

---

---

**Information technology — Programming  
languages — COBOL**

*Technologies de l'information — Langages de programmation — COBOL*



Reference number  
ISO/IEC 1989:2002(E)

**PDF disclaimer**

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2002

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Case postale 56 • CH-1211 Geneva 20  
Tel. + 41 22 749 01 11  
Fax + 41 22 749 09 47  
E-mail [copyright@iso.ch](mailto:copyright@iso.ch)  
Web [www.iso.ch](http://www.iso.ch)

Printed in Switzerland

# Contents

|                                                                                         |              |
|-----------------------------------------------------------------------------------------|--------------|
| <b>Contents</b> .....                                                                   | <b>iii</b>   |
| <b>Tables</b> .....                                                                     | <b>xvii</b>  |
| <b>Figures</b> .....                                                                    | <b>xviii</b> |
| <b>Foreword</b> .....                                                                   | <b>xix</b>   |
| <b>Introduction</b> .....                                                               | <b>xx</b>    |
| <b>1 Scope</b> .....                                                                    | <b>1</b>     |
| <b>2 Normative references</b> .....                                                     | <b>2</b>     |
| <b>3 Conformance to this International Standard</b> .....                               | <b>3</b>     |
| 3.1 A conforming implementation .....                                                   | 3            |
| 3.1.1 Acceptance of standard language elements .....                                    | 3            |
| 3.1.2 Interaction with non-COBOL runtime modules .....                                  | 3            |
| 3.1.3 Interaction between COBOL implementations .....                                   | 3            |
| 3.1.4 Implementor-defined language elements .....                                       | 3            |
| 3.1.5 Processor-dependent language elements .....                                       | 4            |
| 3.1.6 Reserved words .....                                                              | 4            |
| 3.1.7 Standard extensions .....                                                         | 4            |
| 3.1.8 Nonstandard extensions .....                                                      | 4            |
| 3.1.9 Substitute or additional language elements .....                                  | 5            |
| 3.1.10 Archaic language elements .....                                                  | 5            |
| 3.1.11 Obsolete language elements .....                                                 | 5            |
| 3.1.12 Externally-provided functionality .....                                          | 5            |
| 3.1.13 Limits .....                                                                     | 6            |
| 3.1.14 User documentation .....                                                         | 6            |
| 3.1.15 Character substitution .....                                                     | 6            |
| 3.2 A conforming compilation group .....                                                | 6            |
| 3.3 A conforming run unit .....                                                         | 6            |
| 3.4 Relationship of a conforming compilation group to a conforming implementation ..... | 7            |
| 3.5 Relationship of a conforming run unit to a conforming implementation .....          | 7            |
| <b>4 Terms and Definitions</b> .....                                                    | <b>8</b>     |
| <b>5 Description techniques</b> .....                                                   | <b>17</b>    |
| 5.1 General formats .....                                                               | 17           |
| 5.1.1 Keywords .....                                                                    | 17           |
| 5.1.2 Optional words .....                                                              | 17           |
| 5.1.3 Operands .....                                                                    | 17           |

|                                                         |           |
|---------------------------------------------------------|-----------|
| 5.1.4 Level numbers .....                               | 18        |
| 5.1.5 Options .....                                     | 18        |
| 5.1.5.1 Brackets .....                                  | 18        |
| 5.1.5.2 Braces .....                                    | 18        |
| 5.1.5.3 Choice indicators .....                         | 18        |
| 5.1.6 Ellipses .....                                    | 18        |
| 5.1.7 Punctuation .....                                 | 19        |
| 5.1.8 Special characters .....                          | 19        |
| 5.1.9 Meta-terms .....                                  | 19        |
| 5.2 Rules .....                                         | 19        |
| 5.2.1 Syntax rules .....                                | 19        |
| 5.2.2 General rules .....                               | 19        |
| 5.2.3 Argument rules .....                              | 19        |
| 5.2.4 Returned value rules .....                        | 19        |
| 5.3 Arithmetic expressions .....                        | 20        |
| 5.3.1 Textually subscripted operands .....              | 20        |
| 5.3.2 Ellipses .....                                    | 20        |
| 5.4 Integer operands .....                              | 20        |
| 5.5 Informal description .....                          | 20        |
| 5.6 Hyphens in text .....                               | 21        |
| 5.7 Verbal forms for the expression of provisions ..... | 21        |
| <b>6 Reference format .....</b>                         | <b>22</b> |
| 6.1 Indicators .....                                    | 22        |
| 6.1.1 Fixed indicators .....                            | 22        |
| 6.1.2 Floating indicators .....                         | 23        |
| 6.2 Fixed-form reference format .....                   | 24        |
| 6.3 Free-form reference format .....                    | 26        |
| 6.4 Logical conversion .....                            | 28        |
| <b>7 Compiler directing facility .....</b>              | <b>29</b> |
| 7.1 Text manipulation .....                             | 29        |
| 7.1.1 Text manipulation elements .....                  | 30        |
| 7.1.2 COPY statement .....                              | 32        |
| 7.1.3 REPLACE statement .....                           | 36        |
| 7.2 Compiler directives .....                           | 40        |
| 7.2.4 Conditional compilation .....                     | 41        |
| 7.2.5 Compile-time arithmetic expressions .....         | 41        |
| 7.2.6 Compile-time boolean expressions .....            | 41        |
| 7.2.7 Constant conditional expression .....             | 42        |
| 7.2.8 CALL-CONVENTION directive .....                   | 44        |
| 7.2.9 DEFINE directive .....                            | 45        |
| 7.2.10 EVALUATE directive .....                         | 46        |
| 7.2.11 FLAG-85 directive .....                          | 49        |
| 7.2.12 FLAG-NATIVE-ARITHMETIC directive .....           | 51        |
| 7.2.13 IF directive .....                               | 52        |

|                                                        |           |
|--------------------------------------------------------|-----------|
| 7.2.14 LEAP-SECOND directive .....                     | 53        |
| 7.2.15 LISTING directive .....                         | 54        |
| 7.2.16 PAGE directive .....                            | 55        |
| 7.2.17 PROPAGATE directive .....                       | 56        |
| 7.2.18 SOURCE FORMAT directive .....                   | 57        |
| 7.2.19 TURN directive .....                            | 58        |
| <b>8 Language fundamentals .....</b>                   | <b>59</b> |
| 8.1 Character sets .....                               | 59        |
| 8.1.1 Computer's coded character set .....             | 59        |
| 8.1.2 COBOL character repertoire .....                 | 60        |
| 8.1.3 Alphabets .....                                  | 62        |
| 8.1.4 Collating sequences .....                        | 63        |
| 8.2 Locales .....                                      | 64        |
| 8.2.1 Locale field names .....                         | 65        |
| 8.3 Lexical elements .....                             | 66        |
| 8.3.1 Character-strings .....                          | 66        |
| 8.3.1.1 COBOL words .....                              | 66        |
| 8.3.1.2 Literals .....                                 | 74        |
| 8.3.1.3 Picture character-strings .....                | 81        |
| 8.3.2 Separators .....                                 | 81        |
| 8.4 References .....                                   | 83        |
| 8.4.1 Uniqueness of reference .....                    | 83        |
| 8.4.1.1 Qualification .....                            | 83        |
| 8.4.1.2 Subscripts .....                               | 85        |
| 8.4.2 Identifiers .....                                | 87        |
| 8.4.2.1 Identifier .....                               | 87        |
| 8.4.2.2 Function-identifier .....                      | 89        |
| 8.4.2.3 Reference-modification .....                   | 92        |
| 8.4.2.4 Inline method invocation .....                 | 94        |
| 8.4.2.5 Object-view .....                              | 95        |
| 8.4.2.6 EXCEPTION-OBJECT .....                         | 95        |
| 8.4.2.7 NULL .....                                     | 96        |
| 8.4.2.8 SELF and SUPER .....                           | 96        |
| 8.4.2.9 Object property .....                          | 97        |
| 8.4.2.10 Predefined-address .....                      | 98        |
| 8.4.2.11 Data-address-identifier .....                 | 99        |
| 8.4.2.12 Program-address-identifier .....              | 99        |
| 8.4.2.13 LINAGE-COUNTER .....                          | 100       |
| 8.4.2.14 Report counters .....                         | 100       |
| 8.4.3 Condition-name .....                             | 101       |
| 8.4.4 Explicit and implicit references .....           | 102       |
| 8.4.5 Scope of names .....                             | 102       |
| 8.4.5.1 Local and global names .....                   | 103       |
| 8.4.5.2 Scope of program-names .....                   | 104       |
| 8.4.5.3 Scope of class-names and interface-names ..... | 105       |

|                                                                                     |     |
|-------------------------------------------------------------------------------------|-----|
| 8.4.5.4 Scope of method-names .....                                                 | 105 |
| 8.4.5.5 Scope of function-prototype-names .....                                     | 105 |
| 8.4.5.6 Scope of user-function-names .....                                          | 105 |
| 8.4.5.7 Scope of program-prototype-names .....                                      | 106 |
| 8.4.5.8 Scope of compilation-variable-names .....                                   | 106 |
| 8.4.5.9 Scope of parameter-names .....                                              | 106 |
| 8.4.5.10 Scope of property-names .....                                              | 106 |
| 8.5 Data description and representation .....                                       | 107 |
| 8.5.1 Computer independent data description .....                                   | 107 |
| 8.5.1.1 Files and records .....                                                     | 107 |
| 8.5.1.2 Levels .....                                                                | 107 |
| 8.5.1.3 Limitations of character handling .....                                     | 108 |
| 8.5.1.4 Algebraic signs .....                                                       | 108 |
| 8.5.1.5 Alignment of data items in storage .....                                    | 109 |
| 8.5.1.5.1 Alignment of alphanumeric groups and of data items of usage display ..... | 109 |
| 8.5.1.5.2 Alignment of data items of usage national .....                           | 109 |
| 8.5.1.5.3 Alignment of data items of usage bit .....                                | 109 |
| 8.5.1.5.4 Item alignment for increased object-code efficiency .....                 | 110 |
| 8.5.2 Class and category of data items and literals .....                           | 110 |
| 8.5.3 Types .....                                                                   | 113 |
| 8.5.3.1 Weakly-typed items .....                                                    | 114 |
| 8.5.3.2 Strongly-typed group items .....                                            | 114 |
| 8.5.4 Zero-length items .....                                                       | 114 |
| 8.6 Scope and life cycle of data .....                                              | 115 |
| 8.6.1 Global names and local names .....                                            | 115 |
| 8.6.2 External and internal items .....                                             | 115 |
| 8.6.3 Automatic, initial, and static internal items .....                           | 115 |
| 8.6.4 Based entries and based data items .....                                      | 116 |
| 8.6.5 Common, initial, and recursive attributes .....                               | 117 |
| 8.6.6 Sharing data items .....                                                      | 117 |
| 8.7 Operators .....                                                                 | 118 |
| 8.7.1 Arithmetic operators .....                                                    | 118 |
| 8.7.2 Boolean operators .....                                                       | 118 |
| 8.7.3 Concatenation operator .....                                                  | 118 |
| 8.7.4 Invocation operator .....                                                     | 118 |
| 8.7.5 Relational operators .....                                                    | 118 |
| 8.7.6 Logical operators .....                                                       | 119 |
| 8.8 Expressions .....                                                               | 120 |
| 8.8.1 Arithmetic expressions .....                                                  | 120 |
| 8.8.2 Boolean expressions .....                                                     | 124 |
| 8.8.3 Concatenation expressions .....                                               | 125 |
| 8.8.4 Conditional expressions .....                                                 | 126 |
| 8.9 Reserved words .....                                                            | 138 |
| 8.10 Context-sensitive words .....                                                  | 141 |
| 8.11 Intrinsic function names .....                                                 | 143 |
| 8.12 Compiler-directive words .....                                                 | 144 |

|                                                         |            |
|---------------------------------------------------------|------------|
| 8.13 External repository .....                          | 145        |
| <b>9 I-O, objects, and user-defined functions .....</b> | <b>146</b> |
| 9.1 Files .....                                         | 146        |
| 9.1.1 Physical and logical files .....                  | 146        |
| 9.1.2 Record area .....                                 | 146        |
| 9.1.3 File connector .....                              | 146        |
| 9.1.4 Open mode .....                                   | 147        |
| 9.1.5 Sharing file connectors .....                     | 147        |
| 9.1.6 Fixed file attributes .....                       | 147        |
| 9.1.7 Organization .....                                | 147        |
| 9.1.7.1 Sequential .....                                | 147        |
| 9.1.7.2 Relative .....                                  | 148        |
| 9.1.7.3 Indexed .....                                   | 148        |
| 9.1.8 Access modes .....                                | 148        |
| 9.1.8.1 Sequential access mode .....                    | 149        |
| 9.1.8.2 Random access mode .....                        | 149        |
| 9.1.8.3 Dynamic access mode .....                       | 149        |
| 9.1.9 Reel and unit .....                               | 149        |
| 9.1.10 Current volume pointer .....                     | 149        |
| 9.1.11 File position indicator .....                    | 149        |
| 9.1.12 I-O status .....                                 | 150        |
| 9.1.13 Invalid key condition .....                      | 154        |
| 9.1.14 Sharing mode .....                               | 155        |
| 9.1.15 Record locking .....                             | 156        |
| 9.1.16 Sort file .....                                  | 157        |
| 9.1.17 Merge file .....                                 | 157        |
| 9.1.18 Dynamic file assignment .....                    | 157        |
| 9.1.19 Report file .....                                | 157        |
| 9.2 Screens .....                                       | 158        |
| 9.2.1 Terminal screen .....                             | 158        |
| 9.2.2 Function keys .....                               | 158        |
| 9.2.3 CRT status .....                                  | 158        |
| 9.2.4 Cursor .....                                      | 159        |
| 9.2.5 Cursor locator .....                              | 159        |
| 9.2.6 Current screen item .....                         | 160        |
| 9.2.7 Color number .....                                | 160        |
| 9.3 Objects .....                                       | 161        |
| 9.3.1 Objects and classes .....                         | 161        |
| 9.3.2 Object references .....                           | 161        |
| 9.3.3 Predefined object references .....                | 161        |
| 9.3.4 Methods .....                                     | 161        |
| 9.3.5 Method invocation .....                           | 161        |
| 9.3.6 Method prototypes .....                           | 162        |
| 9.3.7 Conformance and interfaces .....                  | 162        |
| 9.3.8 Polymorphism .....                                | 165        |

|                                                  |            |
|--------------------------------------------------|------------|
| 9.3.9 Class inheritance .....                    | 165        |
| 9.3.10 Interface inheritance .....               | 165        |
| 9.3.11 Interface implementation .....            | 166        |
| 9.3.12 Parameterized classes .....               | 166        |
| 9.3.13 Parameterized interfaces .....            | 166        |
| 9.3.14 Object life cycle .....                   | 166        |
| 9.4 User-defined functions .....                 | 167        |
| <b>10 Structured compilation group .....</b>     | <b>168</b> |
| 10.1 Compilation units and runtime modules ..... | 168        |
| 10.2 Source units .....                          | 168        |
| 10.3 Contained source units .....                | 168        |
| 10.4 Source elements and runtime elements .....  | 169        |
| 10.5 COBOL compilation group .....               | 169        |
| 10.5.1 General format .....                      | 169        |
| 10.5.2 Syntax rules .....                        | 172        |
| 10.5.3 General rules .....                       | 173        |
| 10.6 End markers .....                           | 174        |
| <b>11 Identification division .....</b>          | <b>175</b> |
| 11.1 Identification division structure .....     | 175        |
| 11.2 CLASS-ID paragraph .....                    | 176        |
| 11.3 FACTORY paragraph .....                     | 178        |
| 11.4 FUNCTION-ID paragraph .....                 | 179        |
| 11.5 INTERFACE-ID paragraph .....                | 180        |
| 11.6 METHOD-ID paragraph .....                   | 181        |
| 11.7 OBJECT paragraph .....                      | 183        |
| 11.8 OPTIONS paragraph .....                     | 184        |
| 11.8.4 ARITHMETIC clause .....                   | 184        |
| 11.8.5 ENTRY-CONVENTION clause .....             | 185        |
| 11.9 PROGRAM-ID paragraph .....                  | 186        |
| <b>12 Environment division .....</b>             | <b>188</b> |
| 12.1 Environment division structure .....        | 188        |
| 12.2 Configuration section .....                 | 189        |
| 12.2.4 SOURCE-COMPUTER paragraph .....           | 190        |
| 12.2.5 OBJECT-COMPUTER paragraph .....           | 191        |
| 12.2.6 SPECIAL-NAMES paragraph .....             | 194        |
| 12.2.7 REPOSITORY paragraph .....                | 204        |
| 12.3 Input-output section .....                  | 209        |
| 12.3.3 FILE-CONTROL paragraph .....              | 210        |
| 12.3.4 File control entry .....                  | 210        |
| 12.3.4.4 ACCESS MODE clause .....                | 216        |
| 12.3.4.5 ALTERNATE RECORD KEY clause .....       | 217        |
| 12.3.4.6 COLLATING SEQUENCE clause .....         | 218        |
| 12.3.4.7 FILE STATUS clause .....                | 220        |

|                                                |            |
|------------------------------------------------|------------|
| 12.3.4.8 LOCK MODE clause .....                | 221        |
| 12.3.4.9 ORGANIZATION clause .....             | 223        |
| 12.3.4.10 PADDING CHARACTER clause .....       | 224        |
| 12.3.4.11 RECORD DELIMITER clause .....        | 225        |
| 12.3.4.12 RECORD KEY clause .....              | 226        |
| 12.3.4.13 RELATIVE KEY clause .....            | 227        |
| 12.3.4.14 RESERVE clause .....                 | 228        |
| 12.3.4.15 SHARING clause .....                 | 229        |
| 12.3.5 I-O-CONTROL paragraph .....             | 230        |
| 12.3.6 SAME clause .....                       | 230        |
| <b>13 Data division .....</b>                  | <b>232</b> |
| 13.1 Data division structure .....             | 232        |
| 13.2 Explicit and implicit attributes .....    | 232        |
| 13.3 File section .....                        | 233        |
| 13.3.4 File description entry .....            | 234        |
| 13.3.5 Sort-merge file description entry ..... | 237        |
| 13.4 Working-storage section .....             | 238        |
| 13.5 Local-storage section .....               | 239        |
| 13.6 Linkage section .....                     | 240        |
| 13.7 Report section .....                      | 242        |
| 13.7.3 Report description entry .....          | 242        |
| 13.7.4 Report group description entry .....    | 242        |
| 13.7.5 Report subdivisions .....               | 242        |
| 13.8 Screen section .....                      | 244        |
| 13.9 Constant entry .....                      | 245        |
| 13.10 Record description entry .....           | 247        |
| 13.11 77-level data description entry .....    | 248        |
| 13.12 Report description entry .....           | 248        |
| 13.13 Report group description entry .....     | 249        |
| 13.14 Data description entry .....             | 251        |
| 13.15 Screen description entry .....           | 255        |
| 13.16 Data division clauses .....              | 259        |
| 13.16.1 ALIGNED clause .....                   | 259        |
| 13.16.2 ANY LENGTH clause .....                | 260        |
| 13.16.3 AUTO clause .....                      | 261        |
| 13.16.4 BACKGROUND-COLOR clause .....          | 262        |
| 13.16.5 BASED clause .....                     | 263        |
| 13.16.6 BELL clause .....                      | 264        |
| 13.16.7 BLANK clause .....                     | 265        |
| 13.16.8 BLANK WHEN ZERO clause .....           | 266        |
| 13.16.9 BLINK clause .....                     | 267        |
| 13.16.10 BLOCK CONTAINS clause .....           | 268        |
| 13.16.11 CLASS clause .....                    | 269        |
| 13.16.12 CODE clause .....                     | 270        |
| 13.16.13 CODE-SET clause .....                 | 271        |

|                                  |     |
|----------------------------------|-----|
| 13.16.14 COLUMN clause           | 273 |
| 13.16.15 CONTROL clause          | 277 |
| 13.16.16 DEFAULT clause          | 279 |
| 13.16.17 DESTINATION clause      | 280 |
| 13.16.18 Entry-name clause       | 281 |
| 13.16.19 ERASE clause            | 282 |
| 13.16.20 EXTERNAL clause         | 283 |
| 13.16.21 FOREGROUND-COLOR clause | 284 |
| 13.16.22 FORMAT clause           | 285 |
| 13.16.23 FROM clause             | 288 |
| 13.16.24 FULL clause             | 289 |
| 13.16.25 GLOBAL clause           | 290 |
| 13.16.26 GROUP INDICATE clause   | 291 |
| 13.16.27 GROUP-USAGE clause      | 292 |
| 13.16.28 HIGHLIGHT clause        | 293 |
| 13.16.29 INVALID clause          | 294 |
| 13.16.30 JUSTIFIED clause        | 295 |
| 13.16.31 Level-number            | 296 |
| 13.16.32 LINAGE clause           | 297 |
| 13.16.33 LINE clause             | 299 |
| 13.16.34 LOWLIGHT clause         | 303 |
| 13.16.35 NEXT GROUP clause       | 304 |
| 13.16.36 OCCURS clause           | 306 |
| 13.16.37 PAGE clause             | 311 |
| 13.16.38 PICTURE clause          | 313 |
| 13.16.39 PRESENT WHEN clause     | 328 |
| 13.16.40 PROPERTY clause         | 330 |
| 13.16.41 RECORD clause           | 333 |
| 13.16.42 REDEFINES clause        | 336 |
| 13.16.43 RENAMES clause          | 338 |
| 13.16.44 REPORT clause           | 339 |
| 13.16.45 REQUIRED clause         | 340 |
| 13.16.46 REVERSE-VIDEO clause    | 341 |
| 13.16.47 SAME AS clause          | 342 |
| 13.16.48 SECURE clause           | 344 |
| 13.16.49 SELECT WHEN clause      | 345 |
| 13.16.50 SIGN clause             | 346 |
| 13.16.51 SOURCE clause           | 347 |
| 13.16.52 SUM clause              | 348 |
| 13.16.53 SYNCHRONIZED clause     | 351 |
| 13.16.54 TO clause               | 353 |
| 13.16.55 TYPE clause             | 354 |
| 13.16.56 TYPEDEF clause          | 359 |
| 13.16.57 UNDERLINE clause        | 360 |
| 13.16.58 USAGE clause            | 361 |
| 13.16.59 USING clause            | 366 |

|           |                                                 |            |
|-----------|-------------------------------------------------|------------|
| 13.16.60  | VALIDATE-STATUS clause                          | 367        |
| 13.16.61  | VALUE clause                                    | 369        |
| 13.16.62  | VARYING clause                                  | 375        |
| <b>14</b> | <b>Procedure division</b>                       | <b>377</b> |
| 14.1      | Procedure division structure                    | 377        |
| 14.2      | Declaratives                                    | 381        |
| 14.3      | Procedures                                      | 381        |
| 14.3.1    | Sections                                        | 381        |
| 14.3.2    | Paragraphs                                      | 381        |
| 14.4      | Procedural statements and sentences             | 382        |
| 14.4.1    | Conditional phrase                              | 384        |
| 14.4.2    | Scope of statements                             | 384        |
| 14.5      | Execution                                       | 385        |
| 14.5.1    | Run unit organization                           | 385        |
| 14.5.2    | State of a function, method, object, or program | 385        |
| 14.5.3    | Explicit and implicit transfers of control      | 387        |
| 14.5.4    | Item identification                             | 388        |
| 14.5.5    | Results of runtime element execution            | 388        |
| 14.5.6    | Locale identification                           | 388        |
| 14.5.7    | Sending and receiving operands                  | 389        |
| 14.5.8    | Alignment of data within data items             | 389        |
| 14.5.9    | Overlapping operands                            | 390        |
| 14.5.10   | Normal run unit termination                     | 390        |
| 14.5.11   | Abnormal run unit termination                   | 391        |
| 14.5.12   | Condition handling                              | 391        |
| 14.6      | Common phrases and features for statements      | 399        |
| 14.6.1    | At end condition                                | 399        |
| 14.6.2    | Invalid key condition                           | 399        |
| 14.6.3    | ROUNDED phrase                                  | 399        |
| 14.6.4    | SIZE ERROR phrase and size error condition      | 399        |
| 14.6.5    | CORRESPONDING phrase                            | 401        |
| 14.6.6    | Arithmetic statements                           | 401        |
| 14.6.7    | THROUGH phrase                                  | 402        |
| 14.6.8    | RETRY phrase                                    | 403        |
| 14.7      | Conformance for parameters and returning items  | 404        |
| 14.7.1    | Parameters                                      | 404        |
| 14.7.2    | Returning items                                 | 407        |
| 14.8      | Statements                                      | 409        |
| 14.8.1    | ACCEPT statement                                | 409        |
| 14.8.2    | ADD statement                                   | 415        |
| 14.8.3    | ALLOCATE statement                              | 418        |
| 14.8.4    | CALL statement                                  | 420        |
| 14.8.5    | CANCEL statement                                | 426        |
| 14.8.6    | CLOSE statement                                 | 428        |
| 14.8.7    | COMPUTE statement                               | 431        |

|           |                               |            |
|-----------|-------------------------------|------------|
| 14.8.8    | CONTINUE statement            | 432        |
| 14.8.9    | DELETE statement              | 433        |
| 14.8.10   | DISPLAY statement             | 435        |
| 14.8.11   | DIVIDE statement              | 438        |
| 14.8.12   | EVALUATE statement            | 441        |
| 14.8.13   | EXIT statement                | 446        |
| 14.8.14   | FREE statement                | 450        |
| 14.8.15   | GENERATE statement            | 451        |
| 14.8.16   | GO TO statement               | 453        |
| 14.8.17   | GOBACK statement              | 454        |
| 14.8.18   | IF statement                  | 455        |
| 14.8.19   | INITIALIZE statement          | 457        |
| 14.8.20   | INITIATE statement            | 461        |
| 14.8.21   | INSPECT statement             | 462        |
| 14.8.22   | INVOKE statement              | 468        |
| 14.8.23   | MERGE statement               | 472        |
| 14.8.24   | MOVE statement                | 477        |
| 14.8.25   | MULTIPLY statement            | 482        |
| 14.8.26   | OPEN statement                | 484        |
| 14.8.27   | PERFORM statement             | 489        |
| 14.8.28   | RAISE statement               | 495        |
| 14.8.29   | READ statement                | 496        |
| 14.8.30   | RELEASE statement             | 503        |
| 14.8.31   | RESUME statement              | 504        |
| 14.8.32   | RETURN statement              | 505        |
| 14.8.33   | REWRITE statement             | 507        |
| 14.8.34   | SEARCH statement              | 512        |
| 14.8.35   | SET statement                 | 516        |
| 14.8.36   | SORT statement                | 524        |
| 14.8.37   | START statement               | 531        |
| 14.8.38   | STOP statement                | 535        |
| 14.8.39   | STRING statement              | 536        |
| 14.8.40   | SUBTRACT statement            | 539        |
| 14.8.41   | SUPPRESS statement            | 542        |
| 14.8.42   | TERMINATE statement           | 543        |
| 14.8.43   | UNLOCK statement              | 544        |
| 14.8.44   | UNSTRING statement            | 545        |
| 14.8.45   | USE statement                 | 549        |
| 14.8.46   | VALIDATE statement            | 554        |
| 14.8.47   | WRITE statement               | 558        |
| <b>15</b> | <b>Intrinsic functions</b>    | <b>566</b> |
| 15.1      | Types of functions            | 566        |
| 15.2      | Arguments                     | 566        |
| 15.3      | Returned values               | 568        |
| 15.3.1    | Numeric and integer functions | 568        |

|       |                               |     |
|-------|-------------------------------|-----|
| 15.4  | Date conversion functions     | 569 |
| 15.5  | Summary of functions          | 569 |
| 15.6  | ABS function                  | 575 |
| 15.7  | ACOS function                 | 576 |
| 15.8  | ANNUITY function              | 577 |
| 15.9  | ASIN function                 | 578 |
| 15.10 | ATAN function                 | 579 |
| 15.11 | BOOLEAN-OF-INTEGER function   | 580 |
| 15.12 | BYTE-LENGTH function          | 581 |
| 15.13 | CHAR function                 | 582 |
| 15.14 | CHAR-NATIONAL function        | 583 |
| 15.15 | COS function                  | 584 |
| 15.16 | CURRENT-DATE function         | 585 |
| 15.17 | DATE-OF-INTEGER function      | 586 |
| 15.18 | DATE-TO-YYYYMMDD function     | 587 |
| 15.19 | DAY-OF-INTEGER function       | 588 |
| 15.20 | DAY-TO-YYYYDDD function       | 589 |
| 15.21 | DISPLAY-OF function           | 590 |
| 15.22 | E function                    | 591 |
| 15.23 | EXCEPTION-FILE function       | 592 |
| 15.24 | EXCEPTION-FILE-N function     | 593 |
| 15.25 | EXCEPTION-LOCATION function   | 594 |
| 15.26 | EXCEPTION-LOCATION-N function | 595 |
| 15.27 | EXCEPTION-STATEMENT function  | 596 |
| 15.28 | EXCEPTION-STATUS function     | 597 |
| 15.29 | EXP function                  | 598 |
| 15.30 | EXP10 function                | 599 |
| 15.31 | FACTORIAL function            | 600 |
| 15.32 | FRACTION-PART function        | 601 |
| 15.33 | HIGHEST-ALGEBRAIC function    | 602 |
| 15.34 | INTEGER function              | 603 |
| 15.35 | INTEGER-OF-BOOLEAN function   | 604 |
| 15.36 | INTEGER-OF-DATE function      | 605 |
| 15.37 | INTEGER-OF-DAY function       | 606 |
| 15.38 | INTEGER-PART function         | 607 |
| 15.39 | LENGTH function               | 608 |
| 15.40 | LOCALE-COMPARE function       | 609 |
| 15.41 | LOCALE-DATE function          | 610 |
| 15.42 | LOCALE-TIME function          | 611 |
| 15.43 | LOG function                  | 612 |
| 15.44 | LOG10 function                | 613 |
| 15.45 | LOWER-CASE function           | 614 |
| 15.46 | LOWEST-ALGEBRAIC function     | 615 |
| 15.47 | MAX function                  | 616 |
| 15.48 | MEAN function                 | 617 |
| 15.49 | MEDIAN function               | 618 |

|           |                                |            |
|-----------|--------------------------------|------------|
| 15.50     | MIDRANGE function              | 619        |
| 15.51     | MIN function                   | 620        |
| 15.52     | MOD function                   | 621        |
| 15.53     | NATIONAL-OF function           | 622        |
| 15.54     | NUMVAL function                | 623        |
| 15.55     | NUMVAL-C function              | 624        |
| 15.56     | NUMVAL-F function              | 626        |
| 15.57     | ORD function                   | 627        |
| 15.58     | ORD-MAX function               | 628        |
| 15.59     | ORD-MIN function               | 629        |
| 15.60     | PI function                    | 630        |
| 15.61     | PRESENT-VALUE function         | 631        |
| 15.62     | RANDOM function                | 632        |
| 15.63     | RANGE function                 | 633        |
| 15.64     | REM function                   | 634        |
| 15.65     | REVERSE function               | 635        |
| 15.66     | SIGN function                  | 636        |
| 15.67     | SIN function                   | 637        |
| 15.68     | SQRT function                  | 638        |
| 15.69     | STANDARD-COMPARE function      | 639        |
| 15.70     | STANDARD-DEVIATION function    | 640        |
| 15.71     | SUM function                   | 641        |
| 15.72     | TAN function                   | 642        |
| 15.73     | TEST-DATE-YYYYMMDD function    | 643        |
| 15.74     | TEST-DAY-YYYYDDD function      | 644        |
| 15.75     | TEST-NUMVAL function           | 645        |
| 15.76     | TEST-NUMVAL-C function         | 646        |
| 15.77     | TEST-NUMVAL-F function         | 647        |
| 15.78     | UPPER-CASE function            | 648        |
| 15.79     | VARIANCE function              | 649        |
| 15.80     | WHEN-COMPILED function         | 650        |
| 15.81     | YEAR-TO-YYYY function          | 651        |
| <b>16</b> | <b>Standard classes</b>        | <b>652</b> |
| 16.1      | BASE class                     | 652        |
| 16.1.1    | New method                     | 652        |
| 16.1.2    | FactoryObject method           | 652        |
| <b>A</b>  | <b>Communications facility</b> | <b>654</b> |
| A.1       | Data division                  | 655        |
| A.2       | Procedure division             | 667        |
| A.2.1     | ACCEPT MESSAGE COUNT statement | 667        |
| A.2.2     | DISABLE statement              | 668        |
| A.2.3     | ENABLE statement               | 669        |
| A.2.4     | PURGE statement                | 670        |
| A.2.5     | RECEIVE statement              | 671        |

|                                                                                     |            |
|-------------------------------------------------------------------------------------|------------|
| A.2.6 SEND statement .....                                                          | 673        |
| <b>B Language element lists .....</b>                                               | <b>677</b> |
| B.1 Implementor-defined language element list .....                                 | 677        |
| B.2 Undefined language element list .....                                           | 690        |
| B.3 Processor-dependent language element list .....                                 | 696        |
| <b>C Characters permitted in user-defined words .....</b>                           | <b>698</b> |
| C.1 Notation .....                                                                  | 698        |
| C.2 Repertoire of characters permitted in user-defined words .....                  | 698        |
| <b>D Mapping of uppercase letters to lowercase letters .....</b>                    | <b>701</b> |
| D.1 Notations .....                                                                 | 701        |
| D.2 Case mapping list .....                                                         | 701        |
| <b>E Concepts .....</b>                                                             | <b>705</b> |
| E.1 Files .....                                                                     | 705        |
| E.2 Table handling .....                                                            | 714        |
| E.3 Shared memory area .....                                                        | 720        |
| E.4 Compilation group and run unit organization and communication .....             | 721        |
| E.5 Communications facility .....                                                   | 734        |
| E.6 Intrinsic function facility .....                                               | 740        |
| E.7 Debugging .....                                                                 | 742        |
| E.8 Types .....                                                                     | 742        |
| E.9 Addresses and pointers .....                                                    | 745        |
| E.10 Boolean support and bit manipulation .....                                     | 746        |
| E.11 Character sets .....                                                           | 750        |
| E.12 Collating sequences .....                                                      | 752        |
| E.13 Culturally-specific, culturally-adaptable, and multilingual applications ..... | 756        |
| E.14 External switches .....                                                        | 761        |
| E.15 Common exception processing .....                                              | 761        |
| E.16 Standard arithmetic .....                                                      | 763        |
| E.17 Object oriented concepts .....                                                 | 768        |
| E.18 Report writer .....                                                            | 785        |
| E.19 Validate facility .....                                                        | 792        |
| E.20 Conditional expressions .....                                                  | 796        |
| E.21 Examples of the execution of the INSPECT statement .....                       | 800        |
| E.22 Examples of the execution of the PERFORM statement .....                       | 803        |
| E.23 Example of free-form reference format .....                                    | 807        |
| E.24 Conditional compilation .....                                                  | 808        |
| E.25 CALL-CONVENTION directive .....                                                | 809        |
| E.26 ENTRY-CONVENTION clause .....                                                  | 809        |
| <b>F Substantive changes list .....</b>                                             | <b>810</b> |
| F.1 Substantive changes potentially affecting existing programs .....               | 810        |
| F.2 Substantive changes not affecting existing programs .....                       | 823        |

**G Archaic and obsolete language element lists ..... 833**  
G.1 Archaic language elements ..... 833  
G.2 Obsolete language elements ..... 834

**H Known errors in the standard ..... 835**  
H.1 Rationale ..... 835  
H.2 List of errors ..... 835

**I Bibliography ..... 836**

**Index ..... 837**

## Tables

|                                                                                             |     |
|---------------------------------------------------------------------------------------------|-----|
| 1 Verbal forms .....                                                                        | 21  |
| 2 COBOL character repertoire .....                                                          | 61  |
| 3 Class and category relationships for elementary data items .....                          | 111 |
| 4 Combinations of symbols in arithmetic expressions .....                                   | 120 |
| 5 Combination of symbols in boolean expressions .....                                       | 125 |
| 6 Combinations of conditions, logical operators, and parentheses .....                      | 136 |
| 7 Relationship of alphabet-name to coded character set and collating sequence .....         | 199 |
| 8 Category and type of editing .....                                                        | 321 |
| 9 Results of fixed insertion editing .....                                                  | 322 |
| 10 Results of floating insertion editing .....                                              | 323 |
| 11 Format 1 picture symbol order of precedence .....                                        | 326 |
| 12 Format 2 picture symbol order of precedence .....                                        | 327 |
| 13 Procedural statements .....                                                              | 382 |
| 14 Exception-names and exception conditions .....                                           | 394 |
| 15 Relationship of categories of physical files and the format of the CLOSE statement ..... | 428 |
| 16 Combination of operands in the EVALUATE statement .....                                  | 443 |
| 17 Validity of types of MOVE statements .....                                               | 478 |
| 18 Category of figurative constants used in the MOVE statement .....                        | 481 |
| 19 Opening available and unavailable files (file not currently open) .....                  | 485 |
| 20 Opening available shared files that are currently open by another file connector .....   | 485 |
| 21 Permissible I-O statements by access mode and open mode .....                            | 486 |
| 22 Table of functions .....                                                                 | 570 |
| A.1 Communication status key conditions .....                                               | 665 |
| A.2 Error key values .....                                                                  | 666 |
| A.3 SEND statement indicators .....                                                         | 675 |
| E.1 Summary of record lock acquisition and release .....                                    | 713 |
| E.2 Examples of boolean operations .....                                                    | 747 |

## Figures

|                                                                                                      |     |
|------------------------------------------------------------------------------------------------------|-----|
| 1 Fixed-form reference format .....                                                                  | 24  |
| E.1 Format 1 SEARCH statement having two WHEN phrases .....                                          | 718 |
| E.2 Compilation group sample structure example .....                                                 | 722 |
| E.3 Compilation group and run unit structures .....                                                  | 723 |
| E.4 COBOL communication environment .....                                                            | 736 |
| E.5 Hierarchy of queues .....                                                                        | 739 |
| E.6 Manager class .....                                                                              | 773 |
| E.7 Banking hierarchy .....                                                                          | 774 |
| E.8 Example of page layout .....                                                                     | 786 |
| E.9 Evaluation of the condition-1 AND condition-2 AND ... condition-n .....                          | 796 |
| E.10 Evaluation of the condition-1 OR condition-2 OR ... condition-n .....                           | 797 |
| E.11 Evaluation of condition-1 OR condition-2 AND condition-3 .....                                  | 798 |
| E.12 Evaluation of (condition-1 OR NOT condition-2) AND condition-3 AND condition-4 .....            | 799 |
| E.13 The VARYING option of a PERFORM statement with the TEST BEFORE phrase having one condition ...  | 803 |
| E.14 The VARYING option of a PERFORM statement with the TEST BEFORE phrase having two conditions ..  | 804 |
| E.15 The VARYING option of a PERFORM statement with the TEST AFTER phrase having one condition ....  | 805 |
| E.16 The VARYING option of a PERFORM statement with the TEST AFTER phrase having two conditions .... | 806 |

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International standards are drafted in accordance with the rules given in the ISO/IEC Directives, part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 1989 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This first edition of ISO/IEC 1989 cancels and replaces ISO 1989:1985, which has been technically revised. It incorporates Amd.1:1992 and Amd.2:1994. This edition introduces the following significant technical enhancements:

- features for object-oriented programming
- additional features for detection and reporting of exceptions
- a boolean data type for bit handling and boolean operations
- native binary and floating-point data types
- a national character data type for processing multiple-octet coded character sets
- cultural adaptability, multilingual features, and tailoring for a given local language or culture
- increased portability of arithmetic
- free-form source and library text
- compiler directives for portable specification of processing options
- conditional compilation
- an enhanced report writer
- features for data validation
- several enhancements to the CALL statement, including recursion
- improved interoperability with other programming languages
- user-defined functions
- a screen handling facility
- file sharing and record locking
- support for ISO/IEC 10646-1 and ISO/IEC 10646-2 for data interchange

A complete list of technical changes is given in Annex F.

Annexes A through D form a normative part of this International Standard. Annexes E through H are for information only.

## Introduction

COBOL began as a business programming language, but its present use has spread well beyond that to a general-purpose programming language. COBOL is well known for its file handling capabilities, which are extended in this revision by the addition of file sharing and record locking capabilities. Other major enhancements add object-oriented capabilities, handling of national characters, and enhanced interoperability with other programming languages. Annex E, Concepts, includes an explanation of the major new features and is the suggested starting point for the reading of this document.

The previous revision of the COBOL standard was published in 1985 and extended by an amendment that added the Intrinsic Functions module in 1989 and a correction amendment in 1993. Implementors have provided language extensions in response to the demands of their users. Several changes and extensions have, therefore, been made to this revision in order to prevent further divergence, and to ensure consistency and coherence.

Development of the COBOL language began before the invention of formal techniques for specification of programming languages. Hence, the COBOL standard uses its own description techniques, which are described in 5, Description techniques. These techniques involve general formats, which describe the syntax, and natural language.

This revision is a result of the standardization efforts of working group ISO/IEC JTC 1/SC22/WG4 and technical committee J4, a subgroup of Accredited Standards Committee INCITS. During the development of this revision, great care was taken to minimize changes that would impact existing programs. Most substantive changes that potentially impact existing programs were introduced to resolve ambiguities in the previous revision. Details of the substantive changes are given in Annex F, Substantive changes list.

# Information technology — Programming languages — COBOL

## 1 Scope

This International Standard specifies the syntax and semantics of COBOL. Its purpose is to promote a high degree of machine independence to permit the use of COBOL on a variety of data processing systems.

This International Standard specifies:

- The form of a compilation group written in COBOL.
- The effect of compiling a compilation group.
- The effect of executing run units.
- The elements of the language for which a conforming implementation is required to supply a definition.
- The elements of the language for which meaning is explicitly undefined.
- The elements of the language that are dependent on the capabilities of the processor.

This International Standard does not specify:

- The means whereby a compilation group written in COBOL is compiled into code executable by a processor.
- The time at which method, function, or program runtime modules are linked or bound to an activating statement, except that runtime binding occurs of necessity when the identification of the appropriate program or method is not known at compile time.
- The time at which parameterized classes are expanded.
- The mechanism by which locales are defined and made available on a processor.
- The form or content of error, flagging, or warning messages.
- The form and content of listings produced during compilation, if any.
- The form of documentation produced by an implementor of products conforming to this International Standard.
- The sharing of resources other than files among run units.

## **2 Normative references**

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 646, *Information technology — ISO 7-bit coded character set for information interchange*

ISO 1001:1986, *Information processing — File structure and labelling of magnetic tapes for information interchange*

ISO 8601:2000, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 9945-2:1993, *Information technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities*

ISO/IEC TR 10176:2001, *Information technology — Guidelines for the preparation of programming language standards*

ISO/IEC 10646-1, *Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane*

ISO/IEC 10646-2, *Information technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 2: Supplementary Planes*

ISO/IEC 14651:2001, *Information technology — International string ordering and comparison — Method for comparing character strings and description of the common template tailorable ordering (including AMD1)*

ISO/IEC TR 14652:—<sup>1)</sup>, *Information technology — Specification method for cultural conventions*

---

1) To be published.

### 3 Conformance to this International Standard

Clause 3 specifies requirements that an implementation shall fulfill in order to conform to this International Standard and defines the conditions under which a compilation group or run unit conforms in its use of standard features.

#### 3.1 A conforming implementation

To conform to this International Standard, an implementation of standard COBOL shall provide the normative elements specified in 6, Reference format through 16, Standard classes, and meet the criteria of 3.1.1 through 3.1.15.

An implementation may optionally provide the elements specified in Annex A, Communications facility.

##### 3.1.1 Acceptance of standard language elements

An implementation shall accept the syntax and provide the functionality for all standard language elements required by this International Standard and the optional or processor-dependent language elements for which support is claimed.

An implementation shall provide a warning mechanism that optionally can be invoked by the user at compile time to indicate violations of the general formats and the explicit syntax rules of standard COBOL. This warning mechanism shall provide a suboption for selection or suppression of checking for violations of the set of conformance rules specified in 14.7, Conformance for parameters and returning items, and in 9.3.7.1.2, Conformance between interfaces.

There are rules in standard COBOL that are not identified as general formats or syntax rules, but nevertheless specify elements that are syntactically distinguishable. This warning mechanism shall indicate violations of such rules. For elements not specified in general formats or in explicit syntax rules, it is left to the implementor's judgement to determine what is syntactically distinguishable.

There are general rules in standard COBOL that could have been classified as syntax rules. These rules are classified as general rules for the purpose of avoiding syntax checking, and do not reflect errors in standard COBOL. An implementation may, but is not required to, flag violations of such rules.

##### 3.1.2 Interaction with non-COBOL runtime modules

Facilities are provided in this specification that enable transfer of control and sharing of external items between COBOL runtime modules and non-COBOL runtime modules. No requirement is placed on an implementation to support this interaction. When supported, an implementor shall document the languages and the implementations supported.

##### 3.1.3 Interaction between COBOL implementations

Facilities are provided in this specification that enhance the capability of transferring control and sharing external items between COBOL runtime elements translated on COBOL implementations produced by different implementors. No requirement is placed on an implementation to support this interaction. When supported, an implementor shall document the implementations supported.

##### 3.1.4 Implementor-defined language elements

Language elements that depend on implementor definition to complete the specification of the syntax rules or general rules are listed in B.1, Implementor-defined language element list. To meet the requirements of standard COBOL, the implementor shall specify, at a minimum, the implementor-defined language elements that are identified as required. Each implementor-defined language element specified by the implementor shall be documented if the implementor-defined language element is identified as requiring user documentation.

An implementor shall not require the inclusion of non-standard language elements in a compilation group as part of the definition of an implementor-defined language element.

### **3.1.5 Processor-dependent language elements**

Processor refers to the entire computing system that is used to translate compilation groups and execute run units, consisting of both hardware and relevant software. Language elements that depend on specific devices or on a specific processor capability, functionality, or architecture are listed in B.3, Processor-dependent language element list. To meet the requirements of standard COBOL, the implementor shall document the processor-dependent language elements for which the implementation claims support. Language elements that pertain to specific processor-dependent elements for which support is not claimed need not be implemented. The decision of whether to claim support for a processor-dependent language element is within an implementor's discretion. Factors that may be considered include, but are not limited to, hardware capability, software capability, and market positioning of the processor.

When standard-conforming support is claimed for a specific processor-dependent language element, all associated syntax and functionality required for that language element shall be implemented; when a subset of the syntax or functionality is implemented, that subset shall be identified as a standard extension in the implementor's user documentation. The absence of processor-dependent elements from an implementation shall be specified in the implementor's user documentation.

An implementation shall provide a warning mechanism at compile time to indicate use of syntactically-detectable processor-dependent language elements not supported by that implementation. Although this warning mechanism is required to identify processor-dependent elements that are not supported, it is not required to diagnose syntax errors within this unsupported syntax. The implementor is not required to produce executable code when unsupported processor-dependent language elements are used.

### **3.1.6 Reserved words**

An implementation shall recognize as reserved words all the COBOL reserved words specified in 8.9, Reserved words; shall recognize in context all the context-sensitive words specified in 8.10, Context-sensitive words; and shall recognize in compiler directives all the compiler-directive reserved words specified in 8.12, Compiler-directive words.

### **3.1.7 Standard extensions**

An implementor may claim support for all or a subset of the syntax and associated functionality of optional or processor-dependent elements. When an implementor claims support for a subset of the syntax, that syntax is a 'standard extension', provided that the associated functionality is that specified in this International Standard. If different functionality is provided, that syntax is a nonstandard extension.

### **3.1.8 Nonstandard extensions**

Nonstandard extensions are language elements or functionality in an implementation that consist of any of the following:

- 1) documented language elements not defined in this International Standard;
- 2) language elements defined in this International Standard for which different functionality is implemented, where that language element is not required for conformance to this International Standard, and standard support for that element is not claimed by the implementor;
- 3) language elements defined in this International Standard for which different functionality is implemented, where that language element is required for conformance to this International Standard, provided that standard-conforming behavior is also implemented and that an implementor-defined mechanism exists for selection of the nonstandard behavior.

An implementation that introduces additional reserved words as nonstandard extensions conforms to this International Standard, even though the additional reserved words may prevent translation of some conforming compilation groups.

Documentation associated with an implementation shall identify nonstandard extensions for which support is claimed and shall specify any reserved words added for nonstandard extensions.

An implementation shall provide a warning mechanism that optionally can be invoked by the user at compile time to indicate use of a nonstandard extension in a compilation group. This warning mechanism is required to flag only extensions that are syntactically distinguishable.

### **3.1.9 Substitute or additional language elements**

An implementation shall not require the inclusion of substitute or additional language elements in the compilation group in order to accomplish functionality specified for a standard language element.

#### **3.1.10 Archaic language elements**

Archaic language elements are those identified in G.1, Archaic language elements. Archaic language elements should not be used in new compilation groups because better programming practices exist. Because of widespread use, there is no schedule for deleting archaic elements from standard COBOL; however, this may be reevaluated for any future revisions of standard COBOL.

An implementation is required to support archaic language elements of the facilities for which support is claimed. Documentation associated with an implementation shall identify archaic language elements in the implementation.

An implementation shall provide a warning mechanism that optionally can be invoked by the user at compile time to indicate use of an archaic language element in a compilation group.

#### **3.1.11 Obsolete language elements**

Obsolete language elements are identified in G.2, Obsolete language elements. Obsolete language elements will be removed from the next revision of standard COBOL.

No language elements shall be deleted from the next revision of standard COBOL without having first been identified as obsolete in this International Standard.

An implementation is required to support obsolete language elements of the facilities for which support is claimed. Documentation associated with an implementation shall identify all obsolete language elements in the implementation.

An implementation shall provide a warning mechanism that optionally can be invoked by the user at compile time to indicate use of an obsolete element in a compilation group.

#### **3.1.12 Externally-provided functionality**

An implementation may require specifications outside the compilation group to interface with the operating environment to support functionality specified in a compilation group.

An implementation may require the presence in the operating environment of runtime modules or products in addition to the COBOL implementation to support syntax or functionality specified in a compilation group.

NOTE This permits an implementation to require components outside the COBOL implementation, such as pre-compilers, file systems, and sort products.

**3.1.13 Limits**

In general, standard COBOL specifies no upper limit on such things as the number of statements in a compilation group or the number of operands permitted in certain statements. A conforming implementation may place such limits. It is recognized that these limits will vary from one implementation of standard COBOL to another and may prevent the successful translation by a conforming implementation of some compilation groups that meet the requirements of standard COBOL.

**3.1.14 User documentation**

An implementation shall satisfy the user documentation requirements specified in 3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.8, 3.1.10, and 3.1.11 by specification in at least one form of documentation. This may include, but is not limited to, hardcopy manuals, on-line documentation, and user help screens.

Documentation requirements may be met by reference to other documents, including those of the operating environment and other COBOL implementations.

**3.1.15 Character substitution**

The definition of the COBOL character repertoire in 8.1.2, COBOL character repertoire, presents the complete COBOL character repertoire for standard COBOL. When an implementation does not provide a graphic representation for all the basic characters of the COBOL character repertoire, substitute graphics may be specified by the implementor to replace the characters not represented.

**3.2 A conforming compilation group**

A conforming compilation group is one that does not violate the explicitly stated syntactic provisions and specifications of standard COBOL. In order for a compilation group to conform to standard COBOL, it shall not include any language elements not specified in this International Standard. A compilation group that uses elements that are optional, processor-dependent, or implementor-defined in this International Standard is a conforming compilation group, even on implementations where it does not compile successfully due to the use of those elements.

The compilation units contained in a conforming compilation group are conforming compilation units

**3.3 A conforming run unit**

A conforming run unit is one that:

- 1) is composed of one or more runtime modules, each resulting from a successful compilation of a conforming compilation unit, and
- 2) complies with the explicitly stated provisions and specifications of standard COBOL with respect to the runtime behavior of that run unit.

NOTE The inclusion of non-COBOL components in the run unit does not affect the conformance of the run unit.

The processing of a conforming run unit is predictable only to the extent defined in standard COBOL. The results of violating the formats or rules of standard COBOL are undefined unless otherwise specified in this International Standard.

Situations in which the results of executing a statement are explicitly undefined or unpredictable are identified in B.2, Undefined language element list. A COBOL run unit that allows these situations to happen is a conforming run unit, although the resultant execution is not defined by standard COBOL.

### 3.4 Relationship of a conforming compilation group to a conforming implementation

The translation of a conforming compilation group by a conforming implementation is defined only to the extent specified in standard COBOL. It is possible that a conforming compilation group will not be translated successfully. Translation may be unsuccessful due to factors other than lack of conformance of a compilation group.

NOTE These factors can include the use of optional, processor-dependent, or implementor-defined language elements and the limits of an implementation.

### 3.5 Relationship of a conforming run unit to a conforming implementation

The execution of a run unit composed of runtime modules resulting from translation of conforming compilation units is defined only to the extent specified in standard COBOL. It is possible that a conforming run unit will not execute successfully. Execution may be unsuccessful due to factors other than lack of conformance of a run unit.

NOTE These factors can include the logical incorrectness of the compilation units, errors in the data upon which the run unit operates, and the limits of an implementation.

## 4 Terms and Definitions

For the purposes of this International Standard, the following terms and definitions apply:

- 4.1 **absolute item:** An item in a report that has a fixed position on a page.
- 4.2 **activated runtime element:** A function, method, or program placed into the active state.
- 4.3 **activating statement:** A statement that causes the execution of a function, method, or program.
- 4.4 **activating runtime element:** The function, method, or program that executed a given activating statement.
- 4.5 **active state:** The state of a function, method, or program that has been activated but has not yet returned control to the activating runtime element.
- 4.6 **alphabetic character (in the COBOL character repertoire):** A basic letter or a space character.
- 4.7 **alphanumeric character:** Any coded character in an alphanumeric coded character set, whether or not there is an assigned graphic symbol for that coded character.
- 4.8 **alphanumeric character position:** The amount of physical storage required to store, or presentation space required to print or display, a single character of an alphanumeric character set.
- 4.9 **alphanumeric character set; alphanumeric coded character set:** See alphanumeric coded character set.
- 4.10 **alphanumeric coded character set; alphanumeric character set:** A coded character set that the implementor has designated for representation of data items of usage display and alphanumeric literals.
- 4.11 **alphanumeric group item:** Any group item except for:
- a strongly-typed group item
  - a bit group item
  - a national group item.
- 4.12 **argument:** An operand specified in an activating statement that specifies the data to be passed.
- 4.13 **assumed decimal point:** A decimal point position that does not involve the existence of an actual character in a data item. An assumed decimal point has logical meaning with no physical representation.
- 4.14 **based data item:** A data item established by association of a based entry with an actual data item or allocated storage.
- 4.15 **based entry:** A data description entry that serves as a template that is dynamically associated with data items or allocated storage.
- 4.16 **basic letter:** One of the uppercase letters 'A' through 'Z' and the lowercase letters 'a' through 'z' in the COBOL character repertoire.
- 4.17 **bit:** The smallest unit in a computer's storage structure capable of representing two distinct alternatives.
- 4.18 **bit data item:** An elementary data item of category boolean and usage bit or a bit group item.
- 4.19 **block; physical record:** A physical unit of data that is normally composed of one or more logical records.
- 4.20 **boolean character:** A unit of information that consists of the value zero or one. Each boolean character may be represented in storage as a bit, an alphanumeric character, or a national character.

- 4.21 **boolean data item:** A data item capable of containing a boolean value.
- 4.22 **boolean expression:** One or more boolean operands separated by boolean operators.
- 4.23 **boolean position:** The amount of physical storage required to store, or presentation space required to print or display, a single boolean character.
- 4.24 **boolean value:** A value consisting of a sequence of one or more boolean characters.
- 4.25 **byte:** A sequence of bits representing the smallest addressable character unit in the memory of a given computer.
- 4.26 **character (in a coded character set):** A code value that constitutes the coded representation of a letter, digit, symbol, control function, or other member of a set of elements used for the organization, control, or representation of data.

NOTE COBOL processes each code value in a coded character set as though it were a character.

- 4.27 **character (in a screen item):** A graphic character.
- 4.28 **character (in COBOL character repertoire); COBOL character:** A letter, digit, or special character, independent of its coded representation, that is used in formation of the words and separators of COBOL.
- 4.29 **character (in computer storage):** A single code value of a coded character set.
- 4.30 **character boundary:** The leftmost bit of an addressing boundary in the storage of the computer.
- 4.31 **character position:** The amount of physical storage required to store, or presentation space required to print or display, a single character -- either an alphanumeric character or a national character. One element of a coded character set occupies one character position.

NOTE As an example, each element of a combining sequence in the UCS occupies one character position. For a UTF-16 surrogate value, the left surrogate occupies one character position and the right surrogate occupies another character position.

- 4.32 **character-string:** A sequence of contiguous characters that form a COBOL word, a literal, or a picture character-string.
- 4.33 **class (in object orientation):** The entity that defines common behavior and implementation for zero, one, or more objects.
- 4.34 **class (of a data item):** A designation for a set of data items having common attributes or a common range of values, defined by the PICTURE clause, the USAGE clause, or the PICTURE and USAGE clauses in a data description entry; by the definition of a predefined identifier; or by the definition of an intrinsic function.
- 4.35 **class (of data values):** A designation for a set of data values that are permissible in the content of a data item.
- 4.36 **class definition (in object orientation):** A compilation unit that defines a class of objects.
- 4.37 **clause:** An ordered set of consecutive COBOL character-strings whose purpose is to specify an attribute of an entry.
- 4.38 **COBOL character; character (in COBOL character repertoire):** See character (in COBOL character repertoire).
- 4.39 **COBOL character repertoire:** The repertoire of characters used in writing the syntax of a COBOL compilation group, except for comments and the content of non-hexadecimal alphanumeric and national literals.

4.40 **coded character set:** A set of unambiguous rules that establishes a character set and the relationship between the characters of the set and their coded representation. [ISO/IEC 10646-1]

4.41 **combining character:** A UCS member that is intended for combination with the preceding non-combining graphic character or with a sequence of combining characters preceded by a non-combining character.

4.42 **common program:** A program that, despite being directly contained within another program, may be called from any program directly or indirectly contained in that other program.

4.43 **compilation group:** A sequence of one or more compilation units submitted together for compilation.

4.44 **compilation unit:** A source unit that is not nested within another source unit.

4.45 **composite sequence:** A sequence of graphic characters consisting of a non-combining character followed by one or more combining characters. [ISO/IEC 10646-1].

4.46 **conditional statement:** A statement for which the truth value of a specified condition is evaluated and used to determine subsequent flow of control.

4.47 **conformance (for object orientation):** A unidirectional relation that allows an object to be used according to an interface other than the interface of its own class.

4.48 **conformance (for parameters):** The requirements for the relationship between arguments and formal parameters and between returning items in activating and activated runtime elements.

4.49 **control function:** An action that affects the recording, processing, transmission, or interpretation of data, and that has a coded representation consisting of one or more bytes.

NOTE This definition is the same as that in ISO/IEC 10646-1 except that "octets" is replaced by "bytes" because the term "octet" is not used in the COBOL specification.

4.50 **cultural element:** An element of data for computer use that may vary dependent on language, geographical territory, or other cultural circumstances.

4.51 **currency sign:** The COBOL character '\$', used as the default currency symbol in a picture character-string and as the default currency string that appears in the edited format of data items.

NOTE Features exist for selection of other currency strings and currency symbols.

4.52 **currency string:** The set of characters to be placed into numeric-edited data items as a result of editing operations when the item includes a currency symbol in its picture character-string.

4.53 **currency symbol:** The character used in a picture character-string to represent the presence of a currency string.

4.54 **current record:** The record that is available in the record area associated with a file.

4.55 **current volume pointer:** A conceptual entity that points to the current volume of a sequential file.

4.56 **data item:** A unit of data defined by a data description entry or resulting from the evaluation of an identifier.

4.57 **debugging line:** A source line that is optionally compiled, depending on the setting of a debugging mode switch.

4.58 **decimal point; decimal separator:** The character used to represent the radix point. The default is the character period.

4.59 **decimal separator; decimal point:** See decimal point.

- 4.60 **declarative statement:** A USE statement, which defines the conditions under which the procedures that follow the statement will be executed.
- 4.61 **de-editing:** The logical removal of all editing characters from a numeric-edited data item in order to determine that item's unedited numeric value.
- 4.62 **delimited scope statement:** Any statement that is terminated by its explicit scope terminator.
- 4.63 **digit position:** The amount of physical storage required to store, or presentation space required to print or display, a single digit.
- 4.64 **dynamic access:** An access mode in which specific logical records may be obtained from or placed into a mass storage file in a nonsequential manner and obtained from a file in a sequential manner.
- 4.65 **dynamic storage:** Storage that is allocated and released on request during runtime.
- 4.66 **end marker:** A marker for the end of a source unit.
- 4.67 **entry:** A descriptive set of consecutive clauses terminated by a separator period.
- 4.68 **entry convention:** The information required to interact successfully with a function, method, or program.
- 4.69 **exception condition:** A condition detected at runtime that indicates that an error or exception to normal processing has occurred.
- 4.70 **exception object:** An object that acts as an exception condition.
- 4.71 **exception status indicator:** A conceptual entity that exists for each exception-name.
- 4.72 **EXIT FUNCTION statement:** an abbreviation for 'EXIT statement with the FUNCTION phrase'.
- 4.73 **EXIT METHOD statement:** an abbreviation for 'EXIT statement with the METHOD phrase'.
- 4.74 **EXIT PARAGRAPH statement:** an abbreviation for 'EXIT statement with the PARAGRAPH phrase'.
- 4.75 **EXIT PERFORM statement:** an abbreviation for 'EXIT statement with the PERFORM phrase'.
- 4.76 **EXIT PROGRAM statement:** an abbreviation for 'EXIT statement with the PROGRAM phrase'.
- 4.77 **EXIT SECTION statement:** an abbreviation for 'EXIT statement with the SECTION phrase'.
- 4.78 **explicit scope terminator:** A statement-dependent word that by its presence terminates the scope of that statement.
- 4.79 **extend mode:** A mode of file processing in which records may be added at the end of a sequential file, but no records may be deleted, read, or updated.
- 4.80 **extended letter:** A letter, other than the basic letters, in the set of characters defined for the COBOL character repertoire.
- 4.81 **external data:** Data that belongs to the run unit and may be accessed by any runtime element in which it is described.
- 4.82 **external media format:** A form of data suitable for presentation or printing, including any control functions necessary for representation as readable text.
- 4.83 **external switch:** A hardware or software device, defined and named by the implementor, that is used to indicate that one of two alternate states exists.

- 4.84 **factory object:** The single object associated with a class, defined by the factory definition of that class, typically used to create the instance objects of the class.
- 4.85 **file:** A logical entity that represents a collection of logical records. There is one logical file associated with one file connector and there may be several logical files associated with one physical file.
- 4.86 **file connector:** A storage area that contains information about a file and is used as the linkage between a file-name and a physical file and between a file-name and its associated record area.
- 4.87 **file organization:** The permanent logical file structure established at the time that a file is created.
- 4.88 **file position indicator:** A conceptual entity that either is used to facilitate exact specification of the next record to be accessed, or indicates why such a reference cannot be established.
- 4.89 **file sharing:** A cooperative environment that controls concurrent access to the same physical file.
- 4.90 **fixed file attribute:** An attribute of a physical file that is established when the physical file is created and is never changed during the existence of the physical file.
- 4.91 **formal parameter:** A data-name specified in the USING phrase of the procedure division header that gives the name used in the function, method, or program for a parameter.
- 4.92 **function:** An intrinsic or user-defined procedural entity that returns a value based upon the arguments.
- 4.93 **function prototype definition:** A definition that specifies the rules governing the arguments needed for the evaluation of a particular function, the data item resulting from the evaluation of the function, and all other requirements needed for the evaluation of that function.
- 4.94 **graphic character:** A character, other than a control function, that has a visual representation normally handwritten, printed, or displayed. [ISO/IEC 10646-1].
- 4.95 **graphic symbol:** The visual representation of a graphic character or of a composite sequence. [ISO/IEC 10646-1].
- 4.96 **grouping (in locale editing):** The separation of digits into groups in number and currency formatting.
- 4.97 **grouping separator:** The character used to separate digits in numbers for ease of reading. The default is the character comma.
- 4.98 **high-order end:** The leftmost position of a string of characters or a string of bits.
- 4.99 **i-o mode:** A mode of file processing in which records can be read, updated, added, and deleted.
- 4.100 **i-o status:** A conceptual entity that exists for a file, that contains a value indicating the result of the execution of an input-output operation for that file.
- 4.101 **imperative statement:** A statement that specifies an unconditional action or that is a delimited scope statement.
- 4.102 **index:** A storage area or a register, the content of which refers to a particular element in a table.
- 4.103 **indexed organization:** The permanent logical file structure in which each record is identified by the value of one or more keys within that record.
- 4.104 **inheritance (for classes):** A mechanism for using the interface and implementation of one or more classes as the basis for another class.

- 4.105 **inheritance (for interfaces):** A mechanism for using the specification of one or more interfaces as the basis for another interface.
- 4.106 **initial program:** A program that is placed into the initial state every time the program is called.
- 4.107 **initial state:** The state of a function, method, or program when it is first activated in a run unit.
- 4.108 **input mode:** A mode of file processing in which records can only be read.
- 4.109 **instance object:** A single instance of an object defined by a class and created by a factory object.
- 4.110 **interface (of an object):** The names of all the methods defined for the object, including inherited methods; for each of the methods:
- the ordered list of its formal parameters and the description and passing technique associated with each
  - any returned value and its description
  - exceptions that may be raised.
- 4.111 **interface (the language construct):** A grouping of method prototypes.
- 4.112 **internal data:** All data described in a source unit except external data and external file connectors.
- 4.113 **invocation; method invocation:** See method invocation.
- 4.114 **key of reference:** The key, either prime or alternate, currently being used to access records within an indexed file.
- 4.115 **letter:** A basic letter or an extended letter.
- 4.116 **locale:** A facility in the user's information technology environment that specifies language and cultural conventions.
- 4.117 **lock mode:** The state of a file for which record locking is in effect that indicates whether record locking is manual or automatic.
- 4.118 **low-order end:** The rightmost position of a string of characters or a string of bits.
- 4.119 **MCS; message control system:** See message control system.
- 4.120 **message control system; MCS:** A communication control system that supports the processing of messages.
- 4.121 **method:** A procedural entity defined by a method definition within, or inherited by, a class definition as an allowable operation upon objects of that class.
- 4.122 **method data:** Data declared in a method definition.
- 4.123 **method invocation; invocation:** The request to execute a named method on a given object.
- 4.124 **method prototype:** A source element that specifies the information needed for invocation of a method and for conformance checking.
- 4.125 **national character:** A character in a national character set.
- 4.126 **national character position:** The amount of physical storage required to store, or presentation space required to print or display, a single national character.
- 4.127 **national character set; national coded character set:** See national coded character set.

- 4.128 **national coded character set; national character set:** A coded character set that the implementor has designated for representation of data items described as usage national and for national literals.
- 4.129 **national data item:** An elementary data item of class national or a national group item.
- 4.130 **native alphanumeric character set:** The computer's alphanumeric coded character set.
- 4.131 **native arithmetic:** A mode of arithmetic in which the techniques used in handling arithmetic are specified by the implementor.
- 4.132 **native character set:** An implementor-defined character set, either alphanumeric or national or both, that is used for internal processing of a COBOL runtime module. The native character set is that referenced by the keyword NATIVE in the SPECIAL-NAMES paragraph.
- 4.133 **native collating sequence:** An implementor-defined collating sequence, either an alphanumeric collating sequence or a national collating sequence, that is associated with the computer on which a runtime module is executed.
- 4.134 **native national coded character set:** The computer's national coded character set.
- 4.135 **next record:** The record that logically follows the current record of a file.
- 4.136 **null:** The state of a pointer indicating that it contains no address, or the state of an object reference indicating that it contains no reference.
- 4.137 **numeric character (in the rules of COBOL):** A character that belongs to the following set of digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- 4.138 **object:** A unit consisting of data and the methods that act upon that data.
- 4.139 **object data:** Data defined:  
— in the factory definition, except for the data defined in its methods.  
— in the instance definition, except for the data defined in its methods.
- 4.140 **object property; property:** A name that may be used to qualify an object reference to get a value from or pass a value to an object.
- 4.141 **object reference:** An explicitly- or implicitly-defined data item that contains a reference to an object.
- 4.142 **open mode:** The state of a file connector indicating input-output operations that are permitted for the associated file.
- 4.143 **optional file:** A file declared as being not necessarily present each time the runtime module is executed.
- 4.144 **outermost program:** A program, together with its contained programs, that is not contained in any other program.
- 4.145 **output file:** A file that is opened in either the output mode or extend mode.
- 4.146 **output mode:** A mode of file processing in which a file is created and records can only be added to the file.
- 4.147 **physical file:** A physical collection of physical records.
- 4.148 **physical record; block:** See block.
- 4.149 **previous record:** The record that logically precedes the current record of a file.

- 4.150 **procedure:** One or more successive paragraphs or sections in the procedure division.
- 4.151 **procedure branching statement:** A statement that causes the explicit transfer of control to a statement other than the next executable statement in the sequence in which the statements are written.
- 4.152 **processor:** The computing system, both hardware and software, used for compilation of source code or execution of run units, or both.
- 4.153 **program prototype definition:** A definition that specifies the rules governing the class of the parameters expected to be received by a particular subprogram, and any other requirements needed to transfer control to and get control and return information from that subprogram.
- 4.154 **property; object property:** See object property.
- 4.155 **random access:** An access mode in which the value of a key data item identifies the logical record that is obtained from, deleted from, or placed into a relative or indexed file.
- 4.156 **record key:** A data item within a record used to identify that record within an indexed file.
- 4.157 **record lock:** A conceptual entity that is used to control concurrent access to a given record within a shared physical file.
- 4.158 **record locking:** A facility for controlling concurrent access to records in a shared physical file.
- 4.159 **relative item:** An item in a report whose position is specified relative to the previous item.
- 4.160 **relative key:** A data item that contains a relative record number.
- 4.161 **relative organization:** The permanent logical file structure in which each record's logical position is uniquely identified by a relative record number.
- 4.162 **relative record number:** The ordinal number of a record in a file whose organization is relative.
- 4.163 **report:** A printed output described in the report section and generated from those data descriptions.
- 4.164 **report writer:** A comprehensive set of data clauses and statements that enable a print layout to be described according to its general appearance rather than through of a series of procedural steps.
- 4.165 **restricted pointer:** A pointer data item that is restricted to data items of a specified type or to programs with the same signature as a specified program.
- 4.166 **run unit:** One or more runtime modules that interact with one another and that function, at execution time, as an entity to provide problem solutions.
- 4.167 **runtime element:** The executable unit resulting from compiling a source element.
- 4.168 **runtime module:** The result of compiling a compilation unit.
- 4.169 **sequential access:** An access method in which logical records are either placed into a file in the order of execution of the statements writing the records or obtained from a file in the sequence in which the records were written to the file.
- 4.170 **sequential organization:** The permanent logical file structure in which a record is identified by a predecessor-successor relationship established when the record is placed into the file.
- 4.171 **sharing mode:** The state of a file that indicates the mode of file sharing.
- 4.172 **source element:** A source unit excluding any contained source units.

4.173 **source unit:** A sequence of statements beginning with an identification division and finishing with an end marker or the end of the compilation group, including any contained source units.

4.174 **standard arithmetic:** A mode of arithmetic in which the techniques used in handling arithmetic expressions, arithmetic statements, the SUM clause, and certain integer and numeric functions are specified in this International Standard.

4.175 **standard intermediate data item:** A temporary abstract decimal floating-point data item used to hold arithmetic operands when standard arithmetic is in effect.

4.176 **static data:** Data that has its last-used state when a runtime element is re-entered.

4.177 **subclass:** A class that inherits from another class. When two classes in an inheritance relationship are considered together, the subclass is the inheritor or inheriting class; the superclass is the inheritee or inherited class.

NOTE In the industry literature, the term derived class is also often used as an alternative to the term subclass. These terms are equivalent.

4.178 **subject of the entry:** The data item that is being defined by a data description entry.

4.179 **subscript:** A number used to refer to a specific element of a table, or in the case of the subscript 'ALL', to all elements of a table.

4.180 **superclass:** A class that is inherited by another class.

4.181 **surrogate pair:** A coded character representation for a single abstract character of the UTF-16 format of the UCS where the representation consists of a sequence of two two-octet values. The first value of the pair is a high-surrogate and the second is a low-surrogate.

4.182 **type (for type declaration):** A template that contains all the characteristics of a data item and its subordinates.

4.183 **UCS; Universal Multiple-Octet Coded Character set.** See Universal Multiple-Octet Coded Character Set.

4.184 **Universal Multiple-Octet Coded Character Set; UCS.** The coded character set defined by ISO/IEC 10646-1 together with ISO/IEC 10646-2. This coded character set includes the characters used in writing most of the languages of the modern world.

4.185 **universal object reference:** An object reference that is not restricted to a specific class or interface.

4.186 **unsuccessful execution:** The attempted execution of a statement that does not result in the execution of all the operations specified by that statement.

4.187 **variable-occurrence data item:** A table element that is repeated a variable number of times.

4.188 **zero-length item:** An item whose minimum permitted length is zero and whose length at execution time is zero.

## 5 Description techniques

The techniques used to describe standard COBOL are:

- General formats
- Rules
- Arithmetic expressions
- Informal description

### 5.1 General formats

General formats specify the syntax of the elements of standard COBOL and the sequence of arrangement of those elements.

The words, phrases, clauses, punctuation, and operands in each general format shall be written in the compilation group in the sequence given in the general format, unless otherwise specified by the rules of that format.

When more than one arrangement exists for a specific language construct, the general format is separated into multiple formats that are numbered and named.

Elements used in depicting general formats are:

- Keywords
- Optional words
- Operands
- Level numbers
- Options
  - Brackets
  - Braces
  - Choice indicators
- Ellipses
- Punctuation
- Special characters
- Meta-terms that refer to other formats

#### 5.1.1 Keywords

Keywords are reserved words or context-sensitive words. They are shown in uppercase and underlined in general formats. They are required in order to select the functionality associated with that keyword, subject to the conventions specified in 5.1.5, Options, and syntax rules specified for the general format.

#### 5.1.2 Optional words

Optional words are reserved words or context-sensitive words. They are shown in uppercase and not underlined in general formats. They may be written to add clarity when the clause or phrase in which they are defined is written in the source unit.

#### 5.1.3 Operands

An operand is an expression, a literal, or a reference to data or an exception condition. Operands are shown in lowercase and represent values or identification of items, conditions, or objects that the programmer supplies when writing the source unit.

Any term listed below refers to an instance of the corresponding element as described in the text referenced under the column labeled 'described in'. Such instances of the term are represented in lower-case and suffixed with a number (n = 1, 2, ...) for unique reference.

| <b>Operand type</b>                                                      | <b>Described in</b>                                                            | <b>Term (n = 1, 2, 3, ...)</b>                                              |
|--------------------------------------------------------------------------|--------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| Argument                                                                 | 15.2, Arguments                                                                | argument-n                                                                  |
| Expression                                                               | 8.8, Expressions                                                               | arithmetic-expression-n<br>boolean-expression-n<br>conditional-expression-n |
| Integer                                                                  | 5.4, Integer operands                                                          | integer-n                                                                   |
| Literal                                                                  | 8.3.1.2, Literals                                                              | literal-n                                                                   |
| Reference                                                                | 8.4, References                                                                |                                                                             |
| a) User-defined word, including qualification and subscription if needed | 8.3.1.1.1, User-defined words<br>8.4.1.1, Qualification<br>8.4.1.2, Subscripts | Any of the types listed in 8.3.1.1.1 suffixed by -n                         |
| b) Identifier                                                            | 8.4.2.1, Identifier                                                            | identifier-n                                                                |
| c) Exception name                                                        | 14.5.12.1.5, Exception-names and exception conditions                          | exception-name-1                                                            |

**5.1.4 Level numbers**

Specific level numbers appearing in general formats are required to be specified when the formats in which they appear are written in the source unit. Level number forms 1, 2, ..., and 9, may be written as 01, 02, ..., 09, respectively.

**5.1.5 Options**

Options are indicated in a general format by vertically stacking alternative possibilities within brackets, braces, or choice indicators. An option is selected by specifying one of the possibilities from a stack of alternative possibilities or by specifying a unique combination of possibilities from a series of brackets, braces, or choice indicators.

**5.1.5.1 Brackets**

Brackets, [ ], enclosing a portion of a general format indicate that the syntax element contained within or one of the alternatives contained within the brackets may be explicitly specified or that portion of the general format may be omitted. No default is implied for the omitted element.

**5.1.5.2 Braces**

Braces, { }, enclosing a portion of a general format indicate that the syntax element contained within the braces or one of the alternatives contained within the braces shall be explicitly specified or is implicitly selected. If one of the alternatives contains only optional words, that alternative is the default and is selected unless another alternative is explicitly specified.

**5.1.5.3 Choice indicators**

Choice indicators are a pair of bars '|' surrounded by braces or by brackets that enclose a portion of a general format. When enclosed by braces, one or more of the alternatives contained within the choice indicators shall be specified, but any single alternative shall be specified only once. When enclosed by brackets, zero or more of the alternatives contained within the choice indicators shall be specified, but any single alternative may be specified only once. The alternatives may be specified in any order.

**5.1.6 Ellipses**

In the general formats, the ellipsis represents the position at which the user elects repetition of a portion of a format. The portion of the format that may be repeated is determined as follows: given an ellipsis in a format,

scanning right to left, determine the right bracket or right brace delimiter immediately to the left of the ellipsis; continue scanning right to left and determine the logically matching left bracket or left brace delimiter; the ellipsis applies to the portion of the format between the determined pair of delimiters.

NOTE In text other than general formats, the ellipsis ( ... ) shows omission of a word or words when such omission does not impair comprehension. This is the conventional meaning of the ellipsis, and the use becomes apparent in context.

### 5.1.7 Punctuation

The separators comma and semicolon may be used anywhere the separator space is used in general formats and other syntactic specifications. In the compilation group, these separators are interchangeable.

The separator period, when specified in a general format, is required when that format is used.

### 5.1.8 Special characters

Special character words and separators that appear in formats, although not underlined, are required when such portions of the formats are used.

### 5.1.9 Meta-terms

Meta-terms appear in lowercase in general formats and are the names of subsections of general formats. Subsections are specified below the main format and are introduced by the phrase 'where x is:', with x replaced by the meta-term.

## 5.2 Rules

Except for intrinsic functions, rules are categorized as syntax rules and general rules. Intrinsic functions have argument rules and returned value rules instead.

### 5.2.1 Syntax rules

Syntax rules supplement general formats, identify equivalent words, and define or clarify the order in which words or elements may be written to form larger elements such as phrases, clauses, or statements. Syntax rules may also impose restrictions on individual words or elements, relax restrictions implied by words or elements, or define a term that may be used in the remaining syntax rules.

The rules of the PICTURE clause specified in 13.16.38.5, Precedence rules, are syntax rules.

When syntax rules specify that a word is synonymous with, an abbreviation for, or equivalent to another word (or words), those words may be written interchangeably and have the same meaning.

### 5.2.2 General rules

A general rule defines or clarifies the meaning or relationship of meanings of an element or set of elements. It is used to define or clarify the semantics of the statement and the effect that it has on either execution or compilation, and it may define a term that may be used in the remaining general rules.

The rules of the PICTURE clause specified in 13.16.38.4, Editing rules, are general rules.

### 5.2.3 Argument rules

Argument rules specify requirements, constraints, or defaults associated with arguments to intrinsic functions.

### 5.2.4 Returned value rules

Returned value rules specify the semantics of an intrinsic function.

### 5.3 Arithmetic expressions

Some rules contain arithmetic expressions that specify part or all of the results of the COBOL syntax. In presenting the arithmetic expressions, the following additional notation, or different meaning for notation, is used.

#### 5.3.1 Textually subscripted operands

When an operand is textually subscripted, as operand- $j_n$ , the term 'operand-j' identifies a specific operand and 'n' refers to the nth position or occurrence of operand-j.

NOTE An example is in the returned value rules for 15.61, PRESENT-VALUE function.

#### 5.3.2 Ellipses

Ellipses show that the number of terms and operators is variable.

### 5.4 Integer operands

- 1) When the term 'integer-n' ( $n = 1, 2, \dots$ ) is used in a general format and associated rules, it refers to a fixed-point integer literal that shall be unsigned and non-zero unless otherwise specified in the associated rules.
- 2) When the term 'integer' is used as a constraint for an operand in a syntax rule, then
  - a) if that operand is a literal, it shall be an integer literal, as defined in 8.3.1.2.2.1, Fixed-point numeric literals;
  - b) if that operand is a data-name or an identifier, it shall reference one of the following:
    1. an integer intrinsic function,
    2. a fixed-point numeric data item, other than an intrinsic function, whose description does not include any digit positions to the right of the radix point.
- 3) When the term 'integer' is used as a constraint for an operand in a general rule, that operand shall evaluate at runtime as follows:
  - a) If native arithmetic is in effect, the implementor shall define when the operand is an integer.
  - b) If standard arithmetic is in effect, the operand shall be equal to a standard intermediate data item that has the unique value zero or whose decimal fixed-point representation contains only zeros to the right of the decimal point.

#### NOTES

- 1 If the value of the exponent is greater than 31, the value of the standard intermediate data item is an integer.
  - 2 If the value of the exponent is less than 1 then the value of the standard intermediate data item is not an integer.
- c) If native arithmetic is in effect, the implementor shall define when an arithmetic expression is an integer.

### 5.5 Informal description

Substantial parts of the COBOL specification are described informally in text, tables, and diagrams other than general format diagrams. These parts normally specify semantics as described in 5.2.2, General rules, but may also include syntactical requirements in addition to those described in 5.1, General formats, and 5.2.1, Syntax rules. Syntactical requirements are distinguished from semantics by their characteristic of specifying rules for writing source code, as opposed to behavior.

## 5.6 Hyphens in text

A hyphen appearing at the end of a line of text is part of the character-string or word it divides. Hyphens are not added to divide character-strings or words across lines.

## 5.7 Verbal forms for the expression of provisions

Specific verbal forms are used in the normative clauses of this International Standard in order to distinguish among requirements for compliance, provisions allowing a freedom of choice, and recommendations. Those verbal forms are prescribed by ISO/IEC Directives, Part 3, Rules for the structure and drafting of International Standards.

Table 1, Verbal forms, summarizes the prescribed verbal forms and equivalent expressions used in this International Standard.

**Table 1 — Verbal forms**

| Provision                                                                                                                           | Verbal form | Alternative expressions                        |
|-------------------------------------------------------------------------------------------------------------------------------------|-------------|------------------------------------------------|
| a requirement on an implementation or a program, strictly to be followed for compliance to the standard                             | shall       | is required to<br>is, is to                    |
|                                                                                                                                     | shall not   | is not permitted<br>is not allowed             |
| a permission expressed by the standard                                                                                              | may         | is permitted<br>is allowed                     |
|                                                                                                                                     | need not    | is not required to                             |
| a recommendation expressed by the standard; it need not be followed                                                                 | should      | it is recommended that<br>is recommended       |
|                                                                                                                                     | should not  |                                                |
| a capability or possibility open to the user of the standard                                                                        | can         | is able to<br>it is possible to<br>is possible |
|                                                                                                                                     | cannot      |                                                |
| Alternative expressions are sometimes used to express other than a requirement; in such cases, the meaning is evident from context. |             |                                                |

## 6 Reference format

Reference format specifies the conventions for writing COBOL source text and library text. COBOL provides two reference formats: free-form reference format and fixed-form reference format. The two types of reference format may be mixed within and between source text and library text by use of SOURCE FORMAT compiler directives. The default reference format of a compilation group is fixed-form.

The following rules apply to the indicated reference formats:

### 1) Free-form and fixed-form

- a) Reference format is described in terms of character positions on a line on an input-output medium.
- b) A COBOL compiler shall accept source text and library text written in reference format.
- c) The implementor shall specify the meaning of lines and character positions.

NOTE The previous COBOL standard did not state what kind of characters were used, but alphanumeric characters of uniform size were generally assumed.

- d) For purposes of analyzing the text of a compilation group, the first character-string of a compilation group is treated as though it were preceded by a separator space and the last character-string of a compilation group is treated as though it were followed by a separator space.

### 2) Fixed-form

- a) A COBOL compiler shall process fixed-form reference format lines as though the lines had been logically converted from fixed form to free form as described in 6.4, Logical conversion.
- b) After logical conversion, the equivalent free-form lines shall meet the requirements of free-form reference format, except that lines may be longer and all characters of the computer's coded character set shall be retained in alphanumeric and national literals. (See rule 3b.)

### 3) Free-form

- a) The number of character positions on a line may vary from line to line, ranging from a minimum of 0 to a maximum of 255.
- b) The implementor shall specify any control characters that terminate a free-form line, and whether such control characters may be specified in comments and the content of alphanumeric and national literals.

## 6.1 Indicators

Indicators are instructions to the compiler for interpreting reference format. Each indicator is classified as either a fixed indicator or a floating indicator.

### 6.1.1 Fixed indicators

Fixed indicators may be specified in the indicator area of fixed-form reference format as described in 6.2, Fixed-form reference format. The following are fixed indicators:

| <u>Character</u> | <u>Indicator name</u> | <u>Indicates</u>                  |
|------------------|-----------------------|-----------------------------------|
| *                | comment indicator     | a comment line                    |
| /                | comment indicator     | a comment line with page ejection |
| D                | debugging indicator   | a debugging line                  |

|            |                        |                                                                               |
|------------|------------------------|-------------------------------------------------------------------------------|
| d          | debugging indicator    | a debugging line                                                              |
| - (hyphen) | continuation indicator | a continuation line                                                           |
| space      | source indicator       | any line that is not a comment line, a debugging line, or a continuation line |

Fixed indicators are characters in the implementor-defined coded character set or sets used for fixed-form reference format, and are not COBOL characters from the COBOL character repertoire.

NOTE This is significant in that equivalence of alphanumeric and national characters is not required for fixed indicators, nor is it precluded. The intent is to allow upward compatibility with the previous COBOL standard, which had no classification of characters as alphanumeric or national.

### 6.1.2 Floating indicators

Floating indicators may be used in fixed-form or free-form reference format. The following COBOL character-strings are floating indicators:

| <u>Character-string</u> | <u>Indicator name</u>          | <u>Indicates</u>                                                                                                                                                                                                                                                |
|-------------------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| *>                      | comment indicator              | 1) a comment line when specified as the first character-string in the program-text area;<br><br>2) an inline comment when specified following one or more character-strings in the program-text area, subject to the additional rules in 6.1.2.1, Syntax Rules. |
| >>                      | compiler directive             | a compiler-directive line when followed by a compiler-directive word - with or without an intervening space, subject to additional rules in 7.2, Compiler directives.                                                                                           |
| "-<br>'-                | literal continuation indicator | continuation of a literal when specified in an un-terminated literal with the same quotation symbol in its opening delimiter, subject to additional rules in 6.1.2.1, Syntax Rules.                                                                             |
| >>D                     | debugging indicator            | a debugging line when followed by at least one space character                                                                                                                                                                                                  |

#### 6.1.2.1 Syntax Rules

- 1) For purposes of analyzing the text of a compilation group, a space is implied immediately following a floating comment indicator.
- 2) The floating comment indicator of an inline comment shall be preceded by a separator space, and may be specified wherever a separator space may be specified, except:
  - a) as the separator space preceding a floating comment indicator
  - b) following a floating literal continuation indicator.
- 3) All the characters forming a multiple-character floating indicator shall be specified on the same line.
- 4) A floating literal continuation indicator shall be specified only for an alphanumeric, boolean, or national literal. A given literal shall not be continued with more than one form of continuation.
- 5) A floating literal continuation indicator shall not be specified on a line that contains a fixed literal continuation indicator.

- 6) For a continued alphanumeric, boolean, or national literal, the first non-blank character of each continuation line shall be the quotation symbol used in the opening delimiter of the literal.
- 7) A floating debugging indicator may be preceded on a line only by zero or more space characters.
- 8) A floating debugging indicator shall not be specified on a line that contains a fixed debugging indicator or a fixed continuation indicator.

NOTE The debugging indicators and the debugging feature are obsolete and are to be removed in the next revision of this International Standard.

## 6.2 Fixed-form reference format

The format of a fixed-form reference format line is depicted in Figure 1 — Fixed-form reference format.

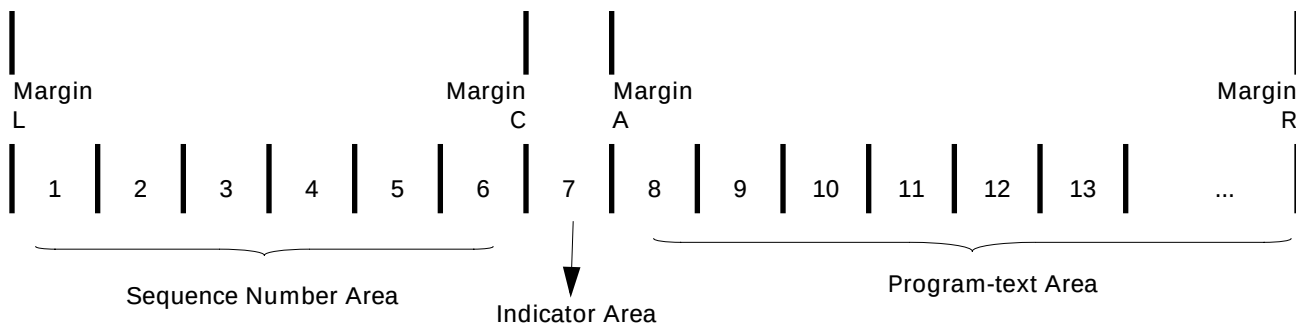


**Figure 1 — Fixed-form reference format**

Margin L is immediately to the left of the leftmost character position of a line.

Margin C is between the 6th and 7th character positions of a line.

Margin A is between the 7th and 8th character positions of a line.

Margin R is immediately to the right of the rightmost character position of the program-text area. The rightmost character position of the program-text area is a fixed position defined by the implementor.

The sequence number occupies six character positions (1-6), and is between margin L and margin C.

The indicator area is the 7th character position of a line.

The program-text area begins in character position 8 and terminates with the character position immediately to the left of margin R.

### 6.2.1 Sequence number area

The sequence number area may be used to label a line of source text or library text. The content of the sequence number area is defined by the user and may consist of any character in the computer's coded character set. There is no requirement that the content of the sequence number area appear in any particular sequence or be unique.

### 6.2.2 Indicator area

The indicator area identifies the type of a source line in accordance with the indicators specified in 6.1.1, Fixed indicators.

### 6.2.3 Program-text area

The program-text area may contain:

- 1) Comment-text of a comment line when the indicator area contains a fixed comment indicator.
- 2) Any of the following or combinations of the following, subject to further syntax specifications, when the indicator area contains a debugging indicator, a continuation indicator, or a source indicator:
  - character-strings (COBOL words)
  - separators
  - comments
  - floating indicators.
- 3) All spaces.

### 6.2.4 Continuation of lines

Any entry, sentence, statement, clause, phrase, or pseudo-text consisting of more than one character-string may be continued by starting subsequent COBOL words, literals, or picture character-strings in the program-text area of a subsequent line.

A COBOL word, literal, or picture character-string may be broken such that part of it appears on one or more subsequent lines. The subsequent lines are continuation lines; the line preceding the continuation line is a continued line. The continuation of a COBOL word, picture character-string, or literal other than an alphanumeric, boolean, or national literal is identified by a fixed continuation indicator in the continuation line.

NOTE Continuation of COBOL words is an archaic feature and its use should be avoided. It is allowed in fixed form for compatibility with the previous COBOL standard.

Continuation of an alphanumeric, boolean, or national literal is indicated when either:

- 1) A line terminates within an alphanumeric or boolean literal without a closing delimiter and the next line that is not a comment line or a blank line contains a fixed continuation indicator; or
- 2) A line terminates within an alphanumeric, boolean, or national literal that ends with a floating literal continuation indicator.

In the case of continuation with a fixed continuation indicator, any spaces at the end of the fixed-form continued line are part of the literal.

In the case of continuation with either a fixed or floating literal continuation indicator, the next line that is not a comment line or a blank line is the continuation line. The first non-space character in the program-text area of the continuation line shall be a quotation symbol matching the quotation symbol used in the opening delimiter. The continuation starts with the character immediately after the quotation symbol in the continuation line.

National literals may be continued only with a floating literal continuation indicator.

All characters composing any multiple-character separator or multiple-character indicator shall be specified on the same line. All characters forming an invocation operator shall be specified on the same line.

Comment lines and blank lines may be interspersed among lines containing the parts of a literal.

If there is no fixed continuation indicator in a line, a space is implied before the first non-blank character in the line for purposes of analyzing the text of the compilation group.

### **6.2.5 Blank lines**

A blank line is one that contains only space characters between margin C and margin R. A blank line may be written as any line of a compilation group.

### **6.2.6 Comments**

A comment consists of a comment indicator followed by comment-text. Any combination of characters from the compile-time computer's coded character set may be included in comment-text.

Comments serve only as documentation and have no effect on the meaning of the compilation group.

A comment may be a comment line or an inline comment.

#### **6.2.6.1 Comment lines**

A comment line is identified by either a fixed comment indicator or a floating comment indicator. All characters following the comment indicator up to margin R are comment-text. A comment line may be written as any line of a compilation group.

If a source listing is being produced, a comment line identified by the fixed comment indicator slant (/) causes page ejection followed by printing of the comment line; comments identified by the fixed comment indicator asterisk (\*) are printed at the next available line position of the listing.

#### **6.2.6.2 Inline comments**

A floating comment indicator preceded by one or more character-strings in the program-text area identifies an inline comment. All characters following the floating comment indicator up to margin R are comment-text. An inline comment may be written on any line of a compilation group except on a line that contains a floating literal continuation indicator.

### **6.2.7 Debugging lines**

A debugging line is identified by either a fixed debugging indicator or a floating debugging indicator.

A debugging line is permitted anywhere in a compilation unit after the SOURCE-COMPUTER paragraph.

After all COPY and REPLACE statements have been processed, a debugging line has the characteristics of a comment line if the WITH DEBUGGING MODE clause is not specified in the SOURCE-COMPUTER paragraph.

NOTE A debugging line is an obsolete element in this International Standard and is to be deleted from the next revision of standard COBOL.

## **6.3 Free-form reference format**

In free-form reference format, the source or library text may be written anywhere on a line, except that there are specific rules for comments, debugging lines, and continuation.

The indicators specified in 6.1.2, Floating indicators, identify specific elements of a compilation group. The entire free-form line constitutes the program-text area of the line.

### **6.3.1 Continuation of lines**

Any entry, sentence, statement, clause, phrase, or pseudo-text consisting of more than one character-string may be continued by writing some of the character-strings and separators that constitute it on successive lines. The last nonblank character of each line is treated as if it were followed by a space.

Alphanumeric literals, boolean literals, and national literals may be continued across lines. The line being continued is called the continued line; subsequent lines are called continuation lines. When such a literal is incomplete at the end of a line, the incomplete portion of the literal shall be terminated by a floating continuation indicator, as defined in 6.1.2, Floating indicators. The continuation indicator may optionally be followed by one or more spaces. The first nonblank character on the continuation line shall be a quotation symbol matching the quotation symbol used in the opening delimiter; the first character after the quotation symbol is the beginning character of the continuation of the literal. At least one alphanumeric character, national character, or hexadecimal digit of the literal content shall be specified on the continued line and on each continuation line.

All characters composing any multiple-character separator or multiple-character indicator shall be specified on the same line. A pair of quotation symbols indicating a single quotation symbol within a literal shall be specified on the same line.

Comment lines and blank lines may be interspersed among lines containing the parts of a literal.

### **6.3.2 Blank lines**

A blank line is one that contains nothing but space characters or is a line with zero character positions. A blank line may appear anywhere in a compilation group.

### **6.3.3 Comments**

A comment consists of a comment indicator followed by comment-text. All characters following the comment indicator up to the end of the line are comment-text.

Any combination of characters from the compile-time computer's coded character set may be included in comment-text, except as indicated in 6, Reference format, rule 3b.

Comments serve only as documentation and have no effect on the meaning of the compilation group.

A comment may be a comment line or an inline comment.

#### **6.3.3.1 Comment lines**

A comment line is identified by a floating comment indicator as the first character-string on a line. A comment line may be written as any line in a compilation group.

#### **6.3.3.2 Inline comments**

A floating comment indicator preceded by one or more character-strings on a line identifies an inline comment. An inline comment may be written on any line of a compilation group except on a line that contains a floating literal continuation indicator.

### **6.3.4 Debugging lines**

A debugging line consists of a debugging indicator, optionally preceded by one or more space characters, followed by all successive character positions on the line. A debugging line is terminated at the end of the line.

A debugging line may be written as any line in a source unit after the SOURCE-COMPUTER paragraph.

After all COPY and REPLACE statements have been processed, a debugging line has the characteristics of a comment line if the WITH DEBUGGING MODE clause is not specified in the SOURCE-COMPUTER paragraph.

NOTE A debugging line is an obsolete element in this International Standard and is to be deleted from the next revision of standard COBOL.

## 6.4 Logical conversion

Source text and library text in fixed-form reference format are logically converted to free-form reference format before the application of replacing and conditional compilation actions. Continued and continuation lines in fixed format and in free format are concatenated to remove continuation indicators, and comments and blank lines are removed from both formats. There is no restriction on the maximum line length of the free-format text resulting from logical conversion.

NOTE Fixed-form reference format is logically converted during compilation to free-form to simplify understanding of other rules of the language, for example, the COPY statement with the REPLACING phrase and the REPLACE statement. No implementor is required to perform an actual conversion as long as the effect is as though it were performed. The rules of reference format and text manipulation apply regardless of whether there is or is not an actual conversion.

The rules of logical conversion are applied to each line of a compilation group in the order that lines of source text and library text are obtained by the compiler. Lines are examined sequentially beginning with the first line of the compilation group and continuing until the end of the compilation group is reached. The resultant logically-converted compilation group is created in free-form reference format as follows:

- 1) If the line is a SOURCE FORMAT directive, the reference format mode is determined and the SOURCE FORMAT directive line is logically discarded.
- 2) If the line is a comment line or a blank line, that line is logically discarded.
- 3) If the line contains an inline comment, the inline comment is replaced by spaces and processing of that line continues.
- 4) If the line is a fixed-form or free-form line that contains a floating literal continuation indicator, the end of the program text area is set to immediately follow the character preceding the continuation indicator. The continuation indicator and any following characters are logically discarded, and processing of that line continues.
- 5) If the line is a fixed-form line that contains a fixed debugging indicator, a free-form line is created in the resultant compilation group that consists of the following: a floating debugging indicator, followed by a space, followed by the content of the fixed-form program-text area.
- 6) If the line is a fixed-form line, contains a source indicator, and is not a continuation line, the program text area of that line is copied to the resultant compilation group.
- 7) If the line is a fixed-form continuation line identified by a fixed continuation indicator:
  - a) if the continued string is an alphanumeric or boolean literal, the content of the program-text area, beginning with the first character after the initial quotation symbol, is appended immediately to the right of the last character in the latest logical line of the resultant compilation group.
  - b) otherwise, the content of the program-text area, beginning with the first non-space character, is appended immediately to the right of the last character in the latest logical line of the resultant compilation group.
- 8) If the line is a free-form line and is not a continuation line, that line is copied to the resultant compilation group.
- 9) If the line is a fixed-form or free-form continuation line that follows a line continued with a floating literal continuation indicator, the content of the program-text area, beginning with the first character after the initial quotation symbol, is appended immediately to the right of the last character in the latest logical line of the resultant compilation group.
- 10) The next input line is obtained and processing iterates at step 1.

At the end of the compilation group, processing continues with the resultant logically-converted compilation group. The implementor shall define the effect on the source listing, if any, of logical conversion.

## 7 Compiler directing facility

The compiler directing facility consists of compiler directing statements for text manipulation, compiler directives for text manipulation, and compiler directives for specifying compilation options.

The actions of compiler directing statements and compiler directives occur in two logical stages of compilation group processing - the text manipulation stage and the compilation stage.

The text manipulation stage accepts an initial compilation group, performs modifications specified by COPY and REPLACE statements and conditional compilation directives, and substitutes compilation variables into constant entries. The result is a structured compilation group for processing by the compilation stage.

The compilation stage completes the compilation process utilizing the structured compilation group.

The following are the compiler directing statements and compiler directives and the stage during which their actions take place:

| <u>Compiler directing statements</u> | <u>Stage</u>        |
|--------------------------------------|---------------------|
| COPY statement                       | Text manipulation   |
| SUPPRESS option                      | Implementor-defined |
| REPLACE statement                    | Text manipulation   |
| <br><u>Compiler directives</u>       | <br><u>Stage</u>    |
| CALL-CONVENTION                      | Compilation         |
| DEFINE                               | Text manipulation   |
| EVALUATE                             | Text-manipulation   |
| FLAG-85                              | Compilation         |
| IF                                   | Text-manipulation   |
| LEAP-SECOND                          | Compilation         |
| LISTING                              | Implementor-defined |
| PAGE                                 | Implementor-defined |
| PROPAGATE                            | Compilation         |
| SOURCE FORMAT                        | Text-manipulation   |
| TURN                                 | Compilation         |

The implementor defines the stage during which actions associated with listings, if any, take place.

The substitution of compilation-variable values into constant entries occurs in the text manipulation stage. The inclusion or exclusion of debugging lines occurs in the compilation stage. The manner and time of expansion of parameterized classes and parameterized interfaces is defined by the implementor, except that it occurs after the text manipulation stage of processing.

### 7.1 Text manipulation

The text manipulation stage of compilation group processing accepts source lines from source text and library text, selectively includes source lines through conditional compilation, and modifies text to produce a structured compilation group.

The following elements and the separators required to distinguish them shall be syntactically correct in the initial source text and library text:

- COPY statements
- compiler directives
- alphanumeric, boolean, and national literals
- fixed and floating indicators
- constant entries specifying a FROM phrase

REPLACE statements shall be syntactically correct after the action of the replacing phrase of the COPY statement.

Other indicators, language elements, and separators need not be syntactically correct until the completion of the text manipulation stage.

Text manipulation consists of processes acting on the lines of source text and library text such that the processes take effect in a specific order. An implementor may optimize the actual processing and interactions in any manner as long as the final result is the same. The following processes are applied in order:

- 1) An expanded compilation group is created in logical free-form reference format — input lines are accepted sequentially; logically converted to free-form reference format as specified in 6.4, Logical conversion; and placed in the expanded compilation group. Library text identified in COPY statements is incorporated; replacing actions are deferred to step 3. COPY statements and their incorporated text shall be identifiable in the expanded compilation group for purposes of any logically subsequent processing associated with a REPLACING phrase.

Lines that appear in the false path of an IF or EVALUATE directive, including library text identified in COPY statements, may be omitted from the expanded compilation group. SOURCE FORMAT directives in the false path shall be processed to correctly interpret input lines.

NOTE Recognition of true and false paths during logical conversion is neither required nor precluded.

The resulting lines constitute an expanded compilation group.

- 2) A conditionally-processed compilation group is created — the expanded compilation group is read and the following compiler directives and substitutions are processed in the order encountered in the expanded compilation group:
  - a) DEFINE, IF, and EVALUATE directives
  - b) substitution of compilation-variable values into constant entries
  - c) the replacing actions of all COPY statements.

The resulting lines constitute a conditionally-processed compilation group.

- 3) A structured compilation group is created — the conditionally-processed compilation group is read and the replacing actions of REPLACE statements are applied in order.

The resulting lines constitute a structured compilation group.

References to a compilation group after text manipulation processing are to the structured compilation group, which contains the lines to be used in the compilation stage.

### **7.1.1 Text manipulation elements**

Language elements referenced and not defined in 7, Compiler directing facility, have the meaning defined in 8, Language fundamentals.

#### **7.1.1.1 Compiler directing statements**

The compiler directing statements are the COPY statement and the REPLACE statement.

#### **7.1.1.2 Source text and library text**

Source text is the primary input to the compiler for a single compilation group. Library text is secondary input to the compiler as a result of processing a COPY statement.