It is a commonplace that computer programming is hard, especially when one aims at creating a program that is correct. What kind of methods should be used to reach that goal is the subject of heated debates. Our thesis is a contribution to these discussions: To understand what computer programming is, and how it should be done, we propose to study how it is actually done — that is, to induce elements of method from factual observation. Our thesis takes the form of a detailed analysis, based on a careful reconstruction, of a particular well-crafted computer program: the first ALGOL 60 system, designed and implemented at the Mathematical Center (now CWI) by E. W. Dijkstra and J. A. Zonneveld, with the assistance of S. J. Christen and M. J. H. Römgens, on an Electrologica X1 computer. It is divided into three main chapters. Chapter I presents the two elements of the problem the Mathematical Center team was facing, namely the ALGOL 60 language and the X1 computer. Chapter II discusses the principles of its solution, explains the implementation choices made by the Mathematical Center team, and compares them to other possible choices. Chapter III presents the details of the Mathematical Center ALGOL 60 system, on an ISO C version of that system, reverse engineered from its X1 assembler source. This program is about 3000 lines long, and is composed of 173 subroutines working on 57 global variables. Finally, our conclusion, in the form of 17 theses and 4 hypotheses, indicates some lessons, in particular on computer programming methods, that we believe can be drawn from the analysis of that particular computer program.

New Insights from Old Programs — The Structure of The First ALGOL 60 System

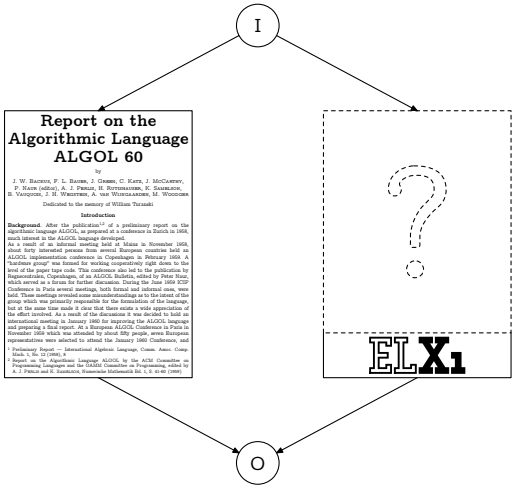
Gauthier van den Hove

Gauthier van den Hove

New Insights from Old Programs

The Structure of The First ALGOL 60 System

With a Foreword by Donald E. Knuth



Copyright © 2017 by Gauthier van den Hove <ghe@fibonacci.org>. All rights reserved. This work may not be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the author. The cover, inside covers, and pp. 1–2, 7–9, 13–15 and 249–302 of this work can however be freely distributed.

Algorithm index: A, 94; C_1, 53; C_2, 55; C_3, 56; C_4, 57; C_5, 65; C_6, 71; C_7, 72; C_8, 73; C_9, 79; C_{10}, 88; C_{11}, 89; C, 96; D_1, 63; D_2, 74; D_3, 76; D_4, 89; D, 97; E_1, 59; E_2, 60; E_3, 75; E_4, 75; E, 98; I, 96; J_1, 77; J, 98; R_1, 53; R_2, 55; R_3, 56; R_4, 57; R_5, 66; R_6, 71; R_7, 72; R_8, 73; R_9, 79; R_{10}, 88; R_{11}, 89; R, 97; X, 61; Y_1, 106; Y_2, 111; Y_3, 113; Y_4, 115; Y_5, 118; Y_6, 124; Y, 128.

1. ⟨ *empty* $_{24,40,77,101,103,105}$ ⟩ ::=
2. ⟨ *basic symbol* $_{(24)}$ ⟩ ::= ⟨ *letter* $_3$ ⟩ | ⟨ *digit* $_4$ ⟩ | ⟨ *logical value* $_5$ ⟩ | ⟨ *delimiter* $_6$ ⟩
3. ⟨ *letter* $_{2,16,37}$ ⟩ ::= *a* | *b* | *c* | *d* | *e* | *f* | *g* | *h* | *i* | *j* | *k* | *l* | *m* | *n* | *o* | *p* | *q* | *r* | *s* | *t* | *u* | *v* | *w* | *x* | *y* | *z*
| *A* | *B* | *C* | *D* | *E* | *F* | *G* | *H* | *I* | *J* | *K* | *L* | *M* | *N* | *O* | *P* | *Q* | *R* | *S* | *T* | *U* | *V* | *W* | *X* | *Y* | *Z*
4. ⟨ *digit* $_{2,16,17}$ ⟩ ::= *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9*
5. ⟨ *logical value* $_{2,51}$ ⟩ ::= **true** | **false**
6. ⟨ *delimiter* $_2$ ⟩ ::= ⟨ *operator* $_7$ ⟩ | ⟨ *separator* $_{12}$ ⟩ | ⟨ *bracket* $_{13}$ ⟩ | ⟨ *declarator* $_{14}$ ⟩ | ⟨ *specificator* $_{15}$ ⟩
7. ⟨ *operator* $_6$ ⟩ ::= ⟨ *arithmetic operator* $_8$ ⟩ | ⟨ *relational operator* $_9$ ⟩ | ⟨ *logical operator* $_{10}$ ⟩ | ⟨ *sequential operator* $_{11}$ ⟩
8. ⟨ *arithmetic operator* $_7$ ⟩ ::= + | − | × | / | ÷ | ↑
9. ⟨ *relational operator* $_{7,50}$ ⟩ ::= < | ≦ | = | ≧ | > | ≠
10. ⟨ *logical operator* $_7$ ⟩ ::= ≡ | ⊃ | ∨ | ∧ | ¬
11. ⟨ *sequential operator* $_7$ ⟩ ::= **go to** | **if** | **then** | **else** | **for** | **do**
12. ⟨ *separator* $_6$ ⟩ ::= , | . | 10 | : | ; | := | ␣ | **step** | **until** | **while** | **comment**
13. ⟨ *bracket* $_6$ ⟩ ::= (|) | [|] | ‘ | ’ | **begin** | **end**
14. ⟨ *declarator* $_6$ ⟩ ::= **own** | **Boolean** | **integer** | **real** | **array** | **switch** | **procedure**
15. ⟨ *specificator* $_6$ ⟩ ::= **string** | **label** | **value**
16. ⟨ *identifier* $_{16,28,32,35,58,59,99,102}$ ⟩ ::= ⟨ *letter* $_3$ ⟩ | ⟨ *identifier* $_{16}$ ⟩ ⟨ *letter* $_3$ ⟩ | ⟨ *identifier* $_{16}$ ⟩ ⟨ *digit* $_4$ ⟩
17. ⟨ *unsigned integer* $_{17,18,19,21,58}$ ⟩ ::= ⟨ *digit* $_4$ ⟩ | ⟨ *unsigned integer* $_{17}$ ⟩ ⟨ *digit* $_4$ ⟩
18. ⟨ *integer* $_{20}$ ⟩ ::= ⟨ *unsigned integer* $_{17}$ ⟩ | + ⟨ *unsigned integer* $_{17}$ ⟩ | − ⟨ *unsigned integer* $_{17}$ ⟩
19. ⟨ *decimal fraction* $_{21}$ ⟩ ::= . ⟨ *unsigned integer* $_{17}$ ⟩
20. ⟨ *exponent part* $_{22}$ ⟩ ::= 10 ⟨ *integer* $_{18}$ ⟩
21. ⟨ *decimal number* $_{22}$ ⟩ ::= ⟨ *unsigned integer* $_{17}$ ⟩ | ⟨ *decimal fraction* $_{19}$ ⟩ | ⟨ *unsigned integer* $_{17}$ ⟩ ⟨ *decimal fraction* $_{19}$ ⟩
22. ⟨ *unsigned number* $_{23,44}$ ⟩ ::= ⟨ *decimal number* $_{21}$ ⟩ | ⟨ *exponent part* $_{20}$ ⟩ | ⟨ *decimal number* $_{21}$ ⟩ ⟨ *exponent part* $_{20}$ ⟩
23. ⟨ *number* ⟩ ::= ⟨ *unsigned number* $_{22}$ ⟩ | + ⟨ *unsigned number* $_{22}$ ⟩ | − ⟨ *unsigned number* $_{22}$ ⟩
24. ⟨ *proper string* $_{25}$ ⟩ ::= ⟨ *any sequence of basic symbols not containing ‘ or ’* $_{(2)}$ ⟩ | ⟨ *empty* $_1$ ⟩
25. ⟨ *open string* $_{25,26}$ ⟩ ::= ⟨ *proper string* $_{24}$ ⟩ | ‘ ⟨ *open string* $_{25}$ ⟩ ’ | ⟨ *open string* $_{25}$ ⟩ ⟨ *open string* $_{25}$ ⟩
26. ⟨ *string* $_{36}$ ⟩ ::= ‘ ⟨ *open string* $_{25}$ ⟩ ’
27. ⟨ *expression* $_{36}$ ⟩ ::= ⟨ *arithmetic expression* $_{49}$ ⟩ | ⟨ *Boolean expression* $_{57}$ ⟩ | ⟨ *designational expression* $_{62}$ ⟩
28. ⟨ *variable identifier* $_{29}$ ⟩ ::= ⟨ *identifier* $_{16}$ ⟩
29. ⟨ *simple variable* $_{34,86}$ ⟩ ::= ⟨ *variable identifier* $_{28}$ ⟩
30. ⟨ *subscript expression* $_{31,60}$ ⟩ ::= ⟨ *arithmetic expression* $_{49}$ ⟩
31. ⟨ *subscript list* $_{31,33}$ ⟩ ::= ⟨ *subscript expression* $_{30}$ ⟩ | ⟨ *subscript list* $_{31}$ ⟩ , ⟨ *subscript expression* $_{30}$ ⟩
32. ⟨ *array identifier* $_{33,36,94}$ ⟩ ::= ⟨ *identifier* $_{16}$ ⟩
33. ⟨ *subscripted variable* $_{34}$ ⟩ ::= ⟨ *array identifier* $_{32}$ ⟩ [⟨ *subscript list* $_{31}$ ⟩]
34. ⟨ *variable* $_{44,51,73,82}$ ⟩ ::= ⟨ *simple variable* $_{29}$ ⟩ | ⟨ *subscripted variable* $_{33}$ ⟩
35. ⟨ *procedure identifier* $_{36,41,84,106}$ ⟩ ::= ⟨ *identifier* $_{16}$ ⟩
36. ⟨ *actual parameter* $_{39}$ ⟩ ::= ⟨ *string* $_{26}$ ⟩ | ⟨ *expression* $_{27}$ ⟩ | ⟨ *array identifier* $_{32}$ ⟩ | ⟨ *switch identifier* $_{59}$ ⟩
| ⟨ *procedure identifier* $_{35}$ ⟩
37. ⟨ *letter string* $_{37,38}$ ⟩ ::= ⟨ *letter* $_3$ ⟩ | ⟨ *letter string* $_{37}$ ⟩ ⟨ *letter* $_3$ ⟩
38. ⟨ *parameter delimiter* $_{39,100}$ ⟩ ::= , |) ⟨ *letter string* $_{37}$ ⟩ : (
39. ⟨ *actual parameter list* $_{39,40}$ ⟩ ::= ⟨ *actual parameter* $_{36}$ ⟩
| ⟨ *actual parameter list* $_{39}$ ⟩ ⟨ *parameter delimiter* $_{38}$ ⟩ ⟨ *actual parameter* $_{36}$ ⟩
40. ⟨ *actual parameter part* $_{41,84}$ ⟩ ::= ⟨ *empty* $_1$ ⟩ | (⟨ *actual parameter list* $_{39}$ ⟩)
41. ⟨ *function designator* $_{44,51}$ ⟩ ::= ⟨ *procedure identifier* $_{35}$ ⟩ ⟨ *actual parameter part* $_{40}$ ⟩
42. ⟨ *adding operator* $_{47}$ ⟩ ::= + | −
43. ⟨ *multiplying operator* $_{46}$ ⟩ ::= × | / | ÷
44. ⟨ *primary* $_{45}$ ⟩ ::= ⟨ *unsigned number* $_{22}$ ⟩ | ⟨ *variable* $_{34}$ ⟩ | ⟨ *function designator* $_{41}$ ⟩ | (⟨ *arithmetic expression* $_{49}$ ⟩)
45. ⟨ *factor* $_{45,46}$ ⟩ ::= ⟨ *primary* $_{44}$ ⟩ | ⟨ *factor* $_{45}$ ⟩ ↑ ⟨ *primary* $_{44}$ ⟩
46. ⟨ *term* $_{46,47}$ ⟩ ::= ⟨ *factor* $_{45}$ ⟩ | ⟨ *term* $_{46}$ ⟩ ⟨ *multiplying operator* $_{43}$ ⟩ ⟨ *factor* $_{45}$ ⟩
47. ⟨ *simple arithmetic expression* $_{47,49}$ ⟩ ::= ⟨ *term* $_{46}$ ⟩ | ⟨ *adding operator* $_{42}$ ⟩ ⟨ *term* $_{46}$ ⟩
| ⟨ *simple arithmetic expression* $_{47}$ ⟩ ⟨ *adding operator* $_{42}$ ⟩ ⟨ *term* $_{46}$ ⟩
48. ⟨ *if clause* $_{49,57,62,78}$ ⟩ ::= **if** ⟨ *Boolean expression* $_{57}$ ⟩ **then**
49. ⟨ *arithmetic expression* $_{27,30,44,49,50,75,80,90,91}$ ⟩ ::= ⟨ *simple arithmetic expression* $_{47}$ ⟩
| ⟨ *if clause* $_{48}$ ⟩ ⟨ *simple arithmetic expression* $_{47}$ ⟩ **else** ⟨ *arithmetic expression* $_{49}$ ⟩
50. ⟨ *relation* $_{51}$ ⟩ ::= ⟨ *arithmetic expression* $_{49}$ ⟩ ⟨ *relational operator* $_9$ ⟩ ⟨ *arithmetic expression* $_{49}$ ⟩
51. ⟨ *Boolean primary* $_{52}$ ⟩ ::= ⟨ *logical value* $_5$ ⟩ | ⟨ *variable* $_{34}$ ⟩ | ⟨ *function designator* $_{41}$ ⟩ | ⟨ *relation* $_{50}$ ⟩
| (⟨ *Boolean expression* $_{57}$ ⟩)
52. ⟨ *Boolean secondary* $_{53}$ ⟩ ::= ⟨ *Boolean primary* $_{51}$ ⟩ | ¬ ⟨ *Boolean primary* $_{51}$ ⟩
53. ⟨ *Boolean factor* $_{53,54}$ ⟩ ::= ⟨ *Boolean secondary* $_{52}$ ⟩ | ⟨ *Boolean factor* $_{53}$ ⟩ ∧ ⟨ *Boolean secondary* $_{52}$ ⟩
54. ⟨ *Boolean term* $_{54,55}$ ⟩ ::= ⟨ *Boolean factor* $_{53}$ ⟩ | ⟨ *Boolean term* $_{54}$ ⟩ ∨ ⟨ *Boolean factor* $_{53}$ ⟩
55. ⟨ *implication* $_{55,56}$ ⟩ ::= ⟨ *Boolean term* $_{54}$ ⟩ | ⟨ *implication* $_{55}$ ⟩ ⊃ ⟨ *Boolean term* $_{54}$ ⟩

Copyright © 2017 by Gauthier van den Hove <ghe@fibonacci.org>. All rights reserved. This work may not be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the author. The cover, inside covers, and pp. 1–2, 7–9, 13–15 and 249–302 of this work can however be freely distributed.

56. ⟨*simple Boolean*$_{56,57}$⟩ ::= ⟨*implication*$_{55}$⟩ | ⟨*simple Boolean*$_{56}$⟩ ≡ ⟨*implication*$_{55}$⟩
57. ⟨*Boolean expression*$_{27,48,51,57,75,80}$⟩ ::= ⟨*simple Boolean*$_{56}$⟩
| ⟨*if clause*$_{48}$⟩ ⟨*simple Boolean*$_{56}$⟩ **else** ⟨*Boolean expression*$_{57}$⟩
58. ⟨*label*$_{61,64,71,72,78,83}$⟩ ::= ⟨*identifier*$_{16}$⟩ | ⟨*unsigned integer*$_{17}$⟩
59. ⟨*switch identifier*$_{36,60,98}$⟩ ::= ⟨*identifier*$_{16}$⟩
60. ⟨*switch designator*$_{61}$⟩ ::= ⟨*switch identifier*$_{59}$⟩ [⟨*subscript expression*$_{30}$⟩]
61. ⟨*simple designational expression*$_{62}$⟩ ::= ⟨*label*$_{58}$⟩ | ⟨*switch designator*$_{60}$⟩ | (⟨*designational expression*$_{62}$⟩)
62. ⟨*designational expression*$_{27,61,62,76,97}$⟩ ::= ⟨*simple designational expression*$_{61}$⟩
| ⟨*if clause*$_{48}$⟩ ⟨*simple designational expression*$_{61}$⟩ | **else** ⟨*designational expression*$_{62}$⟩
63. ⟨*unlabeled basic statement*$_{64}$⟩ ::= ⟨*assignment statement*$_{75}$⟩ | ⟨*go to statement*$_{76}$⟩ | ⟨*dummy statement*$_{77}$⟩
| ⟨*procedure statement*$_{84}$⟩
64. ⟨*basic statement*$_{64,65}$⟩ ::= ⟨*unlabeled basic statement*$_{63}$⟩ | ⟨*label*$_{58}$⟩ : ⟨*basic statement*$_{64}$⟩
65. ⟨*unconditional statement*$_{66,78}$⟩ ::= ⟨*basic statement*$_{64}$⟩ | ⟨*for statement*$_{83}$⟩ | ⟨*compound statement*$_{72}$⟩
| ⟨*block*$_{71}$⟩
66. ⟨*statement*$_{67,79,83,107}$⟩ ::= ⟨*unconditional statement*$_{65}$⟩ | ⟨*conditional statement*$_{79}$⟩
67. ⟨*compound tail*$_{67,69,70}$⟩ ::= ⟨*statement*$_{66}$⟩ **end** | ⟨*statement*$_{66}$⟩ ; ⟨*compound tail*$_{67}$⟩
68. ⟨*block head*$_{68,70}$⟩ ::= **begin** ⟨*declaration*$_{85}$⟩ | ⟨*block head*$_{68}$⟩ ; ⟨*declaration*$_{85}$⟩
69. ⟨*unlabeled compound*$_{72}$⟩ ::= **begin** ⟨*compound tail*$_{67}$⟩
70. ⟨*unlabeled block*$_{71}$⟩ ::= ⟨*block head*$_{68}$⟩ ; ⟨*compound tail*$_{67}$⟩
71. ⟨*block*$_{65,71}$⟩ ::= ⟨*unlabeled block*$_{70}$⟩ | ⟨*label*$_{58}$⟩ : ⟨*block*$_{71}$⟩
72. ⟨*compound statement*$_{65,72}$⟩ ::= ⟨*unlabeled compound*$_{69}$⟩ | ⟨*label*$_{58}$⟩ : ⟨*compound statement*$_{72}$⟩
73. ⟨*left part*$_{74}$⟩ ::= ⟨*variable*$_{34}$⟩ :=
74. ⟨*left part list*$_{74,75}$⟩ ::= ⟨*left part*$_{73}$⟩ | ⟨*left part list*$_{74}$⟩ ⟨*left part*$_{73}$⟩
75. ⟨*assignment statement*$_{63}$⟩ ::= ⟨*left part list*$_{74}$⟩ ⟨*arithmetic expression*$_{49}$⟩
| ⟨*left part list*$_{74}$⟩ ⟨*Boolean expression*$_{57}$⟩
76. ⟨*go to statement*$_{63}$⟩ ::= **go to** ⟨*designational expression*$_{62}$⟩
77. ⟨*dummy statement*$_{63}$⟩ ::= ⟨*empty*$_{1}$⟩
78. ⟨*if statement*$_{78,79}$⟩ ::= ⟨*if clause*$_{48}$⟩ ⟨*unconditional statement*$_{65}$⟩ | ⟨*label*$_{58}$⟩ : ⟨*if statement*$_{78}$⟩
79. ⟨*conditional statement*$_{66}$⟩ ::= ⟨*if statement*$_{78}$⟩ | ⟨*if statement*$_{78}$⟩ **else** ⟨*statement*$_{66}$⟩
80. ⟨*for list element*$_{81}$⟩ ::= ⟨*arithmetic expression*$_{49}$⟩
| ⟨*arithmetic expression*$_{49}$⟩ **step** ⟨*arithmetic expression*$_{49}$⟩ **until** ⟨*arithmetic expression*$_{49}$⟩
| ⟨*arithmetic expression*$_{49}$⟩ **while** ⟨*Boolean expression*$_{57}$⟩
81. ⟨*for list*$_{81,82}$⟩ ::= ⟨*for list element*$_{80}$⟩ | ⟨*for list*$_{81}$⟩ , ⟨*for list element*$_{80}$⟩
82. ⟨*for clause*$_{83}$⟩ ::= **for** ⟨*variable*$_{34}$⟩ := ⟨*for list*$_{81}$⟩ **do**
83. ⟨*for statement*$_{65,83}$⟩ ::= ⟨*for clause*$_{82}$⟩ ⟨*statement*$_{66}$⟩ | ⟨*label*$_{58}$⟩ : ⟨*for statement*$_{83}$⟩
84. ⟨*procedure statement*$_{63}$⟩ ::= ⟨*procedure identifier*$_{35}$⟩ ⟨*actual parameter part*$_{40}$⟩
85. ⟨*declaration*$_{68}$⟩ ::= ⟨*type declaration*$_{89}$⟩ | ⟨*array declaration*$_{96}$⟩ | ⟨*switch declaration*$_{98}$⟩
| ⟨*procedure declaration*$_{108}$⟩
86. ⟨*type list*$_{86,89}$⟩ ::= ⟨*simple variable*$_{29}$⟩ | ⟨*simple variable*$_{29}$⟩ , ⟨*type list*$_{86}$⟩
87. ⟨*type*$_{88,104,108}$⟩ ::= **real** | **integer** | **Boolean**
88. ⟨*local or own type*$_{89,96}$⟩ ::= ⟨*type*$_{87}$⟩ | **own** ⟨*type*$_{87}$⟩
89. ⟨*type declaration*$_{85}$⟩ ::= ⟨*local or own type*$_{88}$⟩ ⟨*type list*$_{86}$⟩
90. ⟨*lower bound*$_{92}$⟩ ::= ⟨*arithmetic expression*$_{49}$⟩
91. ⟨*upper bound*$_{92}$⟩ ::= ⟨*arithmetic expression*$_{49}$⟩
92. ⟨*bound pair*$_{93}$⟩ ::= ⟨*lower bound*$_{90}$⟩ : ⟨*upper bound*$_{91}$⟩
93. ⟨*bound pair list*$_{93,94}$⟩ ::= ⟨*bound pair*$_{92}$⟩ | ⟨*bound pair list*$_{93}$⟩ , ⟨*bound pair*$_{92}$⟩
94. ⟨*array segment*$_{94,95}$⟩ ::= ⟨*array identifier*$_{32}$⟩ [⟨*bound pair list*$_{93}$⟩] | ⟨*array identifier*$_{32}$⟩ , ⟨*array segment*$_{94}$⟩
95. ⟨*array list*$_{95,96}$⟩ ::= ⟨*array segment*$_{94}$⟩ | ⟨*array list*$_{95}$⟩ , ⟨*array segment*$_{94}$⟩
96. ⟨*array declaration*$_{85}$⟩ ::= **array** ⟨*array list*$_{95}$⟩ | ⟨*local or own type*$_{88}$⟩ **array** ⟨*array list*$_{95}$⟩
97. ⟨*switch list*$_{97,98}$⟩ ::= ⟨*designational expression*$_{62}$⟩ | ⟨*switch list*$_{97}$⟩ , ⟨*designational expression*$_{62}$⟩
98. ⟨*switch declaration*$_{85}$⟩ ::= **switch** ⟨*switch identifier*$_{59}$⟩ := ⟨*switch list*$_{97}$⟩
99. ⟨*formal parameter*$_{100}$⟩ ::= ⟨*identifier*$_{16}$⟩
100. ⟨*formal parameter list*$_{100,101}$⟩ ::= ⟨*formal parameter*$_{99}$⟩
| ⟨*formal parameter list*$_{100}$⟩ ⟨*parameter delimiter*$_{38}$⟩ ⟨*formal parameter*$_{99}$⟩
101. ⟨*formal parameter part*$_{106}$⟩ ::= ⟨*empty*$_{1}$⟩ | (⟨*formal parameter list*$_{100}$⟩)
102. ⟨*identifier list*$_{102,103,105}$⟩ ::= ⟨*identifier*$_{16}$⟩ | ⟨*identifier list*$_{102}$⟩ , ⟨*identifier*$_{16}$⟩
103. ⟨*value part*$_{106}$⟩ ::= **value** ⟨*identifier list*$_{102}$⟩ ; | ⟨*empty*$_{1}$⟩
104. ⟨*specifier*$_{105}$⟩ ::= **string** | ⟨*type*$_{87}$⟩ | **array** | ⟨*type*$_{87}$⟩ **array** | **label** | **switch** | **procedure** | ⟨*type*$_{87}$⟩ **procedure**
105. ⟨*specification part*$_{105,106}$⟩ ::= ⟨*empty*$_{1}$⟩ | ⟨*specifier*$_{104}$⟩ ⟨*identifier list*$_{102}$⟩ ;
| ⟨*specification part*$_{105}$⟩ ⟨*specifier*$_{104}$⟩ ⟨*identifier list*$_{102}$⟩ ;
106. ⟨*procedure heading*$_{108}$⟩ ::= ⟨*procedure identifier*$_{35}$⟩ ⟨*formal parameter part*$_{101}$⟩ ;
⟨*value part*$_{103}$⟩ ⟨*specification part*$_{105}$⟩
107. ⟨*procedure body*$_{108}$⟩ ::= ⟨*statement*$_{66}$⟩
108. ⟨*procedure declaration*$_{85}$⟩ ::= **procedure** ⟨*procedure heading*$_{106}$⟩ ⟨*procedure body*$_{107}$⟩
| ⟨*type*$_{87}$⟩ **procedure** ⟨*procedure heading*$_{106}$⟩ ⟨*procedure body*$_{107}$⟩

Copyright © 2017 by Gauthier van den Hove <ghe@fibonacci.org>. All rights reserved. This work may not be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the author. The cover, inside covers, and pp. 1–2, 7–9, 13–15 and 249–302 of this work can however be freely distributed.

ADD, operation, 187, 200.
add idents from pre scan list, subroutine, 148.
add label to pre scan list, macro, 142.
add procedure to pre scan list, macro, 142.
add switch to pre scan list, macro, 142.
add to future refs if ident not yet encountered, subroutine, 148.
add to pre scan list, subroutine, 142.
address mask, constant, 146.
address of RFP, variable, 197.
address type, constant, 196.
after subscript, variable, 179.
AND, operation, 187, 202.
append, subroutine, 150.
append address, subroutine, 156.
append data, macro, 150.
append enter block if necessary, subroutine, 157.
append future, macro, 150.
append leave block, subroutine, 158.
append load address, subroutine, 174.
append load non formal address, subroutine, 174.
append load value, subroutine, 182.
append op, macro, 150.
append procedure call, subroutine, 177.
at begin of expression, variable, 167.
augment pre scan list, subroutine, 141.
block number, variable, 148.
block number mask, constant, 147.
block number position, constant, 147.
bn, variable, 198.
boolean, type, 139.
both integers or convert, subroutine, 199.
build storage function of dynamic array, subroutine, 205.
build storage function of static array, subroutine, 165.
call explicit procedure, subroutine, 211.
CEP, operation, 157, 175, 177, 211.
CEPR, operation, 177, 212.
CFP, operation, 177, 217.
CFPR, operation, 177, 217.
consts, variable, 145.
consts append, subroutine, 149.
consts index, variable, 145.
consts ref type, constant, 146.
copy storage function of value array, subroutine, 206.
cur delim, variable, 140.
cur descr, variable, 146.
cur descr index, variable, 147.
cur tok, variable, 140.
cur tok is formal, variable, 147.
cur tok is integer or real, variable, 140.
cur tok is label or switch, variable, 147.
cur tok is procedure, variable, 147.
cur tok is real, variable, 140.
cur tok is set, variable, 140.
current block index, variable, 142.
display, variable, 198.
DIV, operation, 187, 201.
do generate for local and value arrays, variable, 156.
dynamic arrays counter, variable, 167.
dynamic arrays dimension, variable, 167.
dynamic to static address, subroutine, 202.
end of execution, variable, 198.
end of preliminary scan, variable, 144.
end of translation, variable, 193.
EQ, operation, 187, 202.
equal, subroutine, 201.
EQV, operation, 187, 202.
execution types, type, 196.
false, constant, 139.
fill link and create parameter descriptions, subroutine, 210.
fix future ref, subroutine, 146.
for CEP future ref, variable, 174.
for FOR0 future ref, variable, 175.
for statement future ref, variable, 175.
FOR0, operation, 176, 219.
FOR1, operation, 175, 222.
FOR2, operation, 176, 220.
FOR3, operation, 175, 220.
FOR4, operation, 176, 220.
FOR5, operation, 175, 221.
FOR6, operation, 175, 221.
FOR7, operation, 176, 222.
FOR8, operation, 176, 223.
fp, variable, 198.
future ref type, constant, 146.
future refs, variable, 145.
future refs index, variable, 145.
generate at end of elementary expression, subroutine, 186.
generate at end of expression, subroutine, 191.
generate at end of for statement, subroutine, 177.
generate at end of statement, subroutine, 192.
generate for local and value arrays, subroutine, 156.
generate for simple local variables, subroutine, 155.
generate operations at operator with priority, subroutine, 187.
GEQ, operation, 187, 202.
get address, macro, 146.
get array size, subroutine, 205.
get block number, macro, 147.
get parameter, macro, 198.
get parameter type, macro, 178.
get ref type, macro, 146.
get type, macro, 147.
greater, subroutine, 201.
grow, subroutine, 139.
GTA, operation, 184, 218.
has not been encountered bit, constant, 147.
ident stack, variable, 145.
ident stack top, variable, 145.
IDI, operation, 187, 201.
if integer convert to real, subroutine, 199.
if real convert to integer, subroutine, 199.
IMP, operation, 187, 202.
in actual parameter list, variable, 179.
in expression, variable, 167.
in for clause, variable, 174.
in integer variable declaration, variable, 166.
in procedure heading, variable, 143.
in subscript, variable, 179.
in switch declaration, variable, 162.
IND, operation, 183, 208.
integer, type, 139.
integer addition would overflow, subroutine, 200.
integer multiplication would overflow, subroutine, 200.
integer type, constant, 196.
is by value or array bit, constant, 147.
is dynamic ident bit, constant, 147.
is dynamic parameter, constant, 178.
is formal ident bit, constant, 147.
is formal parameter, constant, 178.
is integer, constant, 147.
is label or switch ident bit, constant, 147.
is local array bit, constant, 147.
is procedure ident bit, constant, 147.
is procedure with address result, constant, 147.
is procedure with non address result, constant, 147.
is real, constant, 147.
is static parameter, constant, 178.
ISF, operation, 168, 205.
IVA, operation, 156, 206.
JMPF, operation, 190, 217.
JMPS, operation, 163, 217.
JMPU, operation, 157, 160, 162, 174, 175, 176, 177, 179, 184, 191, 217.
LAP, operation, 157, 206.
LEQ, operation, 187, 202.
LES, operation, 187, 202.
link size, constant, 157.
load, subroutine, 197.
local simple variables counter, variable, 155.
local variables index, variable, 156.

Copyright © 2017 by Gauthier van den Hove <ghe@fibonacci.org>. All rights reserved. This work may not be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the author. The cover, inside covers, and pp. 1–2, 7–9, 13–15 and 249–302 of this work can however be freely distributed.

lookup and add to consts, subroutine, 149.
lookup ident, subroutine, 147.
main, subroutine, 224.
main stack, variable, 140.
main stack top, variable, 140.
max value array dimensions, constant, 156.
mem, variable, 196.
MOR, operation, 187, 202.
MUL, operation, 187, 201.
NEG, operation, 187, 199.
NEQ, operation, 187, 202.
NON, operation, 187, 202.
non ref type, constant, 146.
object program, variable, 145.
object program index, variable, 145.
OR, operation, 187, 202.
panic, subroutine, 147.
parameter number, variable, 198.
parameter type mask, constant, 178.
pc, variable, 198.
peek, macro, 141.
peek delim, macro, 150.
peek priority, macro, 187.
pop, macro, 141.
pre scan list, variable, 140.
pre scan list end, variable, 140.
preliminary scan, subroutine, 145.
preliminary scan process begin, subroutine, 143.
preliminary scan process block begin, subroutine, 142.
preliminary scan process declaration, subroutine, 143.
preliminary scan process end or semicolon, subroutine, 144.
priority mask, constant, 150.
priority position, constant, 150.
process array, subroutine, 171.
process array reference, subroutine, 183.
process assign, subroutine, 194.
process begin, subroutine, 192.
process begin of dynamic array variables, subroutine, 167.
process begin of procedure statement with parameters, subroutine, 179.
process colon, subroutine, 194.
process comma, subroutine, 194.
process comparison, subroutine, 188.
process delimiter, subroutine, 195.
process do, subroutine, 176.
process dynamic array variables colon or comma, subroutine, 167.
process dynamic simple variables, subroutine, 169.
process else, subroutine, 191.
process end of dynamic array variables, subroutine, 168.
process end of procedure statement with parameters, subroutine, 182.
process end or semicolon, subroutine, 193.
process for, subroutine, 174.
process for assign, subroutine, 175.
process for comma, subroutine, 176.
process formal parameter list, subroutine, 158.
process go to, subroutine, 172.
process if, subroutine, 190.
process integer or Boolean, subroutine, 170.
process label colon, subroutine, 171.
process label or switch ident, subroutine, 184.
process left bracket, subroutine, 183.
process left parenthesis, subroutine, 194.
process logical operation, subroutine, 188.
process own, subroutine, 171.
process parameter comma, subroutine, 181.
process plus or minus, subroutine, 188.
process procedure, subroutine, 160.
process procedure assign, subroutine, 189.
process procedure or label or switch ident declaration, subroutine, 149.
process procedure statement without parameters, subroutine, 177.
process real, subroutine, 170.
process right bracket, subroutine, 185.
process right parenthesis, subroutine, 195.
process static array variables, subroutine, 166.
process static simple variables, subroutine, 168.
process step, subroutine, 175.
process switch, subroutine, 162.
process switch assign, subroutine, 162.
process switch comma, subroutine, 163.
process switch reference, subroutine, 184.
process switch semicolon, subroutine, 163.
process then, subroutine, 190.
process times or divide, subroutine, 188.
process to the power, subroutine, 187.
process until, subroutine, 175.
process value and specification lists, subroutine, 159.
process variable assign, subroutine, 188.
process while, subroutine, 175.
prune, macro, 141.
push, subroutine, 140.
push cur delim with priority, macro, 150.
push on ident stack, subroutine, 146.
put block number, macro, 147.
put ref type, macro, 146.
read toks until next delimiter, subroutine, 140.
real, type, 139.
real type, constant, 196.
ref type mask, constant, 146.
ref type position, constant, 146.
REP, operation, 158, 213.
RFP, operation, 197, 215.
RIP, operation, 181, 214.
RSF, operation, 168, 205.
RSV, operation, 155, 204.
run, subroutine, 223.
RVA, operation, 156, 206.
SCC, operation, 157, 160, 213.
set, macro, 141.
short circuit, subroutine, 213.
si, variable, 198.
sign, subroutine, 221.
skip read toks, variable, 140.
sp, variable, 198.
SSI, operation, 184, 218.
ST, operation, 159, 187, 203.
STA, operation, 187, 204.
STAP, operation, 187, 212.
START, operation, 196, 198.
static variable ref type, constant, 146.
static variables end address, variable, 197.
static variables index, variable, 166.
STOP, operation, 193, 198.
store procedure result, subroutine, 212.
STP, operation, 187, 212.
SUB, operation, 187, 200.
take address, subroutine, 202.
take formal, subroutine, 216.
take result, subroutine, 203.
TER, operation, 182, 186, 203.
TFA, operation, 174, 217.
TFR, operation, 159, 182, 184, 217.
TIAD, operation, 159, 174, 203.
TIAS, operation, 174, 203.
TRAD, operation, 159, 174, 203.
translate, subroutine, 196.
translation types, type, 146.
TRAS, operation, 174, 203.
true, constant, 139.
TTP, operation, 187, 201.
type mask, constant, 147.
type position, constant, 147.
typed word, type, 139.
unspecified type, constant, 196.
update display, subroutine, 213.
VAP, operation, 157, 207.
word, type, 139.
wp, variable, 198.

Copyright © 2017 by Gauthier van den Hove <ghe@fibonacci.org>. All rights reserved. This work may not be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the author. The cover, inside covers, and pp. 1–2, 7–9, 13–15 and 249–302 of this work can however be freely distributed.