :: = <FOR CLAUSE><COMPOUND STATEMENT> |
<BLOCK><STATEMENT><CONDITIONAL STATEMENT>>

(For clause><statement>!

2 - Examples

<label>: <for clause><statement>

FOR Q : = 1 STEP A S UNTIL N DO A[Q] : = B[Q];

FOR k : = 1, V1 x 2 WHILE V1<N do FOR J : = A + B, L, 1 STEP 1 UNTIL N, C + D DO A[k, J] : = B[k, J] !

4.6.3. Semantics.

A for clause causes the statement S which precedes it to be repeatedly executed ~~n~~ zero or more times while both the following conditions remain true :

1) The for-clause is not exhausted, i.e., further values remain to be assigned to its controlled variable.
2) No exit out of S has occurred.

in section 4.6.4

In addition to causing S to be repeated, the for-clause assigns a sequence of values to its controlled variable as described below ;

NB Rephrase

4.6.6. The effect of a go to statement outside a for statement which refers to a statement within the for statement is undefined.

4.6.4. The for list elements.

The ~~expression~~ list gives the rules for obtaining a sequence of values. The elements of the list may be :

1 - arithmetic expressions (E)

*) Nothing to do with with DO

175.1 Naur

FOR STATEMENT

175.1

2 - E_1 STEP E_2 UNTIL E_3,

to mean the sequence of expressions V_0, V_1, ... defined by

V_0 is E_1

and V_n is $V_{n-1} + E_2$ for $n = 1, 2, \ldots .$;

The sequence continuing for as long as the condition

$$(V_n - E_3) \times \text{sign}(E_2) \leq 0$$

is satisfied immediately following the execution of the controlled statement for the expression V_{n-1}, and the sequence being absent if the condition is not satisfied for V_0 at the beginning of execution.

3 - E WHILE B , to mean the sequence of values defined by E as long as the value of B is TRUE

4.6.5. The value of the controlled variable upon exit.
Upon exit out of the statement S (supposed to be compound) through a go to statement the value of the controlled variable will remain the same as it was during the particular execution when the go to statement was encountered.
If the exit is due to exhaustion of the for list, on the other hand, the value of the controlled variable is undefined after the exit.

4.6.6. Go to leading into a for statement.

173 Naur

Wijngaarden

173 - page 3 - section 2.5.

Insert :

2.5.1. Letter string

2.5.-1.1. Syntax

⟨letter string⟩:: = ⟨letter⟩|⟨letter string⟩⟨letter⟩

Wijngaarden

174 - page 16 - section 5.1.3.

5.1.3. Semantics.

Type declarations serve to declare certain quantities to be of a certain type, e.g., integer, Boolean. In every case, the type declared must be compatible with the type defined in 2.7.

Wijngaarden

175 - page 3 - section 2.7.

For the editor

2.7. Types.

A quantity may be declared to be of a specific type, e.g.,real, integer, Boolean (cf. Type declarations 5.1.3). In general, the type of a quantity is the type of the value, if defined, of that quantity.

A type A may be defined to be compatible with the type B. If then the type B is compatible with the type C, then also the Type A is compatible with C. The type " integer " is compatible with the type " real ".

A type D may be defined to be incompatible with the type E. Then also the type E is incompatible with the type D. If then the type F is compatible with the type D it is also incompatible with the type E.

The type " real " is incompatible with the type " Boolean ".

The type of the elements of an array is the type of the array.

8-1 Naur

Rutishauser to document 16 , case 2.5

Since I believe it is a grave error , I propose to accept possibility 2a of 2.5 in place of possibility 1.

Accepted