https://zh.wikipedia.org/wiki/%E5%B7%B4%E7%A7%91%E6%96%AF%E8%8C%83%E5%BC%8F

巴克斯范式是计算机语言设计的基础

SYNTAX SUMMARY

This appendix defines the syntax of ALGOL entities in a notation known as Backus Normal Form (BNF). In this notation, entities are defined in terms of other entities in the following format:

<a> ::= <b> <c>

This notation means that the entity <a> consists of the entity <b> followed by entity <c>. A entity bracketed by < and > is defined further elsewhere in the syntax. Anything not appearing between < and > stands for itself.

<a>表示一个实体，该语法在其他地方也适用。
其他没有包含<>之中的，都表示它自身。

For example

<a> ::= <d> #BEGIN#

If one entity can be defined by more than one sequence, the sequences are separated by vertical lines:

<a> ::= <b> <c> | <d> #BEGIN#

In this example the entity <a> consists of either <b> followed by entity <c>, or <d> followed by the characters #BEGIN#.

The entity being defined can also appear in the definition, in which case the definition is recursive. For example:

递归定义

<a> ::= <b> <c> | <d> #BEGIN# | <a> <e>

This definition means that <a> consists of either <b> followed by entity <c>, or <d> followed by the characters #BEGIN#, followed in either case by any number (including none) of occurrences of <e>.

The representation of syntax in this appendix does not correspond precisely to the representation of syntax in the main text of the manual, but the ultimate syntax of an ALGOL program is the same in both cases.

<empty> ::=

<basic symbol>::= <letter> | <digit> | <logical value> | <special symbol> | <delimiter>

<letter> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | Y | W | X | Y | Z

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

<logical value> ::= #TRUE# | #FALSE#

<special symbol> ::= <any symbol in CDC 64-charaeter set>

<delimiter> ::= <operator> | <separator> | <bracket> | <declarator> | <specificator >

<operator> ::= <arithmetic operator> | <relational operator> | <logical operator > | <sequential operator>

<arithmetic operator> ::= + | - | * | / | // | ** | ^

a**b: a的b次幂
a//b: 整数除法
a^b: a的b次幂

<relational operator> ::= < | #le# | = | #ge# | > | ~=

<logical operator> ::= #EQUIV# | #IMPL# | #and# | #OR# | ~

<sequential operator> ::= #GO TO# | #IF# | #THEN# | #ELSE# | #FOR# | #DO#

<separator> ::= # | , | . | : | ; | := | #STEP# | #UNTIL# | #WHILE# | #COMMENT# | #CODE# | #ALGOL# | #FORTRAN# | #RJ#

<bracket> ::=) | (|] | [| #(# | #)# | #BEGIN# | #END#

<declarator> ::= #OWN# | #BOOLEAN# | #INTEGER# | #REAL# | #ARRAY# | #SWITCH# | #PROCEDURE#

<specificator> ::= #STRING# | #LABEL# | #VALUE# | #VARIABLE# | #SIMPLE# | #FORMAT# | #LIST#

<identifier> ::= <letter> | <identifier><letter> | < identifier><digit>

<ld> ::= <letter> | <digit>

< tail> ::= <ld> | <ld><ld> | <ld><ld><ld> | <ld><ld><ld><ld> | <ld><ld><ld><ld><ld> | <ld><ld><ld><ld> <ld><ld>

<external identifier> ::= <letter> | <letter><tail>

<unsigned integer> ::= <digit> | <unsigned integer><digit>

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

<decimal fraction> ::= .<unsigned integer>

<exponent part> ::= #<integer>

<decimal number> ::= <unsigned integer> | <decimal fraction> | <unsigned integer><decimal fraction>

<unsigned number> ::= <decimal number> | <exponent part> | <decimal number><exponent part>

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

<proper string> ::= <any sequence of characters not containing #(# or #)#> | <empty>

<open string> ::= <proper string> | <proper string>#(#< open string>#)#<open string>

<string> ::= #(#<open string>#)# | #(#<open string>#)#<string>

CONSTITUENTS OF EXPRESSIONS

<variable identifier> ::= <identifier>

<simple variable> ::= <variable identifier>

<subscript expression> ::= <arithmetic expression>

<subscript list> ::= <subscript expression> | <subscript list>, <subscript expression>

<array identifier> ::= <identifier>

<subscripted variable> ::= <array identifier> [<subscript list>]

<variable> ::= <simple variable> | <subscripted variable>

< procedure identifier> ::= < identifier>

<actual parameter> ::= <string> | <expression> | <array identifier> | <switch identifier> | <procedure identifier>

<letter string> ::= <letter> | <letter string><letter>

<parameter delimiter> ::= , |)<letter string>:(

<actual parameter list> ::= <actual parameter> | <actual parameter list> <parameter delimiter> <actual parameter>

<actual parameter part> ::= <empty> | <actual parameter list>

<function designator> ::= <procedure identifier> <actual parameter part>

EXPRESSIONS

<expression> ::= <arithmetic expression> | <Boolean expression> | <designational expression>

<adding operator> ::= + | -

< multiplying operator> ::= * | / | //

<primary> ::= <unsigned number> | <variable> | <function designator> | (<arithmetic expression>)

<factor> ::= <primary> | <factor> ** <primary> | <factor> ^ <primary>

<term> ::= <factor> | <term> <multiplying operator> <factor>

<simple arithmetic> ::= <term> | <adding operator > <term> | <simple arithmetic> <adding operator> <term>

<if clause> ::= #IF# <Boolean expression> #THEN#

<arithmetic expression> ::= <simple arithmetic> | <if clause> <simple arithmetic> #ELSE# <arithmetic expression>

<relational operator > ::= < | #le# | = | #ge# | > | ~=

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

<Boolean primary>::= <logical value> | <variable> | <function designator> | <relation> | (<Boolean expression>)

<Boolean secondary> ::= <Boolean primary> | ~<Boolean primary>

<Boolean factor> ::= <Boolean secondary> | <Boolean factor> #or# <Boolean secondary>

<Boolean term>::= <Boolean factor> | <Boolean term> #and# <Boolean factor>

<implication> ::= <Boolean term> | <implication> #impl# <Boolean term>

<simple Boolean> ::= <implication> | <simple Boolean> #equiv# <implication>

<Boolean expression> ::= <simple Boolean> | <if clause> <simple Boolean> #ELSE# < Boolean expression>

<label> ::= <identifier>

<switch identifier> ::= <identifier>

<switch designator> ::= <switch identifier>[<subscript expression>]

<subscript expression> ::= <arithmetic expression>

<simple designational> ::= <label> | <switeh designator> | (<designational expression>)

<designational expression> ::= <simple designational> | <if clause> <simple designational> #ELSE# <designational expression>

COMPOUND STATEMENTS AND BLOCKS

<compound tail> ::= <statement> #END# | <statement>; <compound tail>

<block head> ::= #BEGIN# <declaration> | <block head >; < declaration >

<unlabeled compound> ::= #BEGIN# <compound tail>

<unlabeled block> ::= <block head>; <compound tail>

<compound statement> ::= <unlabeled compound> | <label>: <compound statement>

<block> ::= <unlabeled block> | <label>: <block>

<program> ::= <block> | <compound statement>

STATEMENTS AND BASIC STATEMENTS

<unlabelled basic statement> ::= <assignment statement> | <go to statement> | <dummy statement> | <procedure statement>

<basic statement> ::= <unlabelled basic statement> | <label>:<basic statement>

<destination>::= <variable> | <procedure identifier>

<left part> ::= <destination> :=

<left part list> ::= <left part> | <left part list><left part>

<assignment statement> ::= <left part list> <arithmetic expression> | <left part list> <Boolean expression>

<go to statement> ::= #GO TO# <designational expression>

<dummy statement> ::= <empty>

<procedure identifier> ::= <identifier>

<actual parameter> ::= <string> | <expression> | <array identifier > | <switch identifier> | <procedure identifier>

<letter string> ::= <letter> | <letter string><letter>

<parameter delimiter> ::= , |)<letter string>:(

<actual parameter list> ::= <actual parameter> | <actual parameter list > <parameter delimiter> <actual parameter>

<actual parameter part> ::= <empty> | (<actual parameter list>)

<procedure statement> ::= <procedure identifier> <actual parameter part>

<statement> ::= <unconditional statement> | <conditional statement> | <for statement>

<for list element> ::= <arithmetic expression> | <arithmetic expression> #STEP# <arithmetic expression> #UNTIL# <arithmetic expression> | <arithmetic expression> #WHILE#< Boolean expression>

<for list> ::= <for list element> | <for list>, <for list element>

<for clause> ::= #FOR# <variable identifier> := <for list> #DO#

<for statement> ::= <for clause><statement> | <label>:<for statement>

<if clause> ::= #IF# <Boolean expression> #THEN#

<unconditional statement> ::= <basic statement> | <compound statement> | <block>

<if statement> ::= <if clause> <unconditional statement>

<conditional statement> ::= <if statement> | <if statement> #ELSE# <statement> | <if clause> <for statement> | <label>: <conditional statement>

DECLARATIONS

<declaration> ::= <type declaration> | <array declaration> | <switch declaration> | <procedure declaration>

<type list> ::= <simple variable> | <simple variable>, <type list>

<type> ::= #REAL# | #INTEGER# | #BOOLEAN#

<local or own> ::= <empty> | #OWN#

<type declaration> ::= <local or own> <type> <type list>

<lower bound> ::= <arithmetic expression>

<upper bound> ::= <arithmetic expression>

<bound pair>::= <lower bound>: <upper bound>

<bound pair list> ::= <bound pair> | <bound pair list>, <bound pair>

<array segment> ::= <array identifier>[<bound pair list>] | <array identifier>, <array segment>

<array list> ::= <array segment> | <array list>, <array segment>

<array declarer> ::= < type> #ARRAY# | #ARRAY#

<array declaration> ::= <local or own><array declarer> <array list>

<switch list> ::= <designational expression> | <switch list> <designational expression>

<switch declaration> ::= #SWITCH# <switch identifier> := <switch list>

<formal parameter> ::= <identifier>

<formal parameter list> ::= <formal parameter> | <formal parameter list> <parameter delimiter> <formal parameter>

<formal parameter part> ::= <empty> | (<formal parameter list>)

<identifier list> ::= <identifier> | <identifier list>, <identifier>

<value part> ::= #VALUE# <identifier list>; | <empty>

<separate specifier> ::= #STRING# | <type> | <array declarer> | #LABEL# | #SWITCH# | #PROCEDURE# | <type>#PROCEDURE# | #VARIABLE# |#SIMPLE# | #FORMAT# | #LIST#

<separate specification part> ::= <empty> | <separate specifier><identifier list>; | <separate specification part> <separate specifier> <identifier list>;

<separate procedure heading> ::= <procedure identifier> <formal parameter part>; <value part> <separate specification part>

<code number> ::= <digit> | <digit><digit> | <digit><digit> <digit> | <digit><digit><digit><digit> | <digit> <digit><digit><digit><digit>

<code identifier> ::= <empty> | <code number> | <external identifier>

<code specifier> ::= #RJ# | <empty>

<code> ::= #CODE# <code specifier> <code identifier> | #ALGOL#<code specifier><code identifier> | #FORTRAN#<code identifier>

<separate procedure body> ::= <code>

<separate procedure declaration> ::= #PROCEDURE# <separate procedure heading>

<separate procedure body> | <type>#PROCEDURE# <separate procedure heading> <separate procedure body>

<specifier> ::= #STRING# | <type> | <array declarer> | #LABEL# | #SWITCH# | #PROCEDURE# | <type> #PROCEDURE#

<specification part> ::= <empty> | <specifier> <identifier list>; | <specification part> <specifier> <identifier list>;

<procedure heading> ::= <procedure identifier> <formal parameter part>; <value part> <specification part>

<procedure body> ::= <statement>

<procedure declaration>::= #PROCEDURE# <procedure heading> <procedure body> | <type> #PROCEDURE# <procedure heading> <procedure body>

I scanned, OCRed, and proofread the Backus-Naur form of Algol-60 from an old CDC Manual I had still lying around. Please do not blame me for any errors. Use at your own risk. I would like to add that the representation here is partially due to the awkward 6 bit character code CDC used. In general, what is written here as hash sign (#) is normally represented by a double quote ("). I replaced some characters in the printed version, where I have no idea what ASCII I should use by the appropriate keywords (e.g., right arrow by #IMPL# or the greater equal sign by #GE#).

Bernhard Treutwein - BdT(at)wildwein(dot)de