

---

---

**Information technology — Programming  
languages, their environments and  
system software interfaces —  
Programming language Extended APL**

*Technologies de l'information — Langues de programmation, leurs  
environnements et interfaces logiciel système — Langage de  
programmation APL étendu*

---

---

Reference number  
ISO/IEC 13751:2001(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 2001

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>1</b> | <b>Scope</b>                                                    | <b>1</b>  |
| <b>2</b> | <b>Normative References</b>                                     | <b>2</b>  |
| <b>3</b> | <b>Form of this International Standard</b>                      | <b>3</b>  |
| 3.1      | Form of Definitions . . . . .                                   | 3         |
| 3.2      | Named Arrays in Examples . . . . .                              | 4         |
| 3.3      | Notes . . . . .                                                 | 5         |
| 3.4      | Cross-References . . . . .                                      | 5         |
| 3.5      | General Definitions . . . . .                                   | 5         |
| <b>4</b> | <b>Compliance</b>                                               | <b>7</b>  |
| 4.1      | Conforming Implementations . . . . .                            | 7         |
| 4.1.1    | Required Behaviour for Conforming Implementations . . . . .     | 7         |
| 4.1.2    | Required Documentation for Conforming Implementations . . . . . | 8         |
| 4.1.2.1  | Documentation of Optional-Facilities . . . . .                  | 8         |
| 4.1.2.2  | Documentation of Implementation-Defined-Facilities . . . . .    | 8         |
| 4.1.2.3  | Consistent Extensions . . . . .                                 | 9         |
| 4.2      | Conforming Programs . . . . .                                   | 9         |
| 4.2.1    | Required Behaviour for Conforming Programs . . . . .            | 9         |
| 4.2.2    | Required Documentation for Conforming Programs . . . . .        | 9         |
| <b>5</b> | <b>Definitions</b>                                              | <b>11</b> |
| 5.1      | Characters . . . . .                                            | 11        |
| 5.2      | Numbers . . . . .                                               | 13        |
| 5.2.1    | Elementary Operations . . . . .                                 | 13        |
| 5.2.2    | Number Constants . . . . .                                      | 14        |
| 5.2.3    | Subsets of the Set of Numbers . . . . .                         | 14        |
| 5.2.4    | Implementation Algorithms . . . . .                             | 16        |
| 5.2.5    | Defined Operations . . . . .                                    | 18        |
| 5.3      | Objects . . . . .                                               | 20        |
| 5.3.1    | Lists . . . . .                                                 | 20        |
| 5.3.2    | Arrays . . . . .                                                | 21        |
| 5.3.3    | Defined-Functions . . . . .                                     | 23        |
| 5.3.4    | Tokens . . . . .                                                | 25        |

|          |                                   |           |
|----------|-----------------------------------|-----------|
| 5.3.4.1  | Metaclasses                       | 26        |
| 5.3.4.2  | Index-List                        | 29        |
| 5.3.5    | Symbols                           | 29        |
| 5.3.6    | Contexts                          | 29        |
| 5.3.7    | Workspaces                        | 30        |
| 5.3.8    | Sessions                          | 30        |
| 5.3.9    | Shared-Variables                  | 32        |
| 5.3.10   | Systems                           | 33        |
| 5.4      | Evaluation Sequences              | 35        |
| 5.4.1    | Evaluation Sequence Phrases       | 36        |
| 5.4.2    | Diagrams                          | 37        |
| 5.5      | Other Terms                       | 38        |
| <b>6</b> | <b>Syntax and Evaluation</b>      | <b>39</b> |
| 6.1      | Introduction                      | 39        |
| 6.1.1    | Evaluate-Line                     | 39        |
| 6.1.2    | Character-Diagrams                | 40        |
| 6.1.3    | Evaluate-Statement                | 47        |
| 6.1.4    | Bind-Token-Class                  | 49        |
| 6.1.5    | Literal-Conversion                | 50        |
| 6.1.6    | Statement-Analysis Token-Diagrams | 50        |
| 6.2      | Reduce-Statement                  | 55        |
| 6.3      | The Phrase Evaluators             | 60        |
| 6.3.1    | Diagrams                          | 60        |
| 6.3.2    | Remove-Parentheses                | 60        |
| 6.3.3    | Evaluate-Nulladic-Function        | 60        |
| 6.3.4    | Evaluate-Monadic-Function         | 61        |
| 6.3.5    | Evaluate-Monadic-Operator         | 62        |
| 6.3.6    | Evaluate-Dyadic-Function          | 63        |
| 6.3.7    | Evaluate-Dyadic-Operator          | 65        |
| 6.3.8    | Evaluate-Indexed-Reference        | 66        |
| 6.3.9    | Evaluate-Assignment               | 67        |
| 6.3.10   | Evaluate-Indexed-Assignment       | 67        |
| 6.3.11   | Evaluate-Variable                 | 68        |
| 6.3.12   | Build-Index-List                  | 68        |
| 6.3.13   | Process-End-of-Statement          | 69        |
| 6.4      | The Form Table                    | 70        |
| <b>7</b> | <b>Scalar Functions</b>           | <b>75</b> |
| 7.1      | Monadic Scalar Functions          | 76        |
| 7.1.1    | Conjugate                         | 76        |
| 7.1.2    | Negative                          | 76        |
| 7.1.3    | Direction                         | 77        |
| 7.1.4    | Reciprocal                        | 77        |
| 7.1.5    | Floor                             | 78        |
| 7.1.6    | Ceiling                           | 78        |
| 7.1.7    | Exponential                       | 79        |

|          |                                                  |            |
|----------|--------------------------------------------------|------------|
| 7.1.8    | Natural Logarithm . . . . .                      | 79         |
| 7.1.9    | Magnitude . . . . .                              | 80         |
| 7.1.10   | Factorial . . . . .                              | 81         |
| 7.1.11   | Pi times . . . . .                               | 82         |
| 7.1.12   | Not . . . . .                                    | 83         |
| 7.2      | Dyadic Scalar Functions . . . . .                | 83         |
| 7.2.1    | Plus . . . . .                                   | 84         |
| 7.2.2    | Minus . . . . .                                  | 84         |
| 7.2.3    | Times . . . . .                                  | 85         |
| 7.2.4    | Divide . . . . .                                 | 85         |
| 7.2.5    | Maximum . . . . .                                | 86         |
| 7.2.6    | Minimum . . . . .                                | 86         |
| 7.2.7    | Power . . . . .                                  | 87         |
| 7.2.8    | Logarithm . . . . .                              | 88         |
| 7.2.9    | Residue . . . . .                                | 89         |
| 7.2.10   | Binomial . . . . .                               | 90         |
| 7.2.11   | Circular Functions . . . . .                     | 91         |
| 7.2.12   | And/LCM . . . . .                                | 93         |
| 7.2.13   | Or/GCD . . . . .                                 | 94         |
| 7.2.14   | Nand . . . . .                                   | 94         |
| 7.2.15   | Nor . . . . .                                    | 95         |
| 7.2.16   | Equal . . . . .                                  | 96         |
| 7.2.17   | Less than . . . . .                              | 97         |
| 7.2.18   | Less than or equal to . . . . .                  | 98         |
| 7.2.19   | Not equal . . . . .                              | 99         |
| 7.2.20   | Greater than or equal to . . . . .               | 100        |
| 7.2.21   | Greater than . . . . .                           | 101        |
| <b>8</b> | <b>Structural Primitive Functions</b>            | <b>102</b> |
| 8.1      | Introduction . . . . .                           | 102        |
| 8.2      | Monadic Structural Primitive Functions . . . . . | 102        |
| 8.2.1    | Ravel . . . . .                                  | 102        |
| 8.2.2    | Shape . . . . .                                  | 103        |
| 8.2.3    | Index Generator . . . . .                        | 104        |
| 8.2.4    | Table . . . . .                                  | 105        |
| 8.2.5    | Depth . . . . .                                  | 106        |
| 8.2.6    | Enlist . . . . .                                 | 107        |
| 8.3      | Dyadic Structural Primitive Functions . . . . .  | 107        |
| 8.3.1    | Reshape . . . . .                                | 107        |
| 8.3.2    | Join . . . . .                                   | 109        |
| <b>9</b> | <b>Operators</b>                                 | <b>110</b> |
| 9.1      | Introduction . . . . .                           | 110        |
| 9.2      | Monadic Operators . . . . .                      | 110        |
| 9.2.1    | Reduction . . . . .                              | 110        |
| 9.2.2    | Scan . . . . .                                   | 113        |
| 9.2.3    | N-wise Reduction . . . . .                       | 115        |

|           |                                                   |            |
|-----------|---------------------------------------------------|------------|
| 9.2.4     | Duplicate . . . . .                               | 118        |
| 9.2.5     | Commute . . . . .                                 | 118        |
| 9.2.6     | Each . . . . .                                    | 119        |
| 9.3       | Dyadic Operators . . . . .                        | 120        |
| 9.3.1     | Outer Product . . . . .                           | 120        |
| 9.3.2     | Inner Product . . . . .                           | 121        |
| 9.3.3     | Rank operator definitions . . . . .               | 123        |
| 9.3.4     | Rank operator deriving monadic function . . . . . | 124        |
| 9.3.5     | Rank operator deriving dyadic function . . . . .  | 125        |
| <b>10</b> | <b>Mixed Functions</b>                            | <b>127</b> |
| 10.1      | Monadic Mixed Functions . . . . .                 | 127        |
| 10.1.1    | Roll . . . . .                                    | 127        |
| 10.1.2    | Grade Up . . . . .                                | 129        |
| 10.1.3    | Grade Down . . . . .                              | 131        |
| 10.1.4    | Reverse . . . . .                                 | 132        |
| 10.1.5    | Monadic Transpose . . . . .                       | 133        |
| 10.1.6    | Matrix Inverse . . . . .                          | 134        |
| 10.1.7    | Execute . . . . .                                 | 135        |
| 10.1.8    | Unique . . . . .                                  | 136        |
| 10.1.9    | First . . . . .                                   | 137        |
| 10.2      | Dyadic Mixed Functions . . . . .                  | 137        |
| 10.2.1    | Join Along an Axis . . . . .                      | 137        |
| 10.2.2    | Index of . . . . .                                | 140        |
| 10.2.3    | Member of . . . . .                               | 141        |
| 10.2.4    | Deal . . . . .                                    | 142        |
| 10.2.5    | Replicate . . . . .                               | 143        |
| 10.2.6    | Expand . . . . .                                  | 145        |
| 10.2.7    | Rotate . . . . .                                  | 147        |
| 10.2.8    | Base Value . . . . .                              | 149        |
| 10.2.9    | Representation . . . . .                          | 151        |
| 10.2.10   | Dyadic Transpose . . . . .                        | 153        |
| 10.2.11   | Take . . . . .                                    | 155        |
| 10.2.12   | Drop . . . . .                                    | 156        |
| 10.2.13   | Matrix Divide . . . . .                           | 157        |
| 10.2.14   | Indexed Reference . . . . .                       | 158        |
| 10.2.15   | Indexed Assignment . . . . .                      | 159        |
| 10.2.16   | Without . . . . .                                 | 161        |
| 10.2.17   | Left . . . . .                                    | 161        |
| 10.2.18   | Right . . . . .                                   | 162        |
| 10.2.19   | Character Grade Definitions . . . . .             | 162        |
| 10.2.20   | Character Grade Down . . . . .                    | 163        |
| 10.2.21   | Character Grade Up . . . . .                      | 164        |
| 10.2.22   | Pick . . . . .                                    | 166        |
| 10.2.23   | Identical . . . . .                               | 167        |
| 10.2.24   | Disclose . . . . .                                | 168        |
| 10.2.25   | Disclose with Axis . . . . .                      | 168        |

|                                           |            |
|-------------------------------------------|------------|
| 10.2.26 Enclose . . . . .                 | 169        |
| 10.2.27 Enclose with Axis . . . . .       | 169        |
| <b>11 System Functions</b>                | <b>170</b> |
| 11.1 Introduction . . . . .               | 170        |
| 11.2 Definitions . . . . .                | 170        |
| 11.3 Diagram . . . . .                    | 171        |
| 11.4 Niladic System Functions . . . . .   | 171        |
| 11.4.1 Time Stamp . . . . .               | 171        |
| 11.4.2 Atomic Vector . . . . .            | 172        |
| 11.4.3 Line Counter . . . . .             | 172        |
| 11.4.4 Event Message . . . . .            | 173        |
| 11.4.5 Event Type . . . . .               | 174        |
| 11.5 Monadic System Functions . . . . .   | 174        |
| 11.5.1 Delay . . . . .                    | 174        |
| 11.5.2 Name Class . . . . .               | 175        |
| 11.5.3 Expunge . . . . .                  | 176        |
| 11.5.4 Name List . . . . .                | 177        |
| 11.5.5 Query Stop . . . . .               | 178        |
| 11.5.6 Query Trace . . . . .              | 179        |
| 11.5.7 Monadic Event Simulation . . . . . | 180        |
| 11.6 Dyadic System Functions . . . . .    | 180        |
| 11.6.1 Name List . . . . .                | 180        |
| 11.6.2 Set Stop . . . . .                 | 181        |
| 11.6.3 Set Trace . . . . .                | 182        |
| 11.6.4 Execute Alternate . . . . .        | 183        |
| 11.6.5 Dyadic Event Simulation . . . . .  | 184        |
| 11.6.6 Transfer Form . . . . .            | 185        |
| <b>12 System Variables</b>                | <b>186</b> |
| 12.1 Definitions . . . . .                | 186        |
| 12.2 Evaluation Sequences . . . . .       | 187        |
| 12.2.1 Comparison Tolerance . . . . .     | 187        |
| 12.2.2 Random Link . . . . .              | 188        |
| 12.2.3 Print Precision . . . . .          | 189        |
| 12.2.4 Index Origin . . . . .             | 190        |
| 12.2.5 Latent Expression . . . . .        | 191        |
| <b>13 Defined Functions</b>               | <b>192</b> |
| 13.1 Introduction . . . . .               | 192        |
| 13.2 Definitions . . . . .                | 192        |
| 13.3 Diagrams . . . . .                   | 196        |
| 13.4 Operations . . . . .                 | 200        |
| 13.4.1 Call-Defined-Function . . . . .    | 200        |
| 13.4.2 Defined-Function-Control . . . . . | 202        |
| 13.4.3 Function Fix . . . . .             | 203        |
| 13.4.4 Character Representation . . . . . | 204        |
| 13.5 Function Editing . . . . .           | 204        |

|           |                                                    |            |
|-----------|----------------------------------------------------|------------|
| 13.5.1    | Evaluate-Function-Definition-Request . . . . .     | 204        |
| 13.5.2    | Evaluate-Editing-Request . . . . .                 | 206        |
| 13.5.3    | Diagrams . . . . .                                 | 207        |
| <b>14</b> | <b>Shared Variables . . . . .</b>                  | <b>210</b> |
| 14.1      | Informal Introduction . . . . .                    | 210        |
| 14.2      | Definitions . . . . .                              | 214        |
| 14.3      | Diagrams . . . . .                                 | 214        |
| 14.4      | Operations . . . . .                               | 214        |
| 14.4.1    | Primary-Name . . . . .                             | 214        |
| 14.4.2    | Surrogate-Name . . . . .                           | 214        |
| 14.4.3    | Degree-of-Coupling . . . . .                       | 215        |
| 14.4.4    | Access-Control-Vector . . . . .                    | 215        |
| 14.4.5    | Offer . . . . .                                    | 215        |
| 14.4.6    | Retract . . . . .                                  | 216        |
| 14.4.7    | Shared-Variable-Reset . . . . .                    | 216        |
| 14.4.8    | Report-State . . . . .                             | 216        |
| 14.4.9    | Signal-Event . . . . .                             | 217        |
| 14.4.10   | Clear-Event . . . . .                              | 217        |
| 14.5      | Shared Variable Forms . . . . .                    | 217        |
| 14.5.1    | Shared Variable Reference . . . . .                | 217        |
| 14.5.2    | Shared Variable Assignment . . . . .               | 218        |
| 14.5.3    | Shared Variable Indexed Assignment . . . . .       | 219        |
| 14.6      | Shared Variable System Functions . . . . .         | 219        |
| 14.6.1    | Shared Variable Access Control Inquiry . . . . .   | 219        |
| 14.6.2    | Shared Variable Query . . . . .                    | 221        |
| 14.6.3    | Shared Variable Degree of Coupling . . . . .       | 222        |
| 14.6.4    | Shared Variable Offer . . . . .                    | 223        |
| 14.6.5    | Shared Variable Retraction . . . . .               | 224        |
| 14.6.6    | Shared Variable Access Control Set . . . . .       | 225        |
| 14.6.7    | Shared Variable State Inquiry . . . . .            | 226        |
| 14.6.8    | Shared Variable Event . . . . .                    | 227        |
| <b>15</b> | <b>Formatting and Numeric Conversion . . . . .</b> | <b>228</b> |
| 15.1      | Introduction . . . . .                             | 228        |
| 15.2      | Numeric Conversion . . . . .                       | 228        |
| 15.2.1    | Numeric-Input-Conversion . . . . .                 | 228        |
| 15.2.2    | Numeric-Output-Conversion . . . . .                | 230        |
| 15.3      | Diagrams . . . . .                                 | 231        |
| 15.4      | Operations . . . . .                               | 233        |
| 15.4.1    | Monadic Format . . . . .                           | 233        |
| 15.4.2    | Dyadic Format . . . . .                            | 237        |
| <b>16</b> | <b>Input and Output . . . . .</b>                  | <b>239</b> |
| 16.1      | Introduction . . . . .                             | 239        |
| 16.2      | Definitions . . . . .                              | 240        |
| 16.2.1    | User Facilities . . . . .                          | 240        |
| 16.2.2    | Implementation Algorithms . . . . .                | 240        |

|                                                    |            |
|----------------------------------------------------|------------|
| 16.2.3 Prompts . . . . .                           | 241        |
| 16.3 Diagrams . . . . .                            | 242        |
| 16.4 Operations . . . . .                          | 242        |
| 16.4.1 Immediate-Execution . . . . .               | 242        |
| 16.4.2 Quad Input . . . . .                        | 244        |
| 16.4.3 Quote Quad Input . . . . .                  | 245        |
| 16.4.4 Quad Output . . . . .                       | 245        |
| 16.4.5 Quote Quad Output . . . . .                 | 246        |
| <b>17 System Commands</b>                          | <b>247</b> |
| 17.1 Introduction . . . . .                        | 247        |
| 17.2 Definitions . . . . .                         | 247        |
| 17.3 Diagrams . . . . .                            | 248        |
| 17.4 Operations . . . . .                          | 248        |
| 17.4.1 Evaluate-System-Command . . . . .           | 248        |
| 17.5 Diagrams and Evaluation Sequences . . . . .   | 249        |
| <b>Annex A (normative) Component Files</b>         | <b>254</b> |
| A.1 Definitions of arguments and results . . . . . | 254        |
| A.2 Definition of functions . . . . .              | 255        |
| A.3 Errors . . . . .                               | 256        |

List of Figures

|   |                               |     |
|---|-------------------------------|-----|
| 1 | Statement Evaluation.         | 59  |
| 2 | Shared Variable Access Rules. | 213 |

List of Tables

|   |                                                      |     |
|---|------------------------------------------------------|-----|
| 1 | The Required Character Set                           | 12  |
| 2 | Relationship between Class-Name and Content          | 27  |
| 3 | The Phrase Table.                                    | 58  |
| 4 | The Form Table                                       | 71  |
| 5 | Actions for the Reduction of an Empty Vector.        | 113 |
| 6 | Actions for the N-wise Reduction of an Empty Vector. | 117 |

## 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.

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

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. 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.

International Standard ISO/IEC 13751 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

Annex A forms a normative part of this International Standard.

## Introduction

APL stands for **A Programming Language**. It is a notation invented by K. E. Iverson in the late 1950s for the description of algorithms, and expanded on and made into the programming system *APL\360* by Iverson and his colleagues Adin Falkoff, Larry Breed, Dick Lathwell, and Roger Moore in the mid-1960s.

This document, **Programming Language APL, Extended**, is a sequel to **Programming Language APL**, ISO 8485 (1989).

The principal differences that the reader will find here have to do with new features that have been added. These topics are:

- without
- greatest common divisor
- least common multiple
- duplicate
- commute
- table
- join along first axis
- mixed arrays
- overbar in names
- underbar in names
- replicate
- character grades
- grades of arrays greater than rank one
- unique
- alpha as a name
- omega as a name
- ambivalent defined functions
- event handling
- n-wise reduction
- complex arithmetic
- left
- right
- function rank operator
- defined operators
- component file system

enclose  
disclose  
enlist  
pick  
depth  
identical  
each  
first

An entry for each of these topics will be found in the index. Some new system commands have been added. Shared variable extensions have been added. Workspace Interchange Standard 2 is given, in which canonical representation vectors of type “E” are used to represent generalised arrays.



# Information technology — Programming languages, their environments and system software interfaces — Programming language Extended APL

## 1 Scope

This International Standard defines the programming language APL and the environment in which APL programs are executed. Its purpose is to facilitate interchange and promote portability of APL programs and programming skills. This International Standard specifies the syntax and semantics of APL programs and the characteristics of the environment in which APL programs are executed.

It also specifies requirements for conformance to this International Standard, including the publication of values and characteristics of implementation properties so that conforming implementations can be meaningfully compared.

This International Standard does not specify:

- implementation properties that are likely to vary with the particular equipment or operating system used;
- required values for implementation limits such as APL workspace size or numeric precision;
- the data structures used to represent APL objects;
- the facilities available through shared variables.

## 2 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 2382-15:1999, *Information technology – Vocabulary – Part 15: Programming languages*.

ISO 8485:1989, *Programming languages – APL*.

*International Register of Coded Character Sets To Be Used With Escape Sequences, Registered character set 68.* (<http://www.itscj.ipsj.or.jp/ISO-IR/068.pdf>)

### 3 Form of this International Standard

This International Standard is a formal model of an APL machine, specified as a collection of finite sets, diagrams, and evaluation sequences, and objects constructed from finite sets, diagrams, and evaluation sequences.

The finite sets are the **implementation-defined character-set**, the **implementation-defined** set of **numbers**, and the enumerated sets **array-type**, **class-names**, **keyboard-states**, **mode-names**, **required-character-set**, and **workspace-presence**.

Diagrams are directed graphs used to designate syntactic forms.

Evaluation sequences are formal procedures that operate on finite sets, diagrams, other evaluation sequences and objects defined in the International Standard.

**Objects** are entities consisting of enumerated set members and other objects. The objects are **list**, **array**, **defined-function**, **defined-operator**, **token**, **symbol**, **context**, **workspace**, **session**, **shared-variable**, and **system**.

Each object has attributes describing its state. The attributes of an array, for example, are its typical element, its shape, and its ravel.

Objects often have defined properties derived from their attributes. The rank of an array, for example, is the shape of the shape of the array.

#### 3.1 Form of Definitions

Defined terms in this International Standard are always set in **bold** and indexed. The index entry begins with the page number of the definition followed by the page numbers of all references to the term. If the definition and a use of a term occur on the same page, that page number will occur twice in the Index.

The following terms occur throughout the document and are not cross-indexed: **character**, **content**, **class**, **item**, and **number**.

**Note:** *Terms in this document include both phrases such as **implementation-parameter** and words*

such as **nil**.

Each definition in this International Standard takes one of four forms.

1. Regular definitions consist of the term being defined followed by a colon and the body of the definition. The term **Boolean** is defined in this way.
2. Members of an enumerated set are defined simply by being listed in the definition of the enumerated set. The term **nil** is defined in this way, as a member of the enumerated set **class-names**.
3. Diagrams are defined by directed graphs. The term **expression** is defined in this way.
4. Definitions of terms that designate evaluation sequences take the following form:
  - The term being defined, such as **scan**.
  - The forms that the evaluation mechanism used in the International Standard recognises as designating this term, such as  $Z \leftarrow f \setminus B$ .
  - An informal introduction indicating the purpose of the procedure. The informal introduction is considered commentary on the International Standard.
  - An evaluation sequence expressed in a formal, though English-like, language defined in the subsection **Evaluation Sequences**. A **conforming-implementation** is required to emulate the behaviour described in the evaluation sequence as modified by the **additional-requirements**, if any.
  - **Examples**, which show effects of the procedure specified by the evaluation sequence. Examples are considered commentary on the International Standard.
  - **Additional Requirements**, giving aspects of the behaviour required of this operation that cannot conveniently be expressed in the evaluation sequence.

## 3.2 Named Arrays in Examples

In the examples in this International Standard, APL identifiers beginning with *N*, such as *N234*, represent numeric arrays whose shape and content are specified by the digits in the identifier. Each digit in the identifier specifies an element of the shape vector; each element of the array, when broken down into digits, gives the index of that element.

For example,

**Example:**

```

      N4
1 2 3 4

      N23
11 12 13
21 22 23
    
```

$N_{234}$   
 111 112 113 114  
 121 122 123 124  
 131 132 133 134  
  
 211 212 213 214  
 221 222 223 224  
 231 232 233 234  
  
 $N_{234}[2;3;1]$   
 231

### 3.3 Notes

This International Standard contains notes that comment on the text of the International Standard, pointing out the significance of definitions, noting relationships between definitions, and otherwise making the text approachable. These notes are set in a different type style than the text of the International Standard, and are prefixed with the word “Note:”. The following is an example of a note.

**Note:** *This is an example of a note.*

Notes never set requirements for conformance. They may suggest desired properties, but such suggestions are not mandatory for conformance.

### 3.4 Cross-References

The heading levels in this International Standard are chapter, section, and subsection. When cross-references are given, they are always to a subsection title. In the Index, subsections are treated like definitions: the page on which a subsection begins is always the first entry in the Index; subsequent page numbers in the index show where references are made to the subsection.

### 3.5 General Definitions

For the purpose of this International Standard, the definitions given in ISO/DIS 2382/15 shall apply, together with the following:

– **Program:** An application.

**Note:** *The term is used in this International Standard to include everything from an APL expression to a collection of workspaces communicating via shared variables.*

- **Implementation:** A combination of computer hardware and software that processes (APL) **programs**.

**Note:** *An implementation is an instance of the object **system** specified by this International Standard.*

- **Facility** (of an implementation): A unit of behaviour. Every **facility** is one of:
  - **Defined-Facility:** A **facility** fully specified in this International Standard and not designated **optional** or **implementation-defined**.
  - **Optional-Facility:** A **facility** fully specified in this International Standard and designated **optional**.
  - **Implementation-Defined-Facility:** A **facility** not fully specified by this International Standard that is designated **implementation-defined**.
  - **Consistent-Extension:** A **facility** not specified in this International Standard that, for a construct this International Standard specifies as producing an error, gives some effect other than signalling the specified error.

## 4 Compliance

### 4.1 Conforming Implementations

An APL **implementation** conforms to this International Standard if it meets the following requirements for both behaviour and documentation.

#### 4.1.1 Required Behaviour for Conforming Implementations

A **conforming-implementation** shall provide all **defined-facilities** and **implementation-defined-facilities**. Each such **facility** shall behave as specified by this International Standard.

A **conforming-implementation** may provide **optional-facilities**. If provided, an **optional-facility** shall behave as specified by this International Standard. Attempted use of an **optional-facility** that is not provided shall cause the **conforming-implementation** to signal an error. A **conforming-implementation** shall not replace the error signalled by a missing **optional-facility** with other behaviour.

A **conforming-implementation** may provide **consistent-extensions**. The presence of a **consistent-extension** shall not affect the behaviour of a **conforming-program**.

A **conforming-implementation** shall use algorithms that produce results that are the same as those produced by the evaluation sequences. Mathematical function algorithms shall have at least the accuracy that the algorithms given in the evaluation sequences would produce.

**Note:** *The evaluation sequences used in this International Standard are intended to specify results, not implementation techniques.*

*The errors produced by the absence of an **optional-facility** cannot be replaced by **consistent-extensions** in a **conforming-implementation**, since this would affect the behaviour of **conforming-programs** that use the **optional-facility**.*

## 4.1.2 Required Documentation for Conforming Implementations

A **conforming-implementation** shall provide a reference document that satisfies the following requirements for the documentation of **optional-facilities**, **implementation-defined-facilities**, and **consistent-extensions**:

### 4.1.2.1 Documentation of Optional-Facilities

A **conforming-implementation** shall document the presence or absence of each of the following **optional-facilities**:

- **Shared-Variable-Protocol.** A mechanism that permits one session to exchange data with other autonomous sessions.
- **Statement-Separator-Facility.** A mechanism for placing more than one statement on a line.
- **Trace-and-Stop-Control.** A mechanism that assists the testing and correction of defined functions.
- **Complex-Arithmetic-Facility.** A mechanism that permits the generation and manipulation of complex-arithmetic results.

### 4.1.2.2 Documentation of Implementation-Defined-Facilities

A **conforming-implementation** shall document the following aspects of **implementation-defined-facilities**:

- A description of the **character-set**. This shall include a table showing the correspondence between index positions in the **atomic-vector** and the members of the **required-character-set**. If the graphic symbols used in a **conforming-implementation** are dissimilar to those in **Table 1**, the correspondence between the graphic symbols used in the implementation and those in **Table 1** shall be given.
- A description of the **numbers**. This shall include a characterisation of the internal representation used for **numbers**.
- Descriptions of the characteristics of each **implementation-algorithm**.
- The value of each **implementation-parameter**.
- A description of each **internal-value-set**.
- A description of additional **event-types**.

### 4.1.2.3 Consistent Extensions

A **conforming-implementation** shall document all **consistent-extensions** it provides. The documentation shall clearly indicate that the use of a **consistent-extension** prevents a **program** from conforming with this International Standard.

**Note:** Implementers of **conforming-implementations** should, in general, be wary of replacing **limit-errors** with **consistent-extensions**, since these errors are the only safeguards a **conforming-program** has when attempting to operate in a **conforming-implementation** whose **implementation-parameters** are inadequate to support it.

For example, if the **limit-error** on **identifier-length-limit** were not signalled, a **conforming-program** with identifiers longer than the local **identifier-length-limit** could malfunction without warning.

## 4.2 Conforming Programs

A **program** conforms to this International Standard if it meets the following requirements for both behaviour and documentation.

### 4.2.1 Required Behaviour for Conforming Programs

A **conforming-program** shall use only those **facilities** specified in this International Standard.

A **conforming-program** shall not use **consistent-extensions**.

A **conforming-program** shall not depend on the signalling of any error by a **conforming-implementation**.

**Note:** **Conforming-programs** cannot depend on error behaviour, since **consistent-extensions** that replace errors are permitted in **conforming-implementations**. A conforming program may make use of certain event handling features unreservedly (for example,  $\square ES$ ), and others with some restrictions. Consider the following example:

$$Z \leftarrow '1' \square EA 'A' \square SVO B'$$

$$Z \leftarrow (0 \ 1 \ 2 \neg 1) [0 \ 1 \ 2 \neg Z]$$

This is conforming because it only depends on the values 0, 1, and 2, which have defined meanings when returned as the result of  $\square SVO$ . The first line can return anything, but the second line filters out all but 0, 1, and 2, hence eliminating any subtle dependencies on error behaviour or consistent extensions.

### 4.2.2 Required Documentation for Conforming Programs

A **conforming-program** shall document which **optional-facilities** it requires.

A **conforming-program** shall document any specific requirements it has for **implementation-parameters**.

**Note:** *A **conforming-program** may or may not work, and may or may not produce identical results on all **conforming-implementations**, because of inherent dependencies on **implementation-parameters** or **implementation-algorithms** such as the algorithm used for matrix inversion.*

*It is suggested that **conforming-programs** provide documentation detailing the values of **implementation-parameters** they require so that their suitability for a given **conforming-implementation** can be determined readily.*

## 5 Definitions

### 5.1 Characters

- **Character-Set:** An **implementation-defined** finite set.
- **Character:** A member of the **implementation-defined** finite set **character-set**.
- **Required-Character-Set:** An enumerated set which is the union of a set of alphabets with the sets of Numeric Characters and Special Characters designated in **Table 1**. The alphabets may be any alphabet; this International Standard uses the Alphabetic Characters shown in **Table 1** to represent the dual-case Latin alphabet. The **character-set** provided by a **conforming-implementation** shall contain all members of the **required-character-set** which results from the choice of alphabet.

#### Additional Requirement:

Members of the **required-character-set** are represented in this International Standard by graphic symbols in a particular typeface, as shown in **Table 1**. The graphic symbols associated with the **required-character-set** in a **conforming-implementation** are **implementation-defined**.

**Note:** *There are conformance requirements associated with the **required-character-set**. A **conforming-implementation** must publish, as part of its required documentation, a table giving the correspondence between the graphic symbols in **Table 1** and the **atomic-vector**. Where the graphic symbols provided are not similar in appearance to those used in this International Standard, the correspondence between the graphic symbols in this International Standard and those provided by the implementation must also be provided.*

*Some of the graphic symbols given in Table 1 are not used to designate APL constructs in this International Standard. They are present to provide all keys on common terminal keyboards with corresponding symbols.*

The names in **Table 1** are not to be considered part of this International Standard.

## ISO/IEC 13751 : 2001 (E)

### Alphabetic Characters

*A B C D E F G H I J K L M N O P Q R S T U V W*  
*X Y Z*  
*A B C D E F G H I J K L M N O P Q R S T U V W*  
*X Y Z*

### Numeric Characters

0 1 2 3 4 5 6 7 8 9

### Special Characters

|                    |                         |                   |
|--------------------|-------------------------|-------------------|
| α alpha            | ∇ del tilde             | ⌞ quote quad      |
| ↓ down arrow       | Δ delta                 | ρ rho             |
| ← left arrow       | ⬆ delta stile           | ⋮ semicolon       |
| → right arrow      | ⬇ delta underbar        | ⌋ down shoe       |
| ↑ up arrow         | ¨ diaeresis             | ⌋ left shoe       |
| – bar              | ◇ diamond               | ⌋ right shoe      |
| blank              | ÷ divide                | ⌋ up shoe         |
| { left brace       | \$ dollar sign          | ⌋ up shoe jot     |
| } right brace      | • dot                   | / slash           |
| [ left bracket     | ∈ epsilon               | \ back slash      |
| ] right bracket    | = equal                 | / slash bar       |
| ∨ down caret       | ≥ greater-than or equal | \ back slash bar  |
| ⋈ down caret tilde | ι iota                  | * star            |
| < left caret       | ◦ jot                   | stile             |
| > right caret      | ≤ less-than or equal    | ⌋ down stile      |
| ^ up caret         | × multiply              | ⌋ up stile        |
| ⋈ up caret tilde   | ≠ not equal             | ⊥ up tack         |
| ○ circle           | — overbar               | ⊥ up tack jot     |
| ⊘ circle backslash | ( left parenthesis      | ⌋ right tack      |
| ⊖ circle bar       | ) right parenthesis     | ⌋ left tack       |
| ⊗ circle star      | + plus                  | ⌋ down tack       |
| ⊘ circle stile     | □ quad                  | ⌋ down tack jot   |
| : colon            | ⊞ quad divide           | ˜ tilde           |
| , comma            | ? query                 | ˜ underbar        |
| ⌣ comma bar        | ¨ diaeresis jot         | ¨ diaeresis tilde |
| ∇ del              | ‘ quote                 | ω omega           |
| ∇ del stile        | ! quote dot             | ≡ equal underbar  |

Table 1: The Required Character Set

## 5.2 Numbers

- **Number-Set:** An **implementation-defined** finite set used to represent arithmetic quantities.
- **Number:** A member of the **number-set**.

**Note:** The **number-set** is an abstraction used in this document to represent the floating-point arithmetic quantities of an arbitrary computer.

### 5.2.1 Elementary Operations

**Elementary-Operation:** One of the following four **implementation-algorithms**.

- *A Plus B.*

**Note:** **Plus** implements addition.

- *A Minus B.*

**Note:** **Minus** implements subtraction.

- *A Times B.*

**Note:** **Times** implements multiplication.

- *A Divided-by B.*

**Note:** **Divided-by** implements division.

**Note:** Each **elementary-operation** is a mapping from the Cartesian product of the **number-set** with itself back onto the union of the **number-set** and the **metaclass error**, as described in the subsection **Implementation-Algorithms**. It is assumed that members of the **metaclass error** are returned by **elementary-operations** when results cannot be represented as **numbers**. In particular, **exponent-overflow** is returned when the result of an **elementary-operation** is too large in magnitude to be represented by a **number**, **exponent-underflow** is returned when the result of an **elementary-operation** is non-zero but too small in magnitude to be represented by a **number** other than **zero**, and **domain-error** is returned by an **elementary-operation** called with arguments for which its mathematical counterpart is undefined, such as **one divided-by zero**.

The treatment of **exponent-overflow** and **exponent-underflow** is not specified by this International Standard, except as suggestions to the implementer. Implementers should avoid having **exponent-overflow** and **exponent-underflow** occur in the intermediate results of **implementation-algorithms** wherever possible. In any case, **exponent-underflow** should not cause a **limit-error**.

The fact that these fundamental algorithms are effectively undefined in the International Standard is intentional. A definition general enough to cover all known and possible floating-point systems was felt to be less useful than the requirement that the internal representation for **numbers** in a **conforming-implementation** be described in the **required-documentation** for the implementation.

### 5.2.2 Number Constants

**Note:** The following **numbers**, used in this International Standard, are defined here as terms to make clear the distinction between APL constant arrays represented as strings of APL digits, such as **1**, and members of the the **implementation-defined** finite set containing **numbers**, such as **one**.

These terms are always set in **bold** when they are used in this formal sense. They are not cross-indexed.

- **Zero:** A **number** such that, for any **number**  $A$ ,  $A$  **plus zero** is  $A$ .
- **One:** A **number** such that, for any **number**  $A$ ,  $A$  **times one** is  $A$ .
- **Negative-One:** **Zero minus one**.
- **Two:** **One plus one**.
- **Three:** **Two plus one**.
- **Four:** **Three plus one**.
- **Five:** **Four plus one**.
- **Six:** **Five plus one**.
- **One-half:** **One divided-by two**.
- **Imaginary-One:** **Negative-One to-the-power one-half**.

### 5.2.3 Subsets of the Set of Numbers

**Note:** The following subsets of the numbers are used to define the domains of operations.

- **Boolean:** **zero** or **one**.

**Note:** *False* is represented by **zero**, *true* by **one**.

**Note:** A **counting-number** is a member of the subset of the **numbers** accessible under a successor function.

- **Positive-Counting-Number:** **one** or any **number** that can be generated by  $A$  **plus one**, where  $A$  is a **positive-counting-number**.
- **Negative-Counting-Number:** The **number negative-one** or any **number** that can be generated by  $N$  **plus negative-one**, where  $N$  is a **negative-counting-number**.
- **Nonnegative-Counting-Number:** A **positive-counting-number** or **zero**.

- **Counting-Number:** A **negative-counting-number** or a **nonnegative-counting-number**.

**Note:** An **integer** is a member of the subset of the **numbers** accessible from **zero** and **one** through addition and subtraction.

- **Positive-Integer:** A **positive-counting-number** or a **number** that can be generated by  $A \text{ plus } B$ , where  $A$  and  $B$  are **positive-integers**.
- **Negative-Integer:** A **negative-counting-number** or a **number** that can be generated by  $A \text{ plus } B$ , where  $A$  and  $B$  are **negative-integers**.
- **Nonnegative-Integer:** A **positive-integer** or **zero**.
- **Integer:** A **negative-integer** or a **nonnegative-integer**.
- **Positive-Number:** A **positive-integer**, or a **number** other than **zero** that can be generated by  $A \text{ times } B$ , by  $A \text{ divided-by } B$ , or by  $A \text{ plus } B$ , where  $A$  and  $B$  are **positive-numbers**.
- **Negative-Number:** A **negative-integer**, or a **number** other than **zero** that can be generated by  $A \text{ times } B$ , by  $A \text{ divided-by } B$ , or by  $C \text{ plus } D$ , where  $A$  is a **positive-number** and  $B$ ,  $C$  and  $D$  are **negative-numbers**.
- **Nonnegative-Number:** A **positive-number** or **zero**.
- **Real-Number:** A **negative-number** or a **nonnegative-number**.
- **Complex-Number:** A **real-number**, or a **number** that can be generated by  $A \text{ plus } (B \text{ times imaginary-one})$ , where  $A$  and  $B$  are **real-numbers**.
- **Complex-Integer:** A **complex-number** that can be generated by  $A \text{ plus } (\text{imaginary-one times } B)$ , where  $A$  and  $B$  are **integers**.
- **Half-plane:** Two numbers are in the same **half-plane** if the signs of either their **real-parts** or **imaginary-parts** are nonzero and are equal.
- **Unit-square:** A **unit-square** contains all numbers for which the integral part of the real parts are the same and the integral parts of the imaginary parts are the same.
- **Arc:** The **arc** of a number is **zero** if the number is **zero**, and otherwise is the **imaginary-part** of the **natural-logarithm** of the number, and has a value greater than minus pi and less than or equal to pi.
- **Fraction:** A **number** lying in the rectangle of the **complex-plane** bounded by the two parallel lines passing through **zero** and **one** at an angle of 135 degrees with the real axis, and the two parallel lines passing through **one** and **negative-one** at an angle of forty-five degrees with the real axis, including the **numbers** on the lines going through **zero** and **negative-one**, but not including the **numbers** on the two lines through **one**.

- **First-Quadrant-Associate:** Either a **nonnegative-number**, or a **number** with both **real-part** and **imaginary-part positive-numbers**, and having the same **magnitude** as a given nonzero **number**, and separated from it by an angle of either 0, 90, 180, or 270 degrees.

**Note:** The **numbers** are assumed to be identical to the **complex-numbers**.

#### Example

Consider a normalised base-2 floating point number system with a two position exponent field and a two position mantissa field with the radix point between the digits. The decimal values representable with this system and the categories into which each falls are as follows:

| Exponent<br>in Base 2 | Mantissa<br>in Base 2 | Value<br>in Base 10 | Boolean | Counting-<br>Number | Integer |
|-----------------------|-----------------------|---------------------|---------|---------------------|---------|
| 00                    | 00                    | 0                   | Y       | Y                   | Y       |
| 00                    | 10                    | 1                   | Y       | Y                   | Y       |
| 00                    | 11                    | 1.5                 |         |                     |         |
| 01                    | 10                    | 2                   |         | Y                   | Y       |
| 01                    | 11                    | 3                   |         | Y                   | Y       |
| 10                    | 10                    | 4                   |         | Y                   | Y       |
| 10                    | 11                    | 6                   |         |                     | Y       |
| 11                    | 10                    | 8                   |         |                     | Y       |
| 11                    | 11                    | 12                  |         |                     | Y       |

As can be seen in this example, **counting-numbers** form a dense subset of the **integers**; the largest **counting-number** is typically a power of the number system base.

A given number, such as a **counting-number**, may have several hardware representations. Except for their effect on system resources, these representations should be indistinguishable to a **conforming-program**.

### 5.2.4 Implementation Algorithms

An **Implementation-Algorithm** is an algorithm used in this International Standard whose behaviour is **implementation-defined**. The following implementation algorithms are used in this International Standard.

- **Conjugate**
- **Cosine**
- **Current-Time**
- **Deal**
- **Display**
- **Divided-by**

- **Exponential**
- **Function-Display**
- **Gamma-Function**
- **Greatest-Common-Divisor**
- **Hyperbolic-Cosine**
- **Hyperbolic-Sine**
- **Hyperbolic-Tangent**
- **Inverse-Cosine**
- **Inverse-Hyperbolic-Cosine**
- **Inverse-Hyperbolic-Sine**
- **Inverse-Hyperbolic-Tangent**
- **Inverse-Sine**
- **Inverse-Tangent**
- **Magnitude**
- **Matrix-Divide**
- **Minus**
- **Modulo**
- **Natural-Logarithm**
- **Next-Definition-Line**
- **Numeric-Input-Conversion**
- **Numeric-Output-Conversion**
- **Pi-Times**
- **Plus**
- **Produce-Canonical-Representation-Vector**
- **Pseudorandom-Number-Generator**
- **Read-Keyboard**
- **Sine**
- **Tangent**
- **Times**

- Time-Stamp
- To-the-Power
- Trace-Display
- Typical-Element-For-Mixed

The following **implementation-algorithms** take a **number** as an argument and return either a **number** or an **error**: **Cosine**, **Exponential**, **Gamma-Function**, **Hyperbolic-Cosine**, **Hyperbolic-Sine**, **Hyperbolic-Tangent**, **Inverse-Cosine**, **Inverse-Hyperbolic-Cosine**, **Inverse-Hyperbolic-Sine**, **Inverse-Hyperbolic-Tangent**, **Inverse-Sine**, **Inverse-Tangent**, **Magnitude**, **Natural-Logarithm**, **Pi-Times**, **Sine**, **Tangent**.

The following **implementation-algorithms** take two **numbers** as arguments and return either a **number** or an **error**: **Divided-by**, **Minus**, **Modulo**, **Plus**, **Times**, **To-the-Power**.

The properties of the remaining **implementation-algorithms** are described in the chapters in which they are used.

### 5.2.5 Defined Operations

- **$A$  Equals  $B$** : An operation that, for  $A$  and  $B$  **numbers**, returns **one** if  $A$  and  $B$  are the same **number**, and **zero** otherwise.
- **Direction of  $A$** : An operation that, for  $A$  a **number**, returns **zero** if  $A$  is **zero**, and otherwise returns the **number** determined by the radial projection of  $A$  onto the **unit-circle**.

**Note:** For  $A$  a **real-number**, the **direction** of  $A$  is either **negative-one** or **zero** or **one**.

- **$A$  is Greater-Than  $B$** : An operation that, for  $A$  and  $B$  **real-numbers** returns **one** if  $A$  is a **positive-number** and  $B$  is a **negative-number**, returns **one** if  $A$  minus  $B$  is a **positive-number**, and returns **zero** otherwise.
- **$A$  is Less-Than  $B$** : An operation that, for  $A$  and  $B$  **real-numbers**, returns  $B$  **greater-than**  $A$ .
- **Negation of  $A$** : An operation that, for any **number**  $A$ , returns **zero minus**  $A$ .
- **Magnitude of  $A$** : An operation that, for  $A$  a **number**, returns the **non-negative real-number** determined by rotating  $A$  onto the **nonnegative real-axis**.
- **Open-Interval-Between  $A$  and  $B$** : An operation that, for any **numbers**  $A$  and  $B$ , returns a subset of the **numbers** as follows: if  $A$  is not **greater-than**  $B$ , the set of all **numbers** **greater-than**  $A$  and **less-than**  $B$ ; otherwise, the **open-interval-between**  $B$  and  $A$ .
- **Closed-Interval-Between  $A$  and  $B$** : An operation that, for any **numbers**  $A$  and  $B$ , returns a subset of the **numbers** consisting of  $A$ ,  $B$ , and the **open-interval-between**  $A$  and  $B$ .
- **Larger-Magnitude of  $A$  and  $B$** : An operation that, for any **numbers**  $A$  and  $B$ , returns the **magnitude** of  $A$  if it is **greater-than** that of  $B$ , and the **magnitude** of  $B$  otherwise.

- **Distance-Between**  $A$  and  $B$ : An operation that, for any two **numbers**  $A$  and  $B$ , returns the **magnitude** of  $A$  **minus**  $B$ .
- $A$  is **Tolerantly-Equal** to  $B$  **Within**  $C$ : An operation that, given three **numbers**  $A$ ,  $B$ , and  $C$ , with  $C$  greater than or equal to zero, returns a **Boolean**  $Z$  determined as follows:
  - If  $A$  **equals**  $B$ , then  $Z$  is **one**.
  - If  $A$  and  $B$  are not in the same **half-plane**, then  $Z$  is **zero**.
  - If the **distance-between**  $A$  and  $B$  is **less-than** or **equals**  $C$  **times** the **larger-magnitude** of  $A$  and  $B$ , then  $Z$  is **one**.
  - Otherwise,  $Z$  is **zero**.
- **Tolerant-Floor** of  $A$  **Within**  $B$ : An operation that, for  $A$  a **number** and  $B$  a **nonnegative-number**, returns a **complex-integer**  $Z$  determined as follows:
  - Let  $A$  be a member of the set of **numbers** in the **unit-square** at the **complex-integer**  $C$ , and let  $D$  be  $A$  **minus**  $C$ .
  - If the sum of the real and imaginary parts of  $D$  is **tolerantly-less-than one within**  $B$ , then  $Z$  is  $C$ .
  - Otherwise, if the **imaginary-part** of  $D$  is **greater-than** the **real-part** of  $D$ , then  $Z$  is  $C$  **plus imaginary-one**.
  - Otherwise,  $Z$  is  $C$  **plus one**.
- $A$  is **Integral-Within**  $B$ : An operation that, for a **number**  $A$  and a **positive-number**  $B$ , returns a **Boolean**  $Z$  determined as follows:
  - Let  $C$  stand for the **negation** of  $A$ .
  - $Z$  is **one** if the **tolerant-floor** of  $C$  **within**  $B$  **equals** the **negation** of the **tolerant-floor** of  $A$  **within**  $B$ , and **zero** otherwise.
- $A$  is a **Near-Integer**: An operation that, for a **number**  $A$ , returns **one** if  $A$  is **integral-within integer-tolerance**, and **zero** otherwise.

**Note:** The foregoing definition contains a forward reference to **integer-tolerance**.

- **Integer-Nearest-to**  $A$ : An operation that, for a **near-integer**  $A$ , returns the **tolerant-floor** of  $A$  **within integer-tolerance**.
- $A$  is **Near-Boolean**: An operation that, for a **near-integer**  $A$ , returns **one** if the **integer-nearest-to**  $A$  is a **Boolean**, and **zero** otherwise.

**Note:** Near-integers and near-Booleans include numbers whose **magnitude** is **less-than integer-tolerance**. The operation **integer-nearest-to** maps such numbers to **zero**.

- $A$  is **Real-Within**  $B$ : An operation that, for a **number**  $A$  and a **positive-number**  $B$ , returns a **Boolean**  $Z$  determined as follows:
  - Let  $R$  stand for the **magnitude** of the **real-part** of  $A$ , and  $I$  for the **magnitude** of the **imaginary-part** of  $A$ .
  - $Z$  is **one** if  $I$  is **less-than-or-equal** either to  $B$ , or to  $R$  **times**  $B$ , and is **zero** otherwise.
- $A$  is **Near-Real**: An operation that, for a **number**  $A$ , returns **one** if  $A$  is **real-within real-tolerance**, and **zero** otherwise.

## 5.3 Objects

**Note:** The description of APL in this document is based on objects—abstract data structures that are described in terms of characters, numbers, and elementary set theory.

Each object has a small set of named attributes that take on values that are characters, numbers, or other objects.

Each object has additionally a small set of operations that can be performed upon it.

All the objects in the description are defined here, as are all the attributes of each object. The operations are introduced as they are needed. All objects, attributes, and operations are cross-referenced in the index.

### 5.3.1 Lists

**Index:** A **nonnegative-counting-number** less than or equal to **index-limit**.

**List:** An object with the following attributes:

- **Index-Set:** A finite set  $I$  of **positive-counting-numbers** chosen so that for every subset of  $I$ , except the empty set, the cardinality of that subset is in  $I$ .
- **Value-Set:** A finite set in a specific correspondence with the **index-set** of the **list**.

**Empty-List:** A **list**, the cardinality of whose **index-set** is **zero**.

**Nonempty-List:** A **list**, the cardinality of whose **index-set** is **greater-than zero**.

**Number-of-Items** in  $L$ : The cardinality of the **index-set** of the **list**  $L$ .

**Item**  $X$  of  $L$ : An operation that, for  $X$  a member of the **index-set** of the **list**  $L$ , returns the member of the **value-set** of  $L$  associated with  $X$  by the correspondence between the **value-set** of  $L$  and the **index-set** of  $L$ .

**Note:** This form of indexing is always in origin one.

**First-Item** in  $L$ : An operation that for a **nonempty-list**  $L$  returns **item one** of  $L$ .

**Last-Item** in  $L$ : An operation that for a **nonempty-list**  $L$ , and for  $C$  the **number-of-items** in  $L$ , returns **item**  $C$  of  $L$ .

**Rest-of**  $L$ : An operation that, for a **nonempty-list**  $L$  whose **index-set** has cardinality  $C$ , returns a second **list**  $R$  whose **index-set** has cardinality  $C$  **minus one**, such that for each **item**  $J$  in the **index-set** of  $R$ , **item**  $J$  of  $R$  is **item** ( $J$  **plus one**) of  $L$ .

**Product-of**  $L$ : An operation defined on a **list of numbers**  $L$  as follows:

If the **number-of-items** in  $L$  is **zero**, **one**.

If the **number-of-items** in  $L$  is **one**, the **first-item** in  $L$ .

If the **number-of-items** in  $L$  is **greater-than one**, **first-item** in  $L$  **times** the **product-of** the **rest-of**  $L$ .

**Prefix:** A (possibly empty) **list**  $P$  is a **prefix** of a **list**  $L$  if the **index-set** of  $P$  is a subset of the **index-set** of  $L$ , and each **item** of  $P$  is the same as the corresponding **item** of  $L$ .

### 5.3.2 Arrays

**Array-Type:** An enumerated set containing the members **character**, **numeric**, and **mixed**.

**Array:** An object with the following attributes:

- **Shape-list:** A **list** of **nonnegative-counting-numbers**.
- **Ravel-list:** A **list**. Each **item** of the **list** is either a **character**, a **number**, or an **array**.  
The **number-of-items** in the **ravel-list** of an **array**  $A$  is the same as the **product-of** the **shape-list** of  $A$ .
- **Type:** A member of the enumerated set **array-type**.  
If the **type** of  $A$  is **character**, the **ravel-list** of  $A$  is a **list** of **characters**; if the **type** of  $A$  is **numeric**, the **ravel-list** of  $A$  is a **list** of **numbers**.

**Sufficient-Type** of  $L$  under  $T$ : the following operation on a **type**  $T$  and a **list**  $L$ .

If  $T$  is **character** or **numeric**, return  $T$

else if  $L$  is a **nonempty-list**

if all items of  $L$  are **numbers**, return the **type numeric**

if all items of  $L$  are **characters**, return the **type character**

otherwise, return the **type mixed**

else return the **type** of the **typical-element** of  $L$ .

**Note:** The foregoing definition is intended to provide for the cases where a mixed-array becomes a non-mixed array, and vice-versa.

**Rank** of  $A$ : An operation that, for an **array** or **array-of-vectors**  $A$ , returns the **number-of-items** in the **shape-list** of  $A$ .

**Simple:** An **array** is **simple** if each item of its **ravel-list** is either a **character** or a **number**.

**Scalar:** An **array** whose **rank** is **zero**.

**Simple-scalar:** A **simple array** whose **rank** is **zero**.

**Max-shape-of**  $B$  An operation that, for  $B$ , an **arrays** defined as follows:

Set  $B1$  to the **first-thingy** of  $B$ .  
 Set  $M1$  to  $\rho B1$ .  
 If the **count-of**  $B$  is **one**, return  $M1$ .  
 Set  $B2$  to the **remainder-of**  $B$ .  
 Set  $M2$  to the **max-shape-of**  $B2$ .  
 If the **shape-list** of  $M1$  differs from the **shape-list** of  $M2$ , signal **rank-error**.  
 Return  $M1 \uparrow M2$

**First-thingy** in  $A$ : An operation that for  $A$ , an **array**, returns an **array**  $B$ , defined as follows:  
 If  $A$  is empty, set  $B1$  to the **typical-element** of  $A$ .  
 Otherwise, set  $B1$  to the **first-item** of the **ravel-list** of  $A$ .  
 If  $B1$  is a **number** or a **character**, set  $B$  to an array, whose **ravel-list** contains the single item  $B1$ , and whose **shape-list** is empty.  
 Otherwise, set  $B$  to  $B1$ .

**Numeric-Scalar with value**  $I$ : For  $I$  a **number**, the **scalar**  $Z$  such that the **type** of  $Z$  is **numeric** and the **ravel-list** of  $Z$  is the **list**  $L$  such that the **number-of-items** in  $L$  is **one** and the **first-item** of  $L$  is  $I$ .

**Count** of  $A$ : For  $A$  an **array**, the **product-of** the **shape-list** of  $A$ .

**Vector**: An **array** whose **rank** is **one**.

**Matrix**: An **array** whose **rank** is **two**.

**Length** of  $A$ : For  $A$  a **vector**, the **count** of  $A$ .

**Typical-Element** of  $A$ : For  $A$  an **array**, if the **type** of  $A$  is **character**, the **character** blank; if the **type** of  $A$  is **numeric**, the **number** **zero**; if the **type** of  $A$  is **mixed**, the **typical-element** is determined by an **implementation-defined-algorithm**.

**Empty**: Of an **array**, the **count** of which is **zero**.

**One-Element-Vector**: A **vector**  $A$  is a **one-element-vector** if the **length** of  $A$  is **one**.

$K$  is a **Valid-Axis** of  $A$ : An operation that, for  $A$  and  $K$  **arrays**, returns **one** if  $K$  is a **scalar** or **one-element-vector**, and the **first-item** in the **ravel-list** of  $K$  is a **near-integer**, the **integer-nearest-to** which is a member of the **index-set** of the **shape-list** of  $A$ ; and returns **zero** otherwise.

**Axis**  $K$  of  $A$ : An operation that, for  $A$  an **array** and  $K$  a **valid-axis** of  $A$ , returns **item**  $K$  of the **shape-list** of  $A$ .

**Array-of-vectors**: An object with the following attributes:

- **Shape-list**: A list of **nonnegative-counting-numbers**.
- **Ravel-list**: Each **vector**  $V$  in the **ravel-list** of an **array-of-vectors**  $A$  has the **type** **sufficient-type** of the **ravel-list** of  $V$  under the **type** of  $A$ .

- **Type:** A member of the enumerated set **array-type**.

**Note:** The term **array-of-vectors** is used only as an expository device.

**Along-Axis  $K$  of  $A$ :** An operation that, for an **array**  $A$  with non-zero **rank**  $N$ , produces  $Z$ , an **array-of-vectors** of **rank**  $N-1$  such that the **shape-list** of  $Z$  is the **shape-list** of  $A$  with **item**  $K$  omitted. Each **item** of the **ravel-list** of  $Z$  is a **vector** whose **length** is the same as **axis**  $K$  of  $A$ .

**Note:** Some operations on vectors extend to arrays of greater rank in a manner similar to **scalar-extension**. **Along-axis** is an expository device used in the definition of these operations.

**Vector-Item  $I$  of  $A$ :** An operation that, for an **array-of-vectors**  $A$ , returns **item**  $I$  of the **ravel-list** of  $A$ .

**Ravel-Along-Axis  $K$  of  $A$ :** An operation that, for an **array**  $A$  with non-zero **rank**  $N$ , produces an **array-of-vectors**  $Z$  such that the **shape-list** of  $Z$  is the **product-of** the **shape-list** of  $A$  with **item**  $K$  omitted, and the **ravel-list** of  $Z$  is the **ravel-list** of **along-axis**  $K$  of  $A$ .

**First-Scalar in  $A$ :** An operation that, for  $A$  a nonempty **array**, returns a **scalar**  $Z$  such that the **ravel-list** of  $Z$  is the **first-item** in the **ravel-list** of  $A$  and the **type** of  $Z$  is the **sufficient-type** of the **ravel-list** of  $Z$  under the **type** of  $A$ .

**Remainder-of  $A$ :** An operation that, for  $A$  a non-empty **vector**, returns a **vector**  $Z$  such that the **length** of  $Z$  is **negative-one plus** the **length** of  $A$ , the **ravel-list** of  $Z$  is the **rest-of** the **ravel-list** of  $A$  and the **type** of  $Z$  is the **sufficient-type** of the **ravel-list** of  $Z$  under the **type** of  $A$ .

**Row  $I$  of  $A$ :** An operation that, for  $A$  an **array** of **rank two** and  $I$  a **number**, returns **vector-item**  $I$  **along-axis two** of  $A$ .

**Number-of-Rows in  $A$ :** An operation that, for  $A$  an **array** of **rank two**, returns **item one** of the **shape-list** of  $A$ .

**Integer-Array-Nearest-to  $A$ :** An operation that, for  $A$  an **array** having the property that each **item** of the **ravel-list** of  $A$  is a **near-integer**, returns a **numeric array**  $Z$  such that the **shape-list** of  $Z$  is the same as the **shape-list** of  $A$  and each **item** of the **ravel-list** of  $Z$  is the **integer-nearest-to** the corresponding **item** of the **ravel-list** of  $A$ .

**Boolean-Array-Nearest-to  $A$ :** An operation that, for  $A$  an **array** having the property that each **item** of the **ravel-list** of  $A$  is a **near-Boolean**, returns the **integer-array-nearest-to**  $A$ .

### 5.3.3 Defined-Functions

**Note:** A **defined-function** represents a function defined by a user. The attributes of a **defined-function** are given here. Operations are described in the **Defined Functions** chapter. **Defined-operators** are a subclass of **defined functions**.

**Defined-Function:** An object with the following attributes:

- **Canonical-Representation:** A character array whose rank is two.
- **Stop-Vector:** A numeric vector.
- **Trace-Vector:** A numeric vector.

### 5.3.4 Tokens

**Class-Names:** A set containing the following members:

|                                      |                                      |
|--------------------------------------|--------------------------------------|
| <b>Assignment-Arrow</b>              | <b>Local-Name</b>                    |
| <b>Axis-Error</b>                    | <b>Monadic-Operator</b>              |
| <b>Branch</b>                        | <b>Nil</b>                           |
| <b>Branch-Arrow</b>                  | <b>Niladic-Defined-Function</b>      |
| <b>Character-Literal</b>             | <b>Niladic-Defined-Function-Name</b> |
| <b>Clear-State-Indicator</b>         | <b>Niladic-System-Function-Name</b>  |
| <b>Colon</b>                         | <b>Not-Copied</b>                    |
| <b>Command-Complete</b>              | <b>Not-Erased</b>                    |
| <b>Committed-Value</b>               | <b>Not-Found</b>                     |
| <b>Complete-Index-List</b>           | <b>Not-Saved</b>                     |
| <b>Constant</b>                      | <b>Numeric-Literal</b>               |
| <b>Defined-Function</b>              | <b>Partial-Index-List</b>            |
| <b>Defined-Function-Name</b>         | <b>Primitive</b>                     |
| <b>Defined-Dyadic-Operator</b>       | <b>Primitive-Function</b>            |
| <b>Defined-Dyadic-Operator-Name</b>  | <b>Rank-Error</b>                    |
| <b>Defined-Monadic-Operator</b>      | <b>Result-Name</b>                   |
| <b>Defined-Monadic-Operator-Name</b> | <b>Right-Argument-Name</b>           |
| <b>Definition-Error</b>              | <b>Right-Axis-Bracket</b>            |
| <b>Distinguished-Identifier</b>      | <b>Right-End-of-Statement</b>        |
| <b>Domain-Error</b>                  | <b>Right-Index-Bracket</b>           |
| <b>Dyadic-Operator</b>               | <b>Right-Operand-Name</b>            |
| <b>Elided-Index-Marker</b>           | <b>Right-Parenthesis</b>             |
| <b>Escape</b>                        | <b>Semicolon</b>                     |
| <b>Implicit-Error</b>                | <b>Shared-Variable</b>               |
| <b>Incorrect-Command</b>             | <b>Shared-Variable-Name</b>          |
| <b>Index-Error</b>                   | <b>Simple-Identifier</b>             |
| <b>Index-Separator</b>               | <b>Small-Circle</b>                  |
| <b>Interrupt</b>                     | <b>Syntax-Error</b>                  |
| <b>Label</b>                         | <b>System-Function-Name</b>          |
| <b>Label-Name</b>                    | <b>System-Variable-Name</b>          |
| <b>Left-Argument-Name</b>            | <b>Unwind</b>                        |
| <b>Left-Axis-Bracket</b>             | <b>Value-Error</b>                   |
| <b>Left-End-of-Statement</b>         | <b>Variable</b>                      |
| <b>Left-Index-Bracket</b>            | <b>Variable-Name</b>                 |
| <b>Left-Operand-Name</b>             |                                      |
| <b>Left-Parenthesis</b>              |                                      |
| <b>Length-Error</b>                  |                                      |
| <b>Limit-Error</b>                   |                                      |

**Token:** An object with the following attributes:

- **Class:** A member of the set **Class-Names**.

- **Content:** A **number**, a **character**, or an object, according to the **class** of the **token**, as indicated in **Table 2**.

**Note:** **Tokens** represent the internal objects manipulated by an implementation of APL.

The **class** of a **token** is a straightforward indication of the sort of object it represents. A **token** of **class nil**, for example, is used to return a **value-error** from a defined function that does not set its result name. Note that **class-names** is an enumerated set. This means that **nil** has no definition other than its literal appearance on this page. Its significance, like the significance of APL characters in this document, lies in the use of its name in evaluation sequences.

The **content** of a **token** varies with the **class** of the **token**. If specified, the content is a **character**, a **list of characters**, a **list of numbers**, an **array**, an **index-list**, a **shared-variable**, or a **defined-function**.

#### 5.3.4.1 Metaclasses

**Note:** Metaclasses are subsets of the enumerated set **class-names**. They are used to shorten evaluation sequences by abstracting a sequence of tests on the **class** of a **token** into a single test for membership in a metaclass. “If *B* is an **error**,” is equivalent to “If the **class** of *B* is in the metaclass **error**.”

- **Metaclass:** A subset of the enumerated set **class-names**.
- **Identifier:** A metaclass containing **simple-identifier** and **distinguished-identifier**.
- **Literal:** A metaclass containing **character-literal** and **numeric-literal**.
- **Lexical-Unit:** A metaclass containing **primitive**, **literal**, and **identifier**.
- **Value:** A metaclass containing **committed-value** and **constant**.
- **Delimiter:** A metaclass containing **primitive-function**, **branch-arrow**, **assignment-arrow**, **left-end-of-statement**, **right-end-of-statement**, **left-index-bracket**, **right-index-bracket**, **elided-index-marker**, **left-axis-bracket**, **right-axis-bracket**, **left-parenthesis**, **right-parenthesis**, **small-circle**, and **semicolon**.
- **Defined-Name:** A metaclass containing **shared-variable-name**, **variable-name**, **defined-function-name**, **defined-dyadic-operator-name**, **defined-monadic-operator-name**, **niladic-defined-function-name**, **result-name**, **left-argument-name**, **right-argument-name**, **label-name**, and **local-name**.
- **Defined-Operator:** A metaclass containing **defined-dyadic-operator** and **defined-monadic-operator**.
- **System-Name:** A metaclass containing **system-variable-name**, **system-function-name**, and **niladic-system-function-name**.
- **Classified-Name:** A metaclass containing the members of **system-name** and **defined-name**.
- **Syntactic-Unit:** A metaclass containing the members of **classified-name** and **delimiter**.

| <b>Class-Name</b>                    | <b>Content</b>       |
|--------------------------------------|----------------------|
| <b>Assignment-Arrow</b>              |                      |
| <b>Axis-Error</b>                    |                      |
| <b>Branch</b>                        | An array             |
| <b>Branch-Arrow</b>                  |                      |
| <b>Character-Literal</b>             | A list of characters |
| <b>Clear-State-Indicator</b>         |                      |
| <b>Colon</b>                         |                      |
| <b>Command-Complete</b>              |                      |
| <b>Committed-Value</b>               | An array             |
| <b>Complete-Index-List</b>           | An index-list        |
| <b>Constant</b>                      | An array             |
| <b>Defined-Function</b>              | A defined-function   |
| <b>Defined-Function-Name</b>         | A list of characters |
| <b>Defined-Dyadic-Operator</b>       | A defined-function   |
| <b>Defined-Dyadic-Operator-Name</b>  | A list of characters |
| <b>Defined-Monadic-Operator</b>      | A defined-function   |
| <b>Defined-Monadic-Operator-Name</b> | A list of characters |
| <b>Definition-Error</b>              |                      |
| <b>Distinguished-Identifier</b>      | A list of characters |
| <b>Domain-Error</b>                  |                      |
| <b>Dyadic-Operator</b>               | A character          |
| <b>Elided-Index-Marker</b>           |                      |
| <b>Escape</b>                        |                      |
| <b>Implicit-Error</b>                |                      |
| <b>Incorrect-Command</b>             |                      |
| <b>Index-Error</b>                   |                      |
| <b>Index-Separator</b>               |                      |
| <b>Interrupt</b>                     |                      |
| <b>Label</b>                         | An array             |
| <b>Label-Name</b>                    | A list of characters |
| <b>Left-Argument-Name</b>            | A list of characters |
| <b>Left-Axis-Bracket</b>             |                      |
| <b>Left-End-of-Statement</b>         |                      |
| <b>Left-Index-Bracket</b>            |                      |
| <b>Left-Operand-Name</b>             | A list of characters |
| <b>Left-Parenthesis</b>              |                      |
| <b>Length-Error</b>                  |                      |
| <b>Limit-Error</b>                   |                      |

Table 2: Relationship between Class-Name and Content

| <b>Class-Name</b>                    | <b>Content</b>       |
|--------------------------------------|----------------------|
| <b>Local-Name</b>                    | A list of characters |
| <b>Monadic-Operator</b>              | A character          |
| <b>Nil</b>                           |                      |
| <b>Niladic-Defined-Function</b>      | A defined-function   |
| <b>Niladic-Defined-Function-Name</b> | A list of characters |
| <b>Niladic-System-Function-Name</b>  | A list of characters |
| <b>Not-Copied</b>                    |                      |
| <b>Not-Erased</b>                    |                      |
| <b>Not-Found</b>                     |                      |
| <b>Not-Saved</b>                     |                      |
| <b>Numeric-Literal</b>               | A list of numbers    |
| <b>Partial-Index-List</b>            | An index-list        |
| <b>Primitive</b>                     | A character          |
| <b>Primitive-Function</b>            | A character          |
| <b>Rank-Error</b>                    |                      |
| <b>Result-Name</b>                   | A list of characters |
| <b>Right-Argument-Name</b>           | A list of characters |
| <b>Right-Axis-Bracket</b>            |                      |
| <b>Right-End-of-Statement</b>        |                      |
| <b>Right-Index-Bracket</b>           |                      |
| <b>Right-Operand-Name</b>            | A list of characters |
| <b>Right-Parenthesis</b>             |                      |
| <b>Semicolon</b>                     |                      |
| <b>Shared-Variable</b>               | A shared-variable    |
| <b>Shared-Variable-Name</b>          | A list of characters |
| <b>Simple-Identifier</b>             | A list of characters |
| <b>Small-Circle</b>                  |                      |
| <b>Syntax-Error</b>                  |                      |
| <b>System-Function-Name</b>          | A list of characters |
| <b>System-Variable-Name</b>          | A list of characters |
| <b>Unwind</b>                        |                      |
| <b>Value-Error</b>                   |                      |
| <b>Variable</b>                      | An array             |
| <b>Variable-Name</b>                 | A list of characters |

Table 2: (Continued)

- **Error:** A metaclass containing **axis-error**, **domain-error**, **implicit-error**, **index-error**, **length-error**, **limit-error**, **rank-error**, **syntax-error**, **value-error**, and **interrupt**.

**Note:** This metaclass includes only errors that occur in the evaluation of APL statements.

- **Report:** A metaclass containing **incorrect-command**, **not-copied**, **not-erased**, **not-found**, and **not-saved**.
- **Exception:** A metaclass containing **branch**, **escape**, **clear-state-indicator**, **unwind**, and the members of **error** and **report**.
- **Result:** A metaclass containing **nil** and the members of **exception** and **value**.

#### 5.3.4.2 Index-List

- **Index-List:** A (possibly empty) **list** consisting of **tokens** whose **class** is either **constant** or **elided-index-marker**.

### 5.3.5 Symbols

**Symbol:** An object with the following attributes:

- **Name:** A **list** of **characters**.
- **Referent-List:** A **list** of **tokens**.

### 5.3.6 Contexts

- **Mode-Names:** An enumerated set containing the members **immediate-execution**, **execute**, **function-definition**, **quad-input**, and **defined-function**.

**Context:** An object with the following attributes:

- **Mode:** An element of the enumerated set **mode-names**.
- **Stack:** A **list** of **tokens**.
- **Current-Line:** A **list** of **characters**.
- **Current-Statement:** A **list** of **tokens**.
- **Current-Function:** If **mode** is **defined-function**, a **defined-function**; otherwise undefined.
- **Current-Line-Number:** If **mode** is **defined-function**, an **index**; otherwise undefined.

### 5.3.7 Workspaces

**Note:** *The workspace is the basic organisational unit in an APL system. A workspace contains data, programs, execution status, and environmental information.*

- **Workspace-Presence:** An enumerated set containing **absent** and **present**.

**Workspace:** An object with the following attributes:

- **Owner:** A **user-identification**.
- **Workspace-Name:** A **list** of **characters**.
- **Symbol-Table:** A **list** of all **symbols** whose **names** are distinct.
- **State-Indicator:** A **list** of **contexts**.
- **Existential-Property:** A member of the enumerated set **workspace-presence**.

**Clear-Workspace:** A **workspace** with the following values:

- **Owner:** **this-owner**.
- **Workspace-Name:** The **clear-workspace-identifier**.
- **Symbol-Table:** A **list** of all **symbols** whose **names** are distinct and whose **referent-lists** each consist of the **list** whose only member is **nil**.
- **State-Indicator:** An empty **list** of **contexts**.
- **Existential-Property:** **absent**.

### 5.3.8 Sessions

**Note:** *A user interacts with an APL system through a session, an abstraction that represents a hypothetical machine capable of carrying out the evaluation sequences in the International Standard.*

- **Session-Identification:** Either a **number** or a **list** of **characters**, depending upon the **implementation-parameter session-identification-type**.
- **User-Identification:** Either a **number** or a **list** of **characters**, depending upon the **implementation-parameter user-identification-type**.

- **Keyboard-States:** An enumerated set containing the members **open-keyboard** and **locked-keyboard**.

**Session:** An object with the following attributes:

- **Active-Workspace:** A **workspace**.
- **This-Session:** A **session-identification**.
- **This-Owner:** A **user-identification**.
- **Attention-Flag:** A member of the enumerated set **Boolean**.
- **Keyboard-State:** A member of the enumerated set **keyboard-states**.
- **Current-Prompt:** A **character vector**.
- **Quote-Quad-Prompt:** A **character vector**.
- **Event-Time:** A **nonnegative-number**.
- **Event-Message:** A **character array** giving, in an **implementation-defined** form, the **event-report** and any other information the implementation deems helpful in locating the error's source.
- **Event-Type:** A two-element integer vector corresponding to the most recent event. Defined values are:

- 0 0 — No error
- 0 1 — Undefined event

There may be additional **implementation-defined** values as well as values supplied by a program via  $\square ES$ .

- **Current-Context:** The **first-item** in the **state-indicator** of the **active-workspace**.
- **Current-Stack:** The **stack** of the **current-context**.
- **Symbol-Named-By  $T$ :** The **symbol** in the **symbol-table** of the **active-workspace** whose **name** is the same as the **content** of the **token  $T$** .
- **Current-Referent of  $T$ :** The **first-item** in the **referent-list** of the **symbol-named-by  $T$** .
- **Current-Class of  $T$ :** The **class** of the **current-referent** of  $T$ .
- **Current-Content of  $T$ :** The **content** of the **current-referent** of  $T$ .
- **Comparison-Tolerance:** The **current-content** of  $\square CT$ .
- **Random-Link:** The **current-content** of  $\square RL$ .

- **Print-Precision:** The **current-content** of  $\square PP$ .
- **Index-Origin:** The **current-content** of  $\square IO$ .
- **Latent-Expression:** The **current-content** of  $\square LX$ .

**System-Parameter:** Any of **comparison-tolerance**, **random-link**, **print-precision**, **index-origin**, or **latent-expression**.

The initial values of **system-parameters** in a **clear-workspace** are **implementation-defined**.

### 5.3.9 Shared-Variables

**Note:** A **shared-variable** is a variable shared between two **sessions**.

Shared Variables are an **optional-facility**.

**Shared-Variable:** An object with the following attributes:

- **Session-A:** A **session-identification**.
- **Session-A-Active:** A **Boolean**.
- **Session-A-ACV:** A **Boolean vector** of **length four**.

**Note:** *ACV stands for access control vector.*

- **Session-B:** A **session-identification**.
- **Session-B-Active:** A **Boolean**.
- **Session-B-ACV:** A **Boolean vector** of **length four**.
- **Shared-Name:** An **identifier**.
- **Shared-Value:** A **token**, either a **constant** or **nil**.
- **State:** An **integer**, either **zero**, **one**, or **two**.
- **Session-A-Event:** A **Boolean**.
- **Session-B-Event:** A **Boolean**.

**Note:** The operations in the chapter **Shared Variables** use the **shared-variable** attributes as follows:

- **Session-A:** The **session-identification** of the first session to offer the **shared-variable**.

- **Session-A-Active:** A Boolean; **one** if session-A is currently sharing this **shared-variable**, **zero** if not.
- **Session-A-ACV:** The contribution of session-A to the ACV.
- **Session-B:** The **session-identification** of the **session** with which session-A offered to share the **shared-variable**; this may be the **general-offer** while **state** is **one**.
- **Session-B-Active:** A Boolean; **one** if session-B is currently sharing this **shared-variable**, **zero** if not.
- **Session-B-ACV:** The contribution of session-B to the ACV.
- **Shared-Name:** An **identifier**, the name session-A designated for this **shared-variable**.
- **Shared-Value:** A **token** of class **constant** or **nil**; the current value of this **shared-variable**.
- **State:** An **integer**, either **zero**, **one**, or **two**, used in combination with the ACV to determine which operations must be delayed.

### 5.3.10 Systems

**Note:** A **system** represents a set of active APL users, a library, and, optionally, a shared variable facility.

**System:** An object with the following attributes:

- **Library:** A list of **workspaces** in which each combination of possible values for the attributes **owner** and **workspace-name** occurs exactly once.
- **Active-Users:** A list of **sessions**.
- **Shared-Variable-List:** A list of **shared-variables**.
- **Implementation-Parameters:** The following quantities, referred to in this International Standard by name, whose values are **implementation-defined**:
  - **Atomic-Vector:** An **implementation-defined character vector** containing every member of the **required-character-set** exactly once.
  - **Initial-Comparison-Tolerance:** A member of the **internal-value-set** for **comparison-tolerance** that is the value of **comparison-tolerance** in a **clear-workspace**.
  - **Initial-Index-Origin:** A member of the **internal-value-set** for **index-origin** that is the value of **index-origin** in a **clear-workspace**.
  - **Initial-Latent-Expression:** A member of the **internal-value-set** for **latent-expression** that is the value of **latent-expression** in a **clear-workspace**.
  - **Initial-Print-Precision:** A member of the **internal-value-set** for **print-precision** that is the value of **print-precision** in a **clear-workspace**.
  - **Initial-Random-Link:** A member of the **internal-value-set** for **random-link** that is the value of **random-link** in a **clear-workspace**.

- **Initial-Event-Message:** The **empty-event-message**.
- **Reduction-Style:** One of the two symbolics **Enclose-Reduction-Style** or **Insert-Reduction-Style**, indicating which of the two styles of reduction the system uses.
- **Initial-Event-Type:** The two-element integer vector  $(0, 0)$ .
- **Clear-Workspace-Identifier:** A list of characters.
- **Positive-Number-Limit:** The **real-number greater-than** all other **real-numbers**.
- **Negative-Number-Limit:** The **real-number less-than** all other **real-numbers**.
- **Positive-Counting-Number-Limit:** The **counting-number greater than** all other **counting-numbers**.
- **Negative-Counting-Number-Limit:** The **counting-number less-than** all other **counting-numbers**.
- **Index-Limit:** The **index** greater than all other indices. This value specifies the maximum value of any **item** of the **shape-list** of any **array**, ignoring storage limitations.
- **Count-Limit:** An **index** not greater than **index-limit** that specifies the maximum value for the **count** of an array, ignoring storage limitations.
- **Rank-Limit:** An **index** not greater than **count-limit** that specifies the maximum value for the **rank** of any array, ignoring storage limitations.
- **Workspace-Name-Length-Limit:** A **positive-counting-number** that specifies the maximum number of **characters** in a **workspace-name**.
- **Identifier-Length-Limit:** A **positive-counting-number** not greater than **count-limit** that specifies the maximum number of **characters** in an **identifier**.
- **Quote-Quad-Output-Limit:** A **nonnegative-counting-number** that specifies the maximum number of **characters** that can be used in a prompt set by **Quote Quad Output**.
- **Comparison-Tolerance-Limit:** The largest **real-number** permitted by the implementation for the system parameter **comparison-tolerance**.
- **Integer-Tolerance:** A **real-number** not greater than **comparison-tolerance-limit** used to determine whether a given number is to be considered integral.
- **Real-Tolerance:** A **nonnegative-number** not greater than **comparison-tolerance-limit** used to determine whether a given number is to be considered real.
- **Full-Print-Precision:** The smallest **positive-counting-number** which, when used as the value of **print-precision**, causes **numeric-output-conversion** to produce a unique **numeric-scalar-literal** for every **number**.
- **Print-Precision-Limit:** The largest **positive-counting-number** permitted by the implementation for the system parameter **print-precision**.

**Note:** **Print-precision-limit** must be at least **full-print-precision** if every unique **number** is to have a unique decimal representation.

- **Exponent-Field-Width:** A **positive-counting-number** giving the number of places, including sign and trailing blanks, used for the exponent field in the result of **dyadic format**.

- **Session-Identification-Type:** A member of the enumerated set **array-type**, either **character** or **numeric**.
- **User-Identification-Type:** A member of the enumerated set **array-type**, either **character** or **numeric**.
- **Indent-Prompt:** A list of **characters** used to indicate to the user that the **session** is in **immediate-execution** mode.
- **Quad-Prompt:** A list of **characters** used to indicate to the user that the **session** is in **quad-input** mode.
- **Function-Definition-Prompt:** A list of **characters** used to indicate to the user that the **session** is in **function-definition** mode.
- **Line-Limit:** A **positive-counting-number** that specifies the maximum number of lines permitted in a defined function, ignoring storage limitations.
- **Definition-Line-Limit:** A **positive-number** that specifies the maximum value of a line number in **function-definition** mode.
- **General-Offer:** A reserved **session-identification** used to indicate that the offerer of a **shared-variable** is willing to share the proffered variable with any other **session**. This **implementation-parameter** is required only if the **optional-facility shared-variable-protocol** is present.

Any action that would cause a limit specified by an implementation parameter to be exceeded shall signal a **limit-error**.

## 5.4 Evaluation Sequences

The evaluation sequences that define APL operations in the remainder of this International Standard are written in English in the imperative mood. The English phrases used in evaluation sequences are restricted to the set specified in this subsection.

Indentation is used in evaluation sequences to indicate scope, typically of the consequent of an implication.

For example, in the evaluation sequence fragment below, the indented text is evaluated only if both *A* and *B* are **vectors**; the “otherwise” clause is evaluated only if at least one of *A* or *B* is not a **vector**.

```
...
  If A is a vector and B is a vector,
    If A is empty and ...
  ...
  Otherwise, return ...
```

Expressions in APL are used in evaluation sequences. A given evaluation sequence uses only APL operations that have been specified earlier in the International Standard. Where indices are generated or used by APL expressions in evaluation sequences, they are evaluated with origin **one**. The APL relational operations are never used in evaluation sequences unless they are qualified with the value to be used for **comparison-tolerance**.

### 5.4.1 Evaluation Sequence Phrases

**Note:** The following phrases are used in evaluation sequences. They are not set in **bold** type nor cross-referenced when employed in evaluation sequences.

- **For all**  $A$ ,  $C$ : An evaluation sequence phrase used to specify that the action or condition specified by the consequent  $C$  is to be performed or checked for every value in the antecedent  $A$ .

Example:

...  
 For all  $I$  in the **index-set** of  $A$ ,  
     Set **item**  $I$  of the **ravel-list** of  $A$  to **zero**.  
 ...

- **For form**  $F$ ,  $C$ :
- **For pattern**  $F$ ,  $C$ :

An evaluation sequence phrase used to specify that the actions listed in the consequent  $C$  are to be performed only if the pattern or form  $F$  is the one that caused this evaluation sequence to be selected.

Example:

For form  $A \oplus B$   
     If  $B$  is a **scalar** ...  
 ...  
 For form  $A \ominus B$   
     If ...

- **If**  $A$ ,  $C$ : An evaluation sequence phrase used to specify that the actions listed in the consequent  $C$  are to be performed only if the value of the antecedent  $A$  is **one**.
- **If**  $T$  is an **mc**,  $C$ : For  $T$  a **token** and **mc** a **metaclass**, an evaluation sequence phrase equivalent to “If the **class** of  $T$  is in the **metaclass** **mc**,  $C$ .”
- **Let**  $A$  **stand for**  $B$ : An evaluation sequence phrase used to indicate that the name  $A$  is to be an abbreviation for the phrase  $B$  in subsequent evaluation sequence lines.
- **Otherwise**,  $C$ : An evaluation sequence phrase used to indicate that the actions listed in consequent  $C$  are to be performed only if the antecedent in the immediately preceding **if** phrase was **zero**. If a consequent is an indented paragraph, the immediately preceding **if** statement is the one at the same level of indentation as the **otherwise** phrase.
- **Repeat**: An evaluation sequence phrase used to indicate that the block of text indented after the **repeat** is to be executed repeatedly until a **return** or **signal** phrase is encountered. The end of a repeated block is indicated by the parenthetic remark “(End of repeated block).”