

```

begin array z, y1, y2, y3 [1 : n] ; real x1, x2, x3, H ; Boolean out ;
  integer k, j ; own real s, Hs ;
  procedure RK1ST (x, y, h, xe, ye) ; real x, h, xe ; array y, ye ;
    comment RK1ST integrates one single Runge-Kutta step with initial values
    x, y [k] which yields the output parameters xe = x + h and ye [k], the latter
    being the solution at xe.
    Important : the parameters n, FKT, z enter RK1ST as non local entities ;

begin array w [1 : n], a [1 : 5] ; integer k, j ;
  a [1] := a [2] := a [5] := h/2 ; a [3] := a [4] := h ; xe := x ;
  for k := 1 step 1 until n do ye [k] := w [k] := y [k] ;
  for j := 1 step 1 until 4 do

    begin FKT (xe, w, n, z) ;
      xe := x + a [j] ;
      for k := 1 step 1 until n do

        begin w [k] := y [k] + a [j] × z [k] ;
          ye [k] := ye [k] + a [j + 1] × z [k]/3
        end k
      end j
    end RK 1ST ;

Begin of program :

  if fi then begin H := xE - x ; s := 0 end else H := Hs ;
  out := false ;

AA : if (x + 2.01 × H - xE > 0) ≡ (H > 0) then
  begin Hs := H ; out := true ; H := (xE - x)/2 end if ;
  RK1ST (x, y, 2 × H, x1, y1) ;

BB : RK1ST (x, y, H, x2, y2) ; RK1ST (x2, y2, H, x3, y3) ;
  for k := 1 step 1 until n do
    if comp (y1 [k], y3 [k], eta) > eps then go to CC ;

    comment comp (a, b, c) is a function designator, the value of which is the absolute value
    of the difference of the mantissae of a and b, after the exponents of these quantities have
    been made equal to the largest of the exponents of the originally given parameters a, b, c ;

    x := x3 ; if out then go to DD ;
    for k := 1 step 1 until n do y [k] := y3 [k] ;
    if s = 5 then begin s := 0 ; H := 2 × H end if ;
    s := s + 1 ; go to AA ;

CC : H := 0.5 × H ; out := false ; x1 := x2 ;
  for k := 1 step 1 until n do y1 [k] := y2 [k] ;
  go to BB ;

DD : for k := 1 step 1 until n do yE [k] := y3 [k]
end RK

```

ALPHABETIC INDEX OF DEFINITIONS OF CONCEPTS AND SYNTACTIC UNITS

All references are given through section or clause numbers. The references are given in three groups:

- def Following the abbreviation "def", reference to the syntactic definition (if any) is given.
- synt Following the abbreviation "synt", references to the occurrences in metalinguistic formulae are given. References already quoted in the def-group are not repeated.
- text Following the word "text", the references to definitions given in the text are given.

The basic symbols represented by signs other than bold-faced words have been collected at the beginning. The examples have been ignored in compiling the index.

- + see : plus
- − see : minus
- × see : multiply
- / ÷ see : divide
- ↑ see : exponentiation
- < ≤ = ≥ > ≠ see: <relational operator>
- ≡ ⊃ ∨ ∧ ¬ see: <logical operator>
- , see : comma
- . see : decimal point
- ₁₀ see : ten
- :
- ; see : semicolon
- := see : colon equal
- see : space
- () see : parentheses
- [] see : subscript bracket
- ' ' see : string quote

- <actual parameter>, def 3.2.1, 4.7.1
- <actual parameter list>, def 3.2.1, 4.7.1
- <actual parameter part>, def 3.2.1, 4.7.1
- <adding operator>, def 3.3.1
- alphabet, text 2.1
- arithmetic, text 3.3.6
- <arithmetic expression>, def 3.3.1 synt 3, 3.1.1, 3.3.1, 3.4.1, 4.2.1, 4.6.1, 5.2.1 text 3.3.3
- <arithmetic operator>, def 2.3 text 3.3.4
- array**, synt 2.3, 5.2.1, 5.4.1
- array, text 3.1.4.1
- <array declaration>, def 5.2.1 synt 5 text 5.2.3
- <array identifier>, def 3.1.1 synt 3.2.1, 4.7.1, 5.2.1 text 2.8
- <array list>, def 5.2.1
- <array segment>, def 5.2.1
- <assignment statement>, def 4.2.1 synt 4.1.1 text 1, 4.2.3
- <basic statement>, def 4.1.1 synt 4.5.1
- <basic symbol>, def 2
- begin**, synt 2.3, 4.1.1
- <block>, def 4.1.1 synt 4.5.1 text 1, 4.1.3, 5

<block head>, def 4.1.1
Boolean, synt 2.3, 5.1.1 text 5.1.3
 <Boolean expression>, def 3.4.1 synt 3, 3.3.1, 4.2.1, 4.5.1, 4.6.1 text 3.4.3
 <Boolean factor>, def 3.4.1
 <Boolean primary>, def 3.4.1
 <Boolean secondary>, def 3.4.1
 <Boolean term>, def 3.4.1
 <bound pair>, def 5.2.1
 <bound pair list>, def 5.2.1
 <bracket>, def 2.3

 <code>, synt 5.4.1 text 4.7.7, 5.4.6
 colon :, synt 2.3, 3.2.1, 4.1.1, 4.5.1, 4.6.1, 4.7.1, 5.2.1
 colon equal :=, synt 2.3, 4.2.1, 4.6.1, 5.3.1
 comma ,, synt 2.3, 3.1.1, 3.2.1, 4.6.1, 4.7.1, 5.1.1, 5.2.1, 5.3.1, 5.4.1
comment, synt 2.3
 comment convention, text 2.3
 <compound statement>, def 4.1.1 synt 4.5.1 text 1
 <compound tail>, def 4.1.1
 <conditional statement>, def 4.5.1 synt 4.1.1 text 4.5.3

 <decimal fraction>, def 2.5.1
 <decimal number>, def 2.5.1 text 2.5.3
 decimal point ., synt 2.3, 2.5.1
 <declaration>, def 5 synt 4.1.1 text 1, 5 (complete section)
 <declarator>, def 2.3
 <delimiter>, def 2.3 synt 2
 <designational expression>, def 3.5.1 synt 3, 4.3.1, 5.3.1 text 3.5.3
 <digit>, def 2.2.1 synt 2, 2.4.1, 2.5.1
 dimension, text 5.2.3.2
 divide / ÷, synt 2.3, 3.3.1 text 3.3.4.2
do, synt 2.3, 4.6.1
 <dummy statement>, def 4.4.1 synt 4.1.1 text 4.4.3

else, synt 2.3, 3.3.1, 3.4.1, 3.5.1, 4.5.1 text 4.5.3.2
 <empty>, def 1.1 synt 2.6.1, 3.2.1, 4.4.1, 4.7.1, 5.4.1
end, synt 2.3, 4.1.1
 entier, text 3.2.5
 exponentiation $\uparrow$, synt 2.3, 3.3.1 text 3.3.4.3
 <exponent part>, def 2.5.1 text 2.5.3
 <expression>, def 3 synt 3.2.1, 4.7.1 text 3 (complete section)

 <factor>, def 3.3.1
false, synt 2.2.2
for, synt 2.3, 4.6.1
 <for clause>, def 4.6.1 text 4.6.3
 <for list>, def 4.6.1 text 4.6.4
 <for list element>, def 4.6.1 text 4.6.4.1, 4.6.4.2, 4.6.4.3
 <formal parameter>, def 5.4.1 text 5.4.3
 <formal parameter list>, def 5.4.1
 <formal parameter part>, def 5.4.1
 <for statement>, def 4.6.1 synt 4.1.1, 4.5.1 text 4.6 (complete clause)
 <function designator>, def 3.2.1 synt 3.3.1, 3.4.1 text 3.2.3, 5.4.4

go to, synt 2.3, 4.3.1
 <go to statement>, def 4.3.1 synt 4.1.1 text 4.3.3

<identifier>, def 2.4.1 synt 3.1.1, 3.2.1, 3.5.1, 5.4.1 text 2.4.3
 <identifier list>, def 5.4.1
if, synt 2.3, 3.3.1, 4.5.1
 <if clause>, def 3.3.1, 4.5.1 synt 3.4.1, 3.5.1 text 3.3.3, 4.5.3.2
 <if statement>, def 4.5.1 text 4.5.3.1
 <implication>, def 3.4.1
integer, synt 2.3, 5.1.1 text 5.1.3
 <integer>, def 2.5.1 text 2.5.4

label, synt 2.3, 5.4.1
 <label>, def 3.5.1 synt 4.1.1, 4.5.1, 4.6.1 text 1, 4.1.3
 <left part>, def 4.2.1
 <left part list>, def 4.2.1
 <letter>, def 2.1 synt 2, 2.4.1, 3.2.1, 4.7.1
 <letter string>, def 3.2.1, 4.7.1
 local, text 4.1.3
 <local or own type>, def 5.1.1 synt 5.2.1
 <logical operator>, def 2.3 synt 3.4.1 text 3.4.5
 <logical value>, def 2.2.2 synt 2, 3.4.1
 <lower bound>, def 5.2.1 text 5.2.4

 minus —, synt 2.3, 2.5.1, 3.3.1 text 3.3.4.1
 multiply ×, synt 2.3, 3.3.1 text 3.3.4.1
 <multiplying operator>, def 3.3.1

 non-local, text 4.1.3
 <number>, def 2.5.1 text 2.5.3, 2.5.4

 <open string>, def 2.6.1
 <operator>, def 2.3
own, synt 2.3, 5.1.1 text 5, 5.2.5

 <parameter delimiter>, def 3.2.1, 4.7.1 synt 5.4.1 text 4.7.6
 parentheses (), synt 2.3, 3.2.1, 3.3.1, 3.4.1, 3.5.1, 4.7.1, 5.4.1, text 3.3.5.2
 plus + , synt 2.3, 2.5.1, 3.3.1 text 3.3.4.1
 <primary>, def 3.3.1
procedure, synt 2.3, 5.4.1
 <procedure body>, def 5.4.1
 <procedure declaration>, def 5.4.1 synt 5 text 5.4.3
 <procedure heading>, def 5.4.1 text 5.4.3
 <procedure identifier>, def 3.2.1 synt 3.2.1, 4.7.1, 5.4.1 text 4.7.5.4
 <procedure statement>, def 4.7.1 synt 4.1.1 text 4.7.3
 <program>, def 4.1.1 text 1
 <proper string>, def 2.6.1

 quantity, text 2.7

real, synt 2.3, 5.1.1 text 5.1.3
 <relation>, def 3.4.1 text 3.4.5
 <relational operator>, def 2.3, 3.4.1

 scope, text 2.7
 semicolon ; , synt 2.3, 4.1.1, 5.4.1
 <separator>, def 2.3
 <sequential operator>, def 2.3

<simple arithmetic expression>, def 3.3.1 text 3.3.3
 <simple Boolean>, def 3.4.1
 <simple designational expression>, def 3.5.1
 <simple variable>, def 3.1.1 synt 5.5.1 text 2.4.3
 space , synt 2.3 text 2.3, 2.6.3
 <specification part>, def 5.4.1 text 5.4.5
 <specifier>, def 2.3
 <specifier>, def 5.4.1
 standard function, text 3.2.4, 3.2.5
 <statement>, def 4.1.1, synt 4.5.1, 4.6.1, 5.4.1 text 4 (complete section)
 statement bracket see: **begin end**
step, synt 2.3, 4.6.1 text 4.6.4.2
string, synt 2.3, 5.4.1
 <string>, def 2.6.1 synt 3.2.1, 4.7.1 text 2.6.3
 string quotes ' ', synt 2.3, 2.6.1 text 2.6.3
 subscript, text 3.1.4.1
 subscript bound, text 5.2.3.1
 subscript brackets [], synt 2.3, 3.1.1, 3.5.1, 5.2.1
 <subscripted variable>, def 3.1.1 text 3.1.4.1
 <subscript expression>, def 3.1.1 synt 3.5.1
 <subscript list>, def 3.1.1
 successor, text 4
switch, synt 2.3, 5.3.1, 5.4.1
 <switch declaration>, def 5.3.1 synt 5 text 5.3.3
 <switch designator>, def 3.5.1 text 3.5.3
 <switch identifier>, def 3.5.1 synt 3.2.1, 4.7.1, 5.3.1
 <switch list>, def 5.3.1

 <term>, def 3.3.1
 ten ₁₀ , synt 2.3, 2.5.1
then, synt 2.3, 3.3.1, 4.5.1
 transfer function, text 3.2.5
true, synt 2.2.2
 <type>, def 5.1.1 synt 5.4.1 text 2.8
 <type declaration>, def 5.1.1 synt 5 text 5.1.3
 <type list> def 5.1.1

 <unconditional statement>, def 4.1.1, 4.5.1
 <unlabelled basic statement>, def 4.1.1
 <unlabelled block>, def 4.1.1
 <unlabelled compound>, def 4.1.1
 <unsigned integer>, def 2.5.1, 3.5.1
 <unsigned number>, def 2.5.1 synt 3.3.1
until, synt 2.3, 4.6.1 text 4.6.4.2
 <upper bound>, def 5.2.1 text 5.2.4

value, synt 2.3, 5.4.1
 value, text 2.8, 3.3.3
 <value part>, def 5.4.1 text 4.7.3.1
 <variable>, def 3.1.1 synt 3.3.1, 3.4.1, 4.2.1, 4.6.1 text 3.1.3
 <variable identifier>, def 3.1.1

while, synt 2.3, 4.6.1 text 4.6.4.3

PART I B

REPORT ON LEVEL 1 AND 2 SUBSETS (ECMA SUBSET)

INTRODUCTION

The following report gives the exact working of the restrictions and changes necessary to change the report on the full language into a report on level 1 and level 2 subsets. Attention must be drawn to some slight deviations between this report and the worked in version in the boxes in Part I A. This is mainly caused by typographical difficulties. For example, where level 1 and 2 subsets require at least 26 letters in the alphabet, level 3 subset permits at most 26 letters.

The change listed under item 2 in the definitions does not apply to level 1 subset. This is the only difference between the subsets of levels 1 and 2.

DEFINITIONS AND REMARKS

Remarks to compiler designer	Remarks to programmer	Definition of Ecma subset in terms of ALGOL 60 Report (The numbers refer to sections and clauses of Part I A)
1. The declarator own need not be implemented.	The use of own variables should be avoided.	5. Delete first two sentences of fourth paragraph. 5.1.1 Delete “ own <type> ”. 5.1.3 Delete last sentence.
2. The recursive use of procedures and recursively defined procedures need not be allowed. No recursive use of procedures means that no call of the procedure itself may occur during the execution of the statements of the body of any procedure, and during the evaluation of those of its actual parameters, the corresponding formal parameters of which are called by name, and during the evaluation of expressions occurring in declarations inside the procedure.	Do not write recursive procedures. Do not use procedures recursively.	5.4.4 Delete last sentence. Add to 4.7.5: “ No call of the procedure itself may occur during the execution of the statements of the body of any procedure, and during the evaluation of those of its actual parameters, the corresponding formal parameters of which are called by name, and during the evaluation of expressions occurring in declarations inside the procedure.”
THIS CHANGE DOES NOT APPLY TO LEVEL 1 SUBSET		
3. Integer labels need not be provided for.	Do not use integer labels.	3.5.1 Delete “ <unsigned integer> ”. 3.5.5 Delete.

Remarks to compiler designer	Remarks to programmer	Definition of Ecma subset in terms of ALGOL 60 Report (The numbers refer to sections and clauses of Part I A)
4. Up to full specification for all parameters may be required. Up to equivalence of the type of an actual parameter and the specified type of the corresponding formal parameter, if called by name, may be required.	Provide each procedure declaration with a complete specification part and maintain equivalence of type between actual and formal parameter called by name.	5.4.5 Third sentence—replaced by “Specifications of all formal parameters if any must be supplied.” 4.7.5.5 Replaced by: “Kind and type of actual parameters must be the same as those of the corresponding formal parameters, if called by name.”
5. The alphabet may not be restricted to less than one case of 26 letters.	Use only one case of letters. (Either small or capital—the typing for a particular implementation will transfer if necessary to the right case).	2.1 Delete “ A ... Z” and “restricted, or”.
6. A limit may be put on the length of identifiers, but this must not be less than 6 significant basic symbols.	Do not rely on the differentiation between two identifiers which have the first six basic symbols in common.	2.4.3 Replace “They may be chosen freely” by “Identifiers may be chosen freely; but the effects due to the occurrence of two different identifiers the first six basic symbols of which are common are undefined”.
7. If the type of an arithmetic expression depends upon the evaluation of an expression or upon the type or value of an actual parameter then it may be taken to be real.	Be careful when making essential use of the type of an arithmetic expression.	3.3.4 Replace the words “the following rules” of the last sentence by “a set of rules. However if the type of an arithmetic expression according to the rules cannot be determined without evaluating an expression or ascertaining the type or value of an actual parameter, it is real. These rules are”.

Remarks to compiler designer	Remarks to programmer	Definition of Ecma subset in terms of ALGOL 60 Report (The numbers refer to sections and clauses of Part I A)
8. The requirement of 4.3.5 need not be implemented.	A go to statement involving an undefined switch designator need not have the effect of a dummy statement. You are well advised to program a check that the value of the subscript expression is within the bounds defined by the switch declaration.	4.3.5 Replace "equivalent to a dummy statement" by "undefined".

PART I C**REPORT ON LEVEL 3 SUBSET
(IFIP SUBSET)****INTRODUCTION**

The definition of level 3 subset appears in the form of informal explanations and changes to be made to the report on the full language ALGOL 60 to convert this into the level 3 subset (as has been done in the boxes in Part I A).

DEFINITIONS AND EXPLANATIONS

Section or clause of Part I A	Subset definitions	Explanation
2.3	Delete from definition of declarator "own ".	The own concept is not included in the subset.
5	Delete first two sentences of fourth paragraph.	
5.1.1	Replace the last two metalinguistic formulae by: "<type declaration> ::= <type> <type list>".	
5.1.3	Delete last sentence.	
5.2.1	Replace the last formula by: " <array declaration> ::= array <array list> <type> array <array list> ".	
5.2.5	Delete "even if an array is declared own ".	Recursive procedures and recursive use of procedures are not included.
4.7.5	Add clause 4.7.5.6: "No call of the procedure itself may occur during the execution of the statements of the body of any procedure, nor during the evaluation of those of its actual parameters, the corresponding formal parameters of which are called by name, nor during the evaluation of expressions occurring in declarations inside the procedure".	
5.4.4	Delete last sentence.	
3.5.1	Delete " <unsigned integer> ".	
3.5.5	Delete.	Integer labels are not provided for.
5.4.5	Replace third sentence by: "Specifications of all formal parameters if any must be supplied".	
4.7.5.5	Replace by: "Kind and type of actual parameters must be the same as those of the corresponding formal parameters, if called by name".	

Section or clause of Part I A	Subset definitions	Explanation
2.1	Delete: " $ A B \dots Y Z$ ". Delete: " , or extended ... delimiter)". Add: "If a particular implementation requires capitals rather than small letters, one must regard them as a hardware representation for the small letters".	Only one case of letters is provided for.
2.4.3	Replace: "They may be chosen freely" by: "Identifiers may be chosen freely; but the effects due to the occurrence of two different identifiers the first six basic symbols of which are common are undefined".	In the subset identifiers are differentiated only up to six leading basic symbols.
3.3.4	Replace the words: "the following rules" of the last sentence by: "a set of rules. However, if the type of an arithmetic expression according to the rules cannot be determined without evaluating an expression or ascertaining the type or value of an actual parameter, it is real . These rules are".	In the subset the type of an arithmetic expression will be in certain cases real where it will be integer in ALGOL 60. Thus arithmetic will be less precise in some cases.
4.3.5	Delete.	The effect of a go to statement involving an undefined switch designator is undefined in the subset.
5.4.4	Add to text: "A function designator must be such that all its possible uses in the form of a procedure statement are equivalent to dummy statements".	
3.3.4.3	Insert between "... rules" and " : " : "with the exception that, if both the basis a and the exponent i are of integer type, then the exponent has to be an unsigned integer, otherwise the result is undefined".	Exponentiation with integer basis and exponent is restricted in the subset.
2.3	Delete: " $ \div$ ".	The so-called integer division is not included in the subset.
3.3.1	Delete: " $ \div$ ".	
3.3.5.1	Delete: " $\div$ ".	

Section or clause of Part I A	Subset definitions	Explanation
3.3.4.2	Replace: "The operations ... both denote" by: "The operation $\langle \text{term} \rangle / \langle \text{factor} \rangle$ denotes". Delete last sentence.	
4.6.1	Replace: " $\langle \text{for clause} \rangle$... do" by: " $\langle \text{for clause} \rangle ::= \text{for } \langle \text{variable identifier} \rangle := \langle \text{for list} \rangle \text{ do}$ ".	The controlled variable in a for clause is restricted in the subset to be a variable identifier.
5.3.1	Replace: " $\langle \text{switch list} \rangle ::= \langle \text{designational expression} \rangle \langle \text{switch list} \rangle, \langle \text{designational expression} \rangle$ " by: " $\langle \text{switch list} \rangle ::= \langle \text{label} \rangle \langle \text{switch list} \rangle, \langle \text{label} \rangle$ ".	In the subset the designational expressions in a switch list are restricted to be labels only.
3.5.1	Replace the last two formulae by: " $\langle \text{designational expression} \rangle ::= \langle \text{label} \rangle \langle \text{switch designator} \rangle$ ".	In the subset only unconditional and unparenthesized designational expressions are provided for. See 5.3.1.
3.5.3	Delete: "In the general case ... is already found." Replace "selects one of the designational expressions ... a recursive process" by: "selects one of the labels contained in the switch list of the switch declaration. The selection is obtained by counting these labels from left to right".	
5.3.3	Replace: "These values ... its associated integer" by: "These values are given as labels entered in the switch list. With each of these labels is associated a positive integer 1, 2, ... obtained by counting the items in the list from left to right. The value of the switch designator corresponding to a given value of the subscript expression (see clause 3.5, "Designational expressions") is the label in the switch list having this given value as its associated integer".	
5.3.4	Delete.	

Section or clause of Part I A	Subset definitions	Explanation
5.3.5	Replace by: "Influence of scopes. If a switch designator occurs outside the scope of a label in the switch list, and an evaluation of this switch designator selects this label, then a possible conflict between the identifier used to denote this label and an identifier whose declaration is valid at the place of the switch designator will be avoided by a suitable change of this latter identifier".	
4.7.3.2	Replace: "after enclosing this ... syntactically possible" by: "this actual parameter being an identifier, or string, otherwise the name replacement is undefined".	In name replacement (call by name) the actual parameter can only be an identifier or a string.
4.7.5	Insert after: "... ALGOL statement" and before ".": "in the sense of this subset".	
5	Insert after "... any one block head." and before "Syntax": "The identifier associated with a quantity declared in a declaration may not occur denoting that quantity more than once between the begin of the block in whose head that declaration occurs and the semicolon which ends that declaration, excepting the case where this occurrence is the occurrence of a procedure identifier in the left part list of an assignment statement in the sense of clause 5.4.4".	
5.4.3	Add: "No identifier may occur more than once in a formal parameter list".	