ELKS: THE EIFFEL LIBRARY KERNEL STANDARD

VINTAGE 95

(Also: vintage 98, Revision 0)

Report identification

TR-EI-48/KL: *The Eiffel Library Standard*. (Earlier title: *The Proposed Eiffel Library Kernel Standard*). Prepared for the Nonprofit International Consortium for Eiffel by:

Interactive Software Engineering

ISE Building, second floor, 270 Storke Road

Goleta, CA 93117 USA

Phone 805-685-1006, Fax 805-685-6869

http://eiffel.com, <info@eiffel.com>

Publication history

First published: 14 July 1994. Version 5: 25 September 1994. Version 6: 10 October 1994.

Version 7: 12 January 1995, integrating the result of the votes of the NICE library committee (December 1994) on classes *ANY*, *PLATFORM* and *GENERAL*..

Version 8, 4 June 1995, approved as the vintage 95 library standard.

The present text is version 9, 26 July 1998. It is not really a new version since the technical content is identical to version 8. The purposes of this revision are:

- To provide a PDF version (earlier releases were available on paper and in Postscript).
- To perform text formating changes, taking advantage of color and circumventing a catastrophic bug of Adobe FrameMaker 5.5.
- To update addresses (mostly on the present page).
- Most importantly, to prepare for ELKS 98. As a working document this can thus be considered as revision 0 of ELKS 98.

Authors

The document was written by Bertrand Meyer and benefited from numerous discussions with Michael Schweitzer. The substance (the kernel specification) is the result of a collective effort with contributions from:

Eric Bezault (ISE, Santa Barbara)

Roger Browne (Everything Eiffel, Great Britain)*

Didier Dupont (SOL, Paris)

Fred Hart (Tower Technology, Atlanta)*

Aaron Hillegas (Tower Technology, Atlanta)

Rock Howard (Tower Technology, Austin)*

Paul Johnson (GEC-Marconi, Great Britain)**

Xavier Le Vourch (ISE, Santa Barbara)

David Morgan (ISE, Santa Barbara)

Bertrand Meyer (ISE, Santa Barbara)

Christine Mingins (Monash University, Melbourne), chairman of the NICE library committee

Michael Schweitzer (Swissoft, Göttingen)*

Philippe Stephan (SCENEC, Paris)

Robert Switzer (Swissoft, Göttingen, and University of Göttingen)*

Steve Tynor (Tower Technology, Atlanta)*

Dino Valente (ISE, Santa Barbara)

The influence of the Gustave kernel proposal (authors marked * above) and the GEC-Marconi kernel proposal (author marked ** above) is gratefully acknowledged.

Copyright notice

Copyright © Nonprofit International Consortium for Eiffel (NICE), 1995. Use and duplication of this document for the work of NICE, in conformance with the bylaws and regulations of NICE, or by members of NICE for any purpose, is permitted provided the present copyright notice is included in every copy. Any other use or duplication requires the permission of the copyright holder.

Contents

Contents		3
0	INTRODUCTION	5
1	CONTENTS OF THIS STANDARD	5
	1.1 Definition: this Standard	5
	1.2 Scope of this Standard	6
	1.3 Other documents	6
2	COMPATIBILITY CONDITIONS	6
	2.1 Definitions	6
	Required Classes	6
	Required Flatshort Form	6
	Flatshort Compatibility	6
	Required Ancestry Links	6
	2.2 Kernel compatibility	7
	Definition	7
	2.3 Flatshort Conventions	7
	Definition	7
	2.4 Flatshort Compatibility	8
	Definition	8
3	REQUIRED CLASSES	9
4	REQUIRED ANCESTRY LINKS	10
5	SHORT FORMS OF REQUIRED CLASSES	13
	5.1 Class GENERAL	13
	5.2 Class ANY	15
	5.3 Class <i>COMPARABLE</i>	16

4	CONTENTS
4	CONTENT

8	INDEX	59	
7	APPENDIX B: DIFFERENCES UP TO ELKS 95	56	
	6.6 Eiffel Kernel Supplier requirements	55	
	6.5 Intermediate corrections	55	
	6.4 Yearly schedule	54	
	6.3 Vintages	54	
	6.2 Cycle time	54	
	6.1 Why plan a process?	54	
6	APPENDIX A: THE KERNEL STANDARDIZATION PROCESS	54	
	5.26 Class REAL_REF	53	
	5.25 Class POINTER_REF	52	
	5.24 Class INTEGER_REF	51	
	5.23 Class DOUBLE_REF	50	
	5.22 Class CHARACTER_REF	49	
	5.21 Class BOOLEAN_REF	48	
	5.19 Class ARGUMENTS 5.20 Class PLATFORM	40 47	
	5.18 Class EXCEPTIONS 5.19 Class ARGUMENTS	45 46	
	5.17 Class MEMORY 5.18 Class EXCEPTIONS	44 45	
	5.16 Class STORABLE 5.17 Class MEMORY	43 44	
	5.15 Class FILE 5.16 Class STORABLE	40	
	5.14 Class STD_FILES	39	
	5.13 Class STRING	35	
	5.12 Class ARRAY	33	
	5.11 Class <i>POINTER</i>	32	
	5.10 Class DOUBLE	29	
	5.9 Class <i>REAL</i>	26	
	5.8 Class INTEGER	23	
	5.7 Class CHARACTER	21	
	5.6 Class BOOLEAN	20	
	5.5 Class <i>NUMERIC</i>	18	
	5.4 Class <i>HASHABLE</i>	17	
_			-

0 INTRODUCTION

[This introduction is not part of the Standard.]

0.1

To favor the interoperability between implementations of Eiffel, it is necessary, along with a precise definition of the language, to have a well-defined set of libraries covering needs that are likely to arise in most applications. This library is known in Eiffel as the Kernel Library.

0.2

The present document defines a standard for the Kernel Library. If an Eiffel implementation satisfies this Standard — under the precise definition of *Kernel Compatibility* given in section 2.2 — it will be able to handle properly any Eiffel system whose use of the Kernel Library only assumes the library properties defined in this Standard.

0.3

The Eiffel Library standardization process, as described in Appendix A of the present document, is based on a dynamic view which, in the spirit of Eiffel's own "feature obsolescence" mechanism, recognizes the need to support evolution while preserving the technology investment of Eiffel users. One of the consequences of this dynamic view is to define *vintages* corresponding to successive improvements of the Standard. The present document describes **Vintage 95**, valid for the calendar year 1995.

1 CONTENTS OF THIS STANDARD

1.1 Definition: this Standard

The Eiffel Kernel Library Standard, denoted in the present document by the phrase "this Standard", is made up of the contents of sections 1 to 5 of the present document, with the exception of elements appearing between square brackets [...] which are comments.

[As a result of the preceding definition the following elements are not part of this Standard: section 0, the table of contents, Appendix A in section 6 (the Kernel Library Standardization process), Appendix B in section 7 (list of differences), the Index in section 8, and elements playing a pure typesetting role such as page headers.]

1.2 Scope of this Standard

This Standard defines a number of library-related conditions that an Eiffel implementation must satisfy. These conditions affect a set of classes known as the kernel library. An implementation that satisfies the conditions described in this Standard will be said to be **kernel-compatible**, a phrase that is abbreviated in this Standard as just "compatible".

[In other contexts it may be preferable to use the full phrase, since the compatibility of an Eiffel implementation also involves other aspects, such as language compatibility.]

[The terms "compatibility" and "compatible" may be felt to be less clear than "conformance" and "conformant". The former are used here, however, since talking about conformance might cause confusions with the Eiffel notion of a type conforming to another.]

1.3 Other documents

The phrase *Eiffel: The Language* as used in this Standard refers to the second printing of the book *Eiffel: The Language*, Prentice Hall, 1992, ISBN 0-13-245-925-7.

For the purposes of this Standard, the definition of the Eiffel language is the definition given by *Eiffel: The Language*.

In case of contradictions between the library specifications given by *Eiffel: The Language* and those given in this Standard, the latter shall take precedence.

2 COMPATIBILITY CONDITIONS

2.1 Definitions

2.1.1 Required Classes

In this Standard, the phrase "Required Classes" denotes a set of classes whose names are those listed in section 3.

2.1.2 Required Flatshort Form

In this Standard, the phrase "Required Flatshort Forms" denotes the flatshort forms given for the Required Classes in section 3.

2.1.3 Flatshort Compatibility

In this Standard, a class is said to be Flatshort-Compatible with one of the short forms given in this Standard if it satisfies the conditions given in section 2 of this Standard.

2.1.4 Required Ancestry Links

In this Standard, the expression "Required Ancestry Links" denotes the inheritance links specified in section 4 of this Standard.

[The term "Ancestry" is used rather than "Inheritance" because the required links may be implemented by indirect rather than direct inheritance, except for which must be a direct heir of *GENERAL* as per rule 4.2, given on page 10.]

2.2.1.5

2.2 Kernel compatibility

2.2.1 Definition

An Eiffel implementation will be said to be kernel-compatible if and only if it includes a set of classes satisfying the following five conditions:

- For each of the Required Classes, the implementation includes a class with the same name.
- All the Required Ancestry Links are present between these classes.
- The flatshort form of each one of these classes is Flatshort-Compatible with the corresponding Required Flatshort Form.
- All the dependents of the Required Classes in the implementation are also included in the implementation.
 - None of the features appearing in the Required Flatshort Forms appears in a Rename clause of any of the implementation's Required Classes.

[These conditions allow a kernel-compatible implementation to include inheritance links other than the ones described in this Standard; condition 2.2.1.4 indicates that for any such link the additional proper ancestors must also be provided by the implementors, since the dependents of a class include its parents.]

[Condition 2.2.1.4 guarantees that if a feature name appears in this Standard both in the Flatshort form of a Required Class and in the flatshort form of one of its proper ancestors, it corresponds to the same feature or to a redefinition of it.]

2.3 Flatshort Conventions

2.3.1 Definition

2.3.1.1

2.3.1.2

2.3.1.3

In the process of assessing for Flatshort Compatibility a class \mathcal{C} from a candidate implementation, the following ten conventions, which have been applied to the Required Flatshort Forms as they appear in this Standard, shall be applied:

- No feature shall be included unless it is generally available (as defined in *Eiffel: The Language*, page 100) or is a general creation procedure (as defined in *Eiffel: The Language*, page 285).
- The Creation clause of the flatshort specification shall include the full specification of all general creation procedures of *C*.
- Any feature of C not inherited from GENERAL shall be included in one of the Feature clauses.

[As a consequence of the last two rules the specification of a creation procedure that is also generally exported will appear twice: in the Creation clause and in a Feature clause. Also note that the "features of a class" include inherited as well as immediate features, so that all features inherited from an ancestor other than *GENERAL* must appear in the flatshort form.]

• A feature f from GENERAL shall be included if and only if C redeclares f.

2.3.1.5	• The header comment of any inherited feature coming from a Required Class A and having the same name in C as in A shall end with a line of the form:
	(From <i>A</i> .)
2.3.1.6	• The header comment of any inherited feature coming from a Required Class A and having a name in C different from its name x in A shall end with a line of the form:
	(From x in A .)
	[The comments defined in the last two rules are applicable regardless of whether ${\cal C}$ redeclares the feature.]
2.3.1.7	• If deferred, C shall appear as deferred class .
2.3.1.8	• Any deferred feature of C shall be marked as deferred .
2.3.1.9	• In case of precondition redeclaration, the successive preconditions shall appear as a single Precondition clause, separated by semicolons.
2.3.1.10	• In case of postcondition redeclaration, the successive preconditions shall appear as a single Postcondition clause, separated by and then .

2.4.1 Definition

2.4.1.4

A class appearing in an Eiffel implementation is said to be Flatshort-Compatible with a class of the same name listed in this Standard if and only if any difference that may exist between its flatshort form ic and the flatshort form sc of the corresponding class as it appears in section 5, where both flatshort forms follow the conventions of section 2.3, belongs to one of the following eleven categories:

- A feature that appears in *ic* but not in *sc*, whose Header_comment includes, as its last line, the mention:
 - -- (Feature not in Kernel Library Standard.)
- An invariant clause that appears in *ic* but not in *sc*.
- For a feature that appears in both *ic* and *sc*, a postcondition clause that appears in *ic* but not in *sc*.
 - For a feature that appears in both *ic* and *sc*, a precondition in *sc* that implies the precondition in *ic*, where the implication is readily provable using rules of mathematical logic.
- For a feature that appears in both *ic* and *sc*, a postcondition or invariant clause in *ic* that implies the corresponding clause in *sc*, where the implication is readily provable using rules of mathematical logic.
- A difference between the Tag_mark of an Assertion_clause in *ic* and its counterpart in *sc*.
- For a feature that appears in both *ic* and *sc*, an argument type in *sc* that is different from the corresponding type in *ic* but conforms to it.

3.4

3.5 3.6

3.7

3.10

3.11

3.14

- 2.4.1.8
 For a feature that from the correspondence of the comment of the correspondence of the comment of the correspondence of the corresponden
 - For a feature that appears in both *ic* and *sc*, an argument type in *ic* that is different from the corresponding type in *sc* but conforms to it.
 - For a feature that appears in both *ic* and *sc*, a line that appears in the Header_comment of *ic* but not in that of *sc*.
 - An Index clause that appears in ic but not in sc.
 - A difference regarding the order in which a feature appears in *ic* and *sc*, the Feature_clause to which it belongs, the Header_comment of such a Feature_clause, or the presence in *ic* of a Feature_clause that has no counterpart in *sc*.

[As a consequence of section 2.4.1.11, the division of classes into one Feature_clause or more, and the labels of these clauses, appear in this document for the sole purpose of readability and ease of opdreference, but are not part of this Standard.]

[The goal pursued by the preceding definition is to make sure that an Eiffel system that follows this Standard will be correctly processed by any compatible implementation, without limiting the implementors' freedom to provide more ambitious facilities.]

3 REQUIRED CLASSES

The Required Classes are the following twenty classes [ordered from the general to the specific, as in section 5]:

	specific, as in section 5]:	
3.1	• GENERAL [flatshort form in section 5.1].	
3.2	• ANY [flatshort form in section 5.2].	
3.3	• COMPARABLE [flatshort form in section 5.3].	

- COMTANABLE [natshort form in section 3.5]
- HASHABLE [flatshort form in section 5.4].
 - NUMERIC [flatshort form in section 5.5].
 BOOLEAN [flatshort form in section 5.6].
- *CHARACTER* [flatshort form in section 5.7].
- 3.8 *INTEGER* [flatshort form in section 5.8].
- *REAL* [flatshort form in section 5.9].
 - DOUBLE [flatshort form in section 5.10].
 - *POINTER* [flatshort form in section 5.10].
- ARRAY [flatshort form in section 5.12].
- *STRING* [flatshort form in section 5.13).
 - STD_FILES [flatshort form in section 5.14].
- FILE [flatshort form in section 5.15].
- STORABLE [flatshort form in section 5.16].

3.17	• <i>MEMORY</i> [flatshort form in section 5.17].
3.18	• EXCEPTIONS [flatshort form in section 5.18].
3.19	• ARGUMENTS [flatshort form in section 5.19].
3.20	• <i>PLATFORM</i> [flatshort form in section 5.20].
3.21	• BOOLEAN_REF [flatshort form in section 5.21].
3.22	• CHARACTER_REF [flatshort form in section 5.22].
3.23	• DOUBLE_REF [flatshort form in section 5.23].
3.24	• INTEGER_REF [flatshort form in section 5.24].
3.25	• POINTER_REF [flatshort form in section 5.25].
3.26	• <i>REAL_REF</i> [flatshort form in section 5.26].
	[The classes appear in this section and section and section 5 in the following order: universal classes: deferred classes for basic classes: basic classes: arrays

definition of basic classes.]

4 REQUIRED ANCESTRY LINKS

The following constitute the required ancestry links [ordered alphabetically, after the first rule, by the name of the applicable descendant class]:

and strings; operating system interface; auxiliary reference classes for the

	first rule, by the name of the applicable descendant class]:
4.1	• Every Required Class except GENERAL is a descendant of ANY
4.2	• ANY is an heir of GENERAL.
4.3	• BOOLEAN is a proper descendant of BOOLEAN_REF.
4.4	• BOOLEAN_REF is a proper descendant of HASHABLE.
4.5	• CHARACTER is a proper descendant of CHARACTER_REF.
4.6	• CHARACTER_REF is a proper descendant of COMPARABLE.
4.7	 CHARACTER_REF is a proper descendant of HASHABLE.
4.8	 DOUBLE is a proper descendant of DOUBLE_REF.
4.9	• DOUBLE_REF is a proper descendant of COMPARABLE.
4.10	• DOUBLE_REF is a proper descendant of HASHABLE.
4.11	 DOUBLE_REF is a proper descendant of NUMERIC.
4.12	• FILE is a proper descendant of MEMORY.
4.13	 INTEGER is a proper descendant of INTEGER_REF.
4.14	 INTEGER_REF is a proper descendant of COMPARABLE.
4.15	• INTEGER_REF is a proper descendant of HASHABLE.
4.16	• <i>INTEGER_REF</i> is a proper descendant of <i>NUMERIC</i> .

4.24

• STRING is a proper descendant of HASHABLE.

[4.1 follows from *Eiffel: The Language*; the language description is considered to be amended in such a way that *PLATFORM* is a class without privileges, to be inherited explicitly by classes which need access to its features.]

5 SHORT FORMS OF REQUIRED CLASSES

5.1 Class GENERAL

indexing

description: "Platform-independent universal properties. This class is an ancestor to all developer-written classes."

class interface

GENERAL

feature -- Access

generating_type: STRING

- -- Name of current object's generating type
- -- (type of which it is a direct instance)

generator: STRING

- -- Name of current object's generating class
- -- (base class of the type of which it is a direct instance)

id_object (id: INTEGER): ANY

- -- Object for which object_id has returned id;
- -- void if none.

object_id: INTEGER

- -- Value identifying current object uniquely;
- -- meaningful only for reference types.

stripped (other: GENERAL): like other

- -- New object with fields copied from current object.
- -- but limited to attributes of type of other.

require

conformance: conforms_to (other)

ensure

stripped_to_other: Result.same_type (other)

feature -- Status report

frozen conforms_to (other: GENERAL): BOOLEAN

- -- Does type of current object conform to type -- of *other* (as per *Eiffel: The Language*,
- chapter13)?

require

other_not_void: other /= Void

```
frozen same_type (other: GENERAL): BOOLEAN
```

-- Is type of current object identical to type of *other*?

require

other_not_void: other /= Void

ensure

definition: Result = (conforms_to (other) and other • conforms_to (Current))

feature -- Comparison

frozen deep_equal (some: GENERAL; other: **like** some): BOOLEAN

- -- Are some and other either both void
- -- or attached to isomorphic object structures?

ensure

shallow_implies_deep: standard_equal (some, other) implies Result

same_type: Result implies some .same_type
 (other)

symmetric: Result implies deep_equal (other, some)

frozen equal (some: GENERAL; other: **like** some): BOOLEAN

- -- Are some and other either both void or attached
- -- to objects considered equal?

ensure

definition: Result = (some = Void and other = Void) or else ((some /= Void and other /= Void) and then some • is_equal (other))

is_equal (other: like Current): BOOLEAN

- -- Is other attached to an object considered equal
- -- to current object?

require

other_not_void: other /= Void

ensure

consistent: standard_is_equal (other) implies
Result

same_type: Result implies same_type (other)
symmetric: Result implies other.is_equal
(Current)

§5.1 CLASS GENERAL 14

frozen standard_equal (some: GENERAL; other: like	frozen standard_copy (other: like Current)
some): BOOLEAN	Copy every field of <i>other</i> onto corresponding field
Are <i>some</i> and <i>other</i> either both void or attached to	of current object.
field-by-field identical objects of the same type?	require
Always uses the default object comparison	other_not_void: other /= Void;
criterion.	type_identity: same_type (other)
ensure	
definition: Result = (some = Void and other = Void) or else ((some /= Void and other /= Void)	ensure
and then some standard_is_equal (other))	is_standard_equal: standard_is_equal (other)
	feature Basic operations
frozen standard_is_equal (other: like Current): BOOLEAN	frozen default: like Current
Is <i>other</i> attached to an object of the same type as	Default value of current type
current object, and field-by-field identical to it?	frozen default_pointer: POINTER
require	Default value of type POINTER
other_not_void: other /= Void	(Avoid the need to write $p \cdot default$ for some p
ensure	of type POINTER.)
<pre>same_type: Result implies same_type (other)</pre>	ensure
symmetric: Result implies other • standard_is_	$Result = Result \cdot default$
equal (Current)	default_rescue
feature Duplication	 Handle exception if no Rescue clause. (Default: do nothing.)
frozen clone (other: GENERAL): like other	frozen do_nothing
 Void if <i>other</i> is void; otherwise new object equal to <i>other</i>.	Execute a null action.
ensure	frozen Void: NONE
equal: equal (Result, other)	Void reference
	feature Output
copy (other: like Current) Update current object using fields of object	io: STD_FILES
attached	Handle to standard file setup
to <i>other</i> , so as to yield equal objects.	out: STRING
require	New string containing terse printable
other_not_void: other /= Void;	representation
type_identity: same_type (other)	of current object
ensure	print (some: GENERAL)
is_equal: is_equal (other)	Write terse external representation of <i>some</i> on standard output.
frozen deep_clone (other: GENERAL): like other Void if other is void: otherwise, new object	frozen tagged_out: STRING
structure	New string containing printable representation
recursively duplicated from the one attached to	of
other	current object, each field preceded by its attribute
ensure	name, a colon and a space.
deep_equal: deep_equal (other, Result)	invariant
<pre>frozen standard_clone (other: GENERAL): like other Void if other is void; otherwise new object</pre>	reflexive_equality: standard_is_equal (Current)
field-by-field identical to <i>other</i> .	reflexive_conformance: conforms_to (Current)
Always uses the default copying semantics.	
ensure	involutive_object_id: id_object (object_id) = Current
equal: standard equal (Result, other)	end

5.2 Class ANY

indexing

description: "Project-wide universal properties. This class is an ancestor to all developer-written classes. ANY inherits from GENERAL and may be customized for individual projects or teams."

class interface

ANY

end

§5.3 CLASS COMPARABLE 16

5.3 Class COMPARABLE

```
indexing
                                                                       is_equal (other: like Current): BOOLEAN
                                                                               -- Is other attached to an object considered equal
   description: "Objects that may be compared according
                                                                               -- to current object?
      to a total order relation"
                                                                               -- (Redefined from GENERAL.)
                                                                           require
   note: "The basic operation is "<" (less than); others are
                                                                               other_not_void: other /= Void
      defined in terms of this operation and is equal."
deferred class interface
                                                                               symmetric: Result implies other.is_equal (Current)
    COMPARABLE
                                                                               consistent: standard_is_equal (other) implies
                                                                                 Result
feature -- Comparison
                                                                               trichotomy: Result = (not (Current < other) and
   infix "<" (other: like Current): BOOLEAN
                                                                                 not (other < Current))
           -- Is current object less than other?
                                                                       max (other: like Current): like Current
       require
                                                                               -- The greater of current object and other
                                                                           require
           other_exists: other /= Void
                                                                               other_exists: other /= Void
       deferred
                                                                           ensure
       ensure
                                                                               current_if_not_smaller: (Current >= other)
           asymmetric: Result implies not (other < Current)
                                                                                 implies (Result = Current)
                                                                               other_if_smaller: (Current < other) implies</pre>
   infix "<=" (other: like Current): BOOLEAN
                                                                                 (Result = other)
           -- Is current object less than or equal to other?
                                                                       min (other: like Current): like Current
       require
                                                                               -- The smaller of current object and other
           other_exists: other /= Void
                                                                           require
       ensure
                                                                               other_exists: other /= Void
           definition: Result = (Current < other) or is_equal
              (other)
                                                                               current_if_not_greater: (Current <= other)</pre>
                                                                                 implies (Result = Current)
   infix ">=" (other: like Current): BOOLEAN
                                                                               other_if_greater: (Current > other) implies (Result
           -- Is current object greater than or equal to other?
                                                                                 = other)
       require
                                                                       three_way_comparison (other: like Current): INTEGER)
           other_exists: other /= Void
                                                                               -- If current object equal to other, 0; if smaller,
                                                                               ---1; if greater, 1.
       ensure
                                                                           require
           definition: Result = (other <= Current)
                                                                               other_exists: other /= Void
   infix ">" (other: like Current): BOOLEAN
           -- Is current object greater than other?
                                                                               equal\_zero: (Result = 0) = is\_equal (other)
                                                                              smaller\_negative: (Result = -1) = (Current < other)
       require
                                                                               greater\_positive: (Result = 1) = (Current > other)
           other_exists: other /= Void
                                                                   invariant
       ensure
                                                                       irreflexive_comparison: not (Current < Current)
           definition: Result = (other < Current)
                                                                   end
```

5.4 Class HASHABLE

```
indexing
  description: "Values that may be hashed into an integer
  index, for use as keys in hash tables."

deferred class interface
  HASHABLE

feature -- Access
  hash_code: INTEGER
    -- Hash code value
  deferred
  ensure
    good_hash_value: Result >= 0
end
```

§5.5 CLASS NUMERIC 18

5.5 Class NUMERIC

```
indexing
                                                                      infix "-" (other: like Current): like Current
                                                                              -- Result of subtracting other
   description: "Objects to which numerical operations are
                                                                          require
      applicable"
                                                                              other_exists: other /= Void
   note: "The model is that of a commutative ring."
                                                                          deferred
deferred class interface
                                                                          ensure
   NUMERIC
                                                                              result_exists: Result /= Void
feature -- Access
                                                                      infix "*" (other: like Current): like Current
                                                                              -- Product by other
    one: like Current
           -- Neutral element for "*" and "/"
                                                                          require
                                                                              other_exists: other /= Void
       deferred
                                                                          deferred
       ensure
                                                                          ensure
           Result_exists: Result /= Void
                                                                              result_exists: Result /= Void
    zero: like Current
                                                                      infix "/" (other: like Current): like Current
           -- Neutral element for "+" and "-"
                                                                              -- Division by other
       deferred
                                                                          require
       ensure
                                                                              other_exists: other /= Void
           Result_exists: Result /= Void
                                                                              good_divisor: divisible (other)
feature -- Status report
                                                                          deferred
   divisible (other: like Current): BOOLEAN
                                                                          ensure
           -- May current object be divided by other?
                                                                              result_exists: Result /= Void
       require
                                                                      infix "^" (other: NUMERIC): NUMERIC
                                                                              -- Current object to the power other
           other_exists: other /= Void
       deferred
                                                                              other_exists: other /= Void
    exponentiable (other: NUMERIC): BOOLEAN
                                                                              good_exponent: exponentiable (other)
           -- May current object be elevated to the power other?
                                                                          deferred
       require
                                                                          ensure
           other_exists: other /= Void
                                                                              result_exists: Result /= Void
       deferred
                                                                       prefix "+": like Current
feature -- Basic operations
                                                                              -- Unary plus
   infix "+" (other: like Current): like Current
                                                                          deferred
           -- Sum with other (commutative).
                                                                          ensure
       require
                                                                              result_exists: Result /= Void
           other_exists: other /= Void
                                                                       prefix "-": like Current
                                                                              -- Unary minus
       deferred
                                                                          deferred
       ensure
           result_exists: Result /= Void
                                                                          ensure
           commutative: equal (Result, other + Current)
                                                                              result_exists: Result /= Void
```

invariant

```
neutral_addition: equal (Current + zero, Current)
self_subtraction: equal (Current - Current, zero)
neutral_multiplication: equal (Current * one, Current)
self_division: divisible (Current) implies equal (Current / Current, one)
end
```

§5.6 CLASS BOOLEAN 20

5.6 Class *BOOLEAN*

```
indexing
                                                                                                                                                                                                                     infix "or" (other: BOOLEAN): BOOLEAN
                                                                                                                                                                                                                                            -- Boolean disjunction with other
           description: "Truth values, with the boolean
                  operations"
                                                                                                                                                                                                                                 require
expanded class interface
                                                                                                                                                                                                                                            other_exists: other /= Void
            BOOLEAN
                                                                                                                                                                                                                                 ensure
feature -- Access
                                                                                                                                                                                                                                            Result_exists: Result /= Void
            hash code: INTEGER
                                                                                                                                                                                                                                            de\_morgan: Result = not (not Current and (not Current
                                   -- Hash code value
                                                                                                                                                                                                                                                    other))
                                  -- (From HASHABLE.)
                                                                                                                                                                                                                                            commutative: Result = (other or Current)
                       ensure
                                  good_hash_value: Result >= 0
                                                                                                                                                                                                                                            consistent_with_semi_strict: Result implies
                                                                                                                                                                                                                                                    (Current or else other)
feature -- Basic operations
                                                                                                                                                                                                                     infix "or else" (other: BOOLEAN): BOOLEAN
           infix "and" (other: BOOLEAN): BOOLEAN
                                   -- Boolean conjunction with other
                                                                                                                                                                                                                                             -- Boolean semi-strict disjunction with other
                      require
                                                                                                                                                                                                                                require
                                   other_exists: other /= Void
                                                                                                                                                                                                                                            other_exists: other /= Void
                       ensure
                                                                                                                                                                                                                                 ensure
                                  Result_exists: Result /= Void
                                 de\_morgan: Result = not (not Current or (n
                                                                                                                                                                                                                                            Result | exists: Result |= Void
                                                                                                                                                                                                                                            de_morgan: Result = not (not Current and then
                                  commutative: Result = (other and Current)
                                                                                                                                                                                                                                                    (not other))
                                  consistent_with_semi_strict: Result implies
                                                                                                                                                                                                                      infix "xor" (other: BOOLEAN): BOOLEAN
                                         (Current and then other)
                                                                                                                                                                                                                                             -- Boolean exclusive or with other
          infix "and then" (other: BOOLEAN): BOOLEAN
                                   -- Boolean semi-strict conjunction with other
                                                                                                                                                                                                                                 require
                       require
                                                                                                                                                                                                                                            other_exists: other /= Void
                                  other_exists: other /= Void
                                                                                                                                                                                                                                 ensure
                       ensure
                                                                                                                                                                                                                                            definition: Result = ((Current or other) and not
                                  Result_exists: Result /= Void
                                                                                                                                                                                                                                                    (Current and other))
                                  de\_morgan: Result = \mathbf{not} (\mathbf{not} \ Current \ \mathbf{or} \ \mathbf{else} (\mathbf{not} \ 
                                         other))
                                                                                                                                                                                                          feature -- Output
           infix "implies" (other: BOOLEAN): BOOLEAN
                                                                                                                                                                                                                     out: STRING
                                   -- Boolean implication of other
                                                                                                                                                                                                                                             -- Printable representation of boolean
                                  -- (semi-strict)
                       require
                                                                                                                                                                                                           invariant
                                  other_exists: other /= Void
                                                                                                                                                                                                                      involutive_negation: is_equal (not (not Current))
                       ensure
                                                                                                                                                                                                                     non_contradiction: not (Current and (not Current))
                                  definition: Result = (not Current or else other)
                                                                                                                                                                                                                      completeness: Current or (not Current)
           prefix "not": BOOLEAN
                                   -- Negation.
                                                                                                                                                                                                           end
```

5.7 Class CHARACTER

```
infix ">" (other: like Current): BOOLEAN
indexing
                                                                            -- Is current object greater than other?
   description: "Characters, with comparison operations
                                                                            -- (From COMPARABLE.)
      and an ASCII code"
                                                                         require
expanded class interface
                                                                            other_exists: other /= Void
   CHARACTER
                                                                         ensure
feature -- Access
                                                                            definition: Result = (other < Current)
   code: INTEGER
                                                                     max (other: like Current): like Current
           -- Associated integer value
                                                                            -- The greater of current object and other
   hash code: INTEGER
                                                                            -- (From COMPARABLE.)
           -- Hash code value
                                                                         require
           -- (From HASHABLE.)
                                                                            other_exists: other /= Void
       ensure
           good\_hash\_value: Result >= 0
                                                                            current_if_not_smaller: (Current >= other)
feature -- Comparison
                                                                              implies (Result = Current)
   infix "<" (other: like Current): BOOLEAN
                                                                            other_if_smaller: (Current < other) implies
           -- Is other greater than current character?
                                                                              (Result = other)
           -- (From COMPARABLE.)
                                                                     min (other: like Current): like Current
       require
                                                                            -- The smaller of current object and other
           other_exists: other /= Void
                                                                            -- (From COMPARABLE.)
       ensure
                                                                         require
           asymmetric: Result implies not (other < Current)
                                                                            other_exists: other /= Void
   infix "<=" (other:like Current): BOOLEAN</pre>
           -- Is current character less than or equal to other?
                                                                            current_if_not_greater: (Current <= other)</pre>
           -- (From COMPARABLE.)
                                                                              implies (Result = Current)
       require
                                                                            other_if_greater: (Current > other) implies
           other_exists: other /= Void
                                                                              (Result = other)
                                                                     three_way_comparison (other: like Current): INTEGER
       ensure
                                                                            -- If current object equal to other, 0; if smaller,
           definition: Result = (Current < other) or is_equal
                                                                            ---1; if greater, 1.
                                                                            -- (From COMPARABLE.)
   infix ">=" (other: like Current): BOOLEAN
                                                                         require
           -- Is current object greater than or equal to other?
                                                                            other_exists: other /= Void
           -- (From COMPARABLE.)
       require
                                                                         ensure
           other_exists: other /= Void
                                                                            equal\_zero: (Result = 0) = is\_equal (other)
                                                                            smaller: (Result = -1) = Current < other
       ensure
                                                                            greater\_positive: (Result = 1) = Current > other
           definition: Result = (other <= Current)
```

§5.7 CLASS CHARACTER 22

```
feature -- Output

out: STRING

-- Printable representation of character
-- (From GENERAL.)

invariant

irreflexive_comparison: not (Current < Current)

end
```

5.8 Class INTEGER

```
indexing
                                                                     infix "<=" (other: like Current): BOOLEAN</pre>
                                                                             -- Is current object less than or equal to other?
   description: "Integer values"
                                                                             -- (From COMPARABLE.)
                                                                         require
expanded class interface
                                                                             other_exists: other /= Void
   INTEGER
                                                                         ensure
feature -- Access
                                                                             definition: Result = (Current < other) or is_equal
   hash_code: INTEGER is
                                                                     infix ">=" (other: like Current): BOOLEAN
           -- Hash code value
                                                                             -- Is current object greater than or equal to other?
           -- (From HASHABLE.)
                                                                             -- (From COMPARABLE.)
           good_hash_value: Result >= 0
                                                                             other_exists: other /= Void
   one: like Current
                                                                         ensure
                                                                             definition: Result = (other <= Current)
           -- Neutral element for "*" and "/"
           -- (From NUMERIC.)
                                                                     infix ">" (other: like Current): BOOLEAN
                                                                             -- Is current object greater than other?
       ensure
                                                                             -- (From COMPARABLE.)
           Result_exists: Result /= Void
                                                                         require
           value: Result = 1
                                                                             other_exists: other /= Void
   sign: INTEGER
           -- Sign value (0, -1 \text{ or } 1)
                                                                             definition: Result = (other < Current)
       ensure
                                                                     max (other: like Current): like Current
                                                                             -- The greater of current object and other
           three_way: Result = three_way_comparison
                                                                             -- (From COMPARABLE.)
             (zero)
                                                                         require
   zero: like Current
                                                                             other_exists: other /= Void
           -- Neutral element for "+" and "-"
                                                                         ensure
           -- (From NUMERIC.)
                                                                             current_if_not_smaller: (Current >= other)
       ensure
                                                                               implies (Result = Current)
           Result_exists: Result /= Void
                                                                             other_if_smaller: (Current < other) implies
                                                                               (Result = other)
           value: Result = 0
                                                                     min (other: like Current): like Current
feature -- Comparison
                                                                             -- The smaller of current object and other
                                                                             -- (From COMPARABLE.)
   infix "<" (other: like Current): BOOLEAN
                                                                         require
           -- Is other greater than current integer?
                                                                             other_exists: other /= Void
           -- (From COMPARABLE.)
                                                                         ensure
       require
                                                                             current_if_not_greater: (Current <= other)</pre>
           other_exists: other /= Void
                                                                               implies (Result = Current)
       ensure
                                                                             other_if_greater: (Current > other) implies
           asymmetric: Result implies not (other < Current)
                                                                               (Result = other)
```

§5.8 CLASS INTEGER 24

three_way_comparison (other: like Current): INTEGER	<pre>infix "+" (other: like Current): like Current</pre>	
If current object equal to <i>other</i> , 0; if smaller,1; if greater, 1.	Sum with other (From NUMERIC.)	
(From COMPARABLE.)	require	
require	other_exists: other /= Void	
other exists: other /= Void	ensure	
omer_exists.omer/= void	result_exists: Result /= Void	
ensure	commutative: equal (Result, other + Current)	
$equal_zero: (Result = 0) = is_equal (other)$	infix "-" (other: like Current): like Current	
smaller: (Result = 1) = Current < other	Result of subtracting other	
$greater_positive: (Result = -1) = Current > other$	(From NUMERIC.)	
feature Status report	require other_exists: other /= Void	
•		
divisible (other: like Current): BOOLEAN	ensure result exists: Result /= Void	
May current object be divided by <i>other</i> ?	-	
(From NUMERIC.)	infix "/" (other: like Current): DOUBLE	
require	Division by other	
other_exists: other /= Void	require other_exists: other /= Void	
ensure	good_divisor: divisible (other)	
value: $Result = (other /= 0)$		
, ,	<pre>ensure result_exists: Result /= Void</pre>	
exponentiable (other: NUMERIC): BOOLEAN		
May current object be elevated to the power other?	infix "//" (other: like Current): like Current Integer division of Current by other	
(From NUMERIC.)	(From "/" in NUMERIC.)	
	require	
require	other_exists: other /= Void	
other_exists: other /= Void	<pre>good_divisor: divisible (other)</pre>	
ensure	ensure	
safe_values: (other • conforms_to (Current) or	result_exists: divisible (other)	
$(other.conforms_to(0.0) $ and $(Current>=0)))$	<pre>infix "\\" (other: like Current): like Current</pre>	
implies Result	Remainder of the integer division of Current by	
feature Basic operations	other	
abs: like Current	require	
Absolute value	other_exists: other /= Void	
	good_divisor: divisible (other)	
ensure	ensure	
non_negative: Result >= 0	result_exists: Result /= Void	
same_absolute_value: (Result = Current) or	infix "^" (other: NUMERIC): DOUBLE	
(Result = -Current)	Integer power of Current by <i>other</i> (From <i>NUMERIC</i> .)	
infix "*" (other: like Current): like Current	require	
Product by <i>other</i>	other_exists: other /= Void	
(From NUMERIC.)	good_exponent: exponentiable (other)	
require	ensure	
other_exists: other /= Void	result_exists: Result /= Void	
_	_	

```
prefix "+": like Current
           -- Unary plus
-- (From NUMERIC.)
       ensure
           result_exists: Result /= Void
   prefix "-": like Current
           -- Unary minus
-- (From NUMERIC.)
       ensure
           result_exists: Result /= Void
feature -- Output
   out: STRING
           -- Printable representation of current object
           -- (From GENERAL.)
invariant
   irreflexive_comparison: not (Current < Current)</pre>
   neutral_addition: equal (Current + zero, Current)
   self_subtraction: equal (Current - Current, zero)
   neutral_multiplication: equal (Current * one, Current)
   self_division: divisible (Current) implies equal (Current /
      Current, one)
   sign_times_abs: equal (sign*abs, Current)
end
```

§5.9 CLASS REAL 26

5.9 Class REAL

```
indexing
                                                                     infix ">=" (other: like Current): BOOLEAN
   description: "Real values, single precision"
                                                                             -- Is current object greater than or equal to other?
expanded class interface
                                                                             -- (From COMPARABLE.)
   REAL
                                                                         require
feature -- Access
                                                                             other_exists: other /= Void
   hash code: INTEGER
                                                                         ensure
           -- Hash code value
           -- (From HASHABLE.)
                                                                             definition: Result = (other <= Current)
                                                                     infix ">" (other: like Current): BOOLEAN
           good_hash_value: Result >= 0
                                                                             -- Is current object greater than other?
   one: like Current
                                                                             -- (From COMPARABLE.)
           -- Neutral element for "*" and "/"
           -- (From NUMERIC.)
                                                                         require
       ensure
                                                                             other_exists: other /= Void
           Result_exists: Result /= Void
           value: Result = 1.0
                                                                         ensure
   sign: INTEGER
                                                                             definition: Result = (other < Current)
           -- Sign value (0, -1 \text{ or } 1)
                                                                     max (other: like Current): like Current
       ensure
                                                                             -- The greater of current object and other
           three_way: Result = three_way_comparison (zero)
                                                                             -- (From COMPARABLE.)
   zero: like Current
           -- Neutral element for "+" and "-"
                                                                         require
           -- (From NUMERIC.)
                                                                             other_exists: other /= Void
       ensure
           Result_exists: Result /= Void
                                                                         ensure
           value: Result = 0.0
                                                                             current_if_not_smaller: (Current >= other)
feature -- Comparison
                                                                               implies (Result = Current)
   infix "<" (other: like Current): BOOLEAN
                                                                             other_if_smaller: (Current < other) implies</pre>
           -- Is other greater than current real?
                                                                               (Result = other)
           -- (From COMPARABLE.)
       require
                                                                     min (other: like Current): like Current
           other_exists: other /= Void
                                                                             -- The smaller of current object and other
       ensure
                                                                             -- (From COMPARABLE.)
           asymmetric: Result implies not (other < Current)
                                                                         require
   infix "<=" (other: like Current): BOOLEAN</pre>
           -- Is current object less than or equal to other?
                                                                             other_exists: other /= Void
           -- (From COMPARABLE.)
                                                                         ensure
       require
           other_exists: other /= Void
                                                                             current_if_not_greater: (Current <= other)
                                                                               implies (Result = Current)
       ensure
           definition: Result = (Current < other) or is_equal
                                                                             other_if_greater: (Current > other) implies (Result
             (other)
                                                                               = other)
```

```
three_way_comparison (other: like Current): INTEGER
                                                                      truncated_to_integer: INTEGER
           -- If current object equal to other, 0; if smaller,
                                                                              -- Integer part (same sign, largest absolute
           -- -1; if greater, 1.
                                                                              -- value no greater than current object's)
           -- (From COMPARABLE.)
                                                                   feature -- Basic operations
       require
           other_exists: other /= Void
                                                                      abs: like Current
                                                                              -- Absolute value
       ensure
           equal\_zero: (Result = 0) = is\_equal (other)
                                                                          ensure
           smaller: (Result = -1) = Current < other
                                                                              non_negative: Result >= 0
           greater\_positive: (Result = I) = Current > other
                                                                              same_absolute_value: (Result = Current) or
feature -- Status report
                                                                                (Result = -Current)
                                                                      infix "*" (other: like Current): like Current
   divisible (other: like Current): BOOLEAN
           -- May current object be divided by other?
                                                                              -- Product by other
           -- (From NUMERIC.)
                                                                              -- (From NUMERIC.)
       require
                                                                          require
           other_exists: other /= Void
                                                                              other_exists: other /= Void
       ensure
                                                                          ensure
          not\_exact\_zero: Result implies (other \neq 0.0)
                                                                              result_exists: Result /= Void
   exponentiable (other: NUMERIC): BOOLEAN
                                                                      infix "+" (other: like Current): like Current
           -- May current object be elevated to the power
           other?
                                                                              -- Sum with other
           -- (From NUMERIC.)
                                                                              -- (From NUMERIC.)
       require
                                                                          require
           other_exists: other /= Void
                                                                              other_exists: other /= Void
       ensure
                                                                          ensure
           safe_values: (other • conforms_to (0) or
             (other conforms_to (Current) and (Current>=
                                                                              result exists: Result /= Void
             (0.0)) implies Result
                                                                              commutative: equal (Result, other + Current)
feature -- Conversion
                                                                      infix "-" (other: like Current): like Current
   ceiling: INTEGER
                                                                              -- Result of subtracting other
           -- Smallest integral value no smaller than current
                                                                              -- (From NUMERIC.)
           object
                                                                          require
       ensure
                                                                              other_exists: other /= Void
           result_no_smaller: Result >= Current
          close_enough: Result - Current < one
                                                                          ensure
                                                                              result_exists: Result /= Void
   floor: INTEGER
           -- Greatest integral value no greater than current
                                                                      infix "/" (other: like Current): like Current
           object
                                                                              -- Division by other
       ensure
                                                                              -- (From NUMERIC.)
           result_no_greater: Result <= Current
                                                                          require
           close_enough: Current - Result < one
                                                                              other_exists: other /= Void
   rounded: INTEGER
                                                                              good_divisor: divisible (other)
           -- Rounded integral value
                                                                          ensure
       ensure
           definition: Result = sign * ((abs + 0.5).floor)
                                                                              result_exists: Result /= Void
```

§5.9 CLASS REAL 28

```
infix "^" (other: NUMERIC): DOUBLE
           -- Current real to the power other
           -- (From NUMERIC.)
       require
           other_exists: other /= Void
           good_exponent: exponentiable (other)
       ensure
           result_exists: Result /= Void
   prefix "+": like Current
           -- Unary plus
           -- (From NUMERIC.)
       ensure
           result_exists: Result /= Void
   prefix "-": like Current
           -- Unary minus
           -- (From NUMERIC.)
       ensure
           result_exists: Result /= Void
feature -- Output
   out: STRING
           -- Printable representation of real value
           -- (From GENERAL.)
invariant
   irreflexive_comparison: not (Current < Current)</pre>
   neutral_addition: equal (Current + zero, Current)
   self_subtraction: equal (Current - Current, zero)
   neutral_multiplication: equal (Current * one, Current)
   self_division: divisible (Current) implies equal (Current /
      Current, one)
   sign_times_abs: equal (sign*abs, Current)
end
```

5.10 Class DOUBLE

```
indexing
                                                                     infix "<=" (other: like Current): BOOLEAN</pre>
                                                                            -- Is current object less than or equal to other?
   description: "Real values, double precision"
                                                                            -- (From COMPARABLE.)
expanded class interface
                                                                         require
                                                                            other_exists: other /= Void
   DOUBLE
                                                                         ensure
feature -- Access
                                                                            definition: Result = (Current < other) or is_equal
   hash_code: INTEGER
                                                                     infix ">=" (other: like Current): BOOLEAN
           -- Hash code value
                                                                            -- Is current object greater than or equal to other?
           -- (From HASHABLE.)
                                                                            -- (From COMPARABLE.)
       ensure
                                                                         require
           good\_hash\_value: Result >= 0
                                                                            other_exists: other /= Void
   one: like Current
           -- Neutral element for "*" and "/"
                                                                            definition: Result = (other <= Current)
           -- (From NUMERIC.)
                                                                     infix ">" (other: like Current): BOOLEAN
                                                                            -- Is current object greater than other?
       ensure
                                                                            -- (From COMPARABLE.)
           Result_exists: Result /= Void
                                                                         require
           value: Result = 1.0
                                                                            other_exists: other /= Void
   sign: INTEGER
           -- Sign value (0, -1 or 1)
                                                                            definition: Result = (other < Current)
       ensure
                                                                     max (other: like Current): like Current
                                                                            -- The greater of current object and other
           three_way: Result = three_way_comparison
                                                                            -- (From COMPARABLE.)
             (zero)
                                                                         require
   zero: like Current
                                                                            other_exists: other /= Void
           -- Neutral element for "+" and "-"
                                                                         ensure
           -- (From NUMERIC.)
                                                                            current_if_not_smaller: (Current >= other)
       ensure
                                                                              implies (Result = Current)
                                                                            other_if_smaller: (Current < other) implies
           Result_exists: Result /= Void
                                                                              (Result = other)
           value: Result = 0.0
                                                                     min (other: like Current): like Current
feature -- Comparison
                                                                            -- The smaller of current object and other
   infix "<" (other: like Current): BOOLEAN
                                                                            -- (From COMPARABLE.)
           -- Is other greater than current double?
                                                                         require
           -- (From COMPARABLE.)
                                                                            other_exists: other /= Void
       require
                                                                         ensure
                                                                            current_if_not_greater: (Current <= other)</pre>
           other exists: other /= Void
                                                                              implies (Result = Current)
       ensure
                                                                            other_if_greater: (Current > other) implies
           asymmetric: Result implies not (other < Current)
                                                                              (Result = other)
```

§5.10 CLASS DOUBLE 30

```
three_way_comparison (other: like Current): INTEGER
                                                                       truncated_to_integer: INTEGER
           -- If current object equal to other, 0; if smaller,
                                                                              -- Integer part (same sign, largest absolute
           ---1; if greater, 1.
                                                                              -- value no greater than current object's)
       require
                                                                       truncated_to_real: REAL
           other_exists: other /= Void
                                                                              -- Real part (same sign, largest absolute
           -- (From COMPARABLE.)
                                                                              -- value no greater than current object's)
       ensure
                                                                   feature -- Basic operations
           equal\_zero: (Result = 0) = is\_equal (other)
                                                                       abs: like Current
           smaller: (Result = -1) = Current < other
                                                                              -- Absolute value
           greater\_positive: (Result = 1) = Current > other
                                                                          ensure
feature -- Status report
                                                                              non negative: Result \geq 0
   divisible (other: like Current): BOOLEAN
                                                                              same_absolute_value: (Result = Current) or
           -- May current object be divided by other?
                                                                                (Result = -Current)
           -- (From NUMERIC.)
                                                                      infix "*" (other: like Current): like Current
       require
                                                                              -- Product by other
           other_exists: other /= Void
                                                                              -- (From NUMERIC.)
       ensure
                                                                          require
           not\_exact\_zero: Result implies (other \neq 0.0)
                                                                              other_exists: other /= Void
   exponentiable (other: NUMERIC): BOOLEAN
                                                                          ensure
           -- May current object be elevated to the power
                                                                              result_exists: Result /= Void
           other?
           -- (From NUMERIC.)
                                                                       infix "+" (other: like Current): like Current
       require
                                                                              -- Sum with other
                                                                              -- (From NUMERIC.)
           other_exists: other /= Void
                                                                          require
       ensure
                                                                              other_exists: other /= Void
           safe_values: (other • conforms_to (0) or
             (other • conforms_to (Current) and (Current >=
                                                                          ensure
             (0.0))) implies Result
                                                                              result exists: Result /= Void
feature -- Conversion
                                                                              commutative: equal (Result, other + Current)
   ceiling: INTEGER
                                                                       infix "-" (other: like Current): like Current
           -- Smallest integral value no smaller than current
                                                                              -- Result of subtracting other
           object
                                                                              -- (From NUMERIC.)
       ensure
                                                                          require
           result_no_smaller: Result >= Current
                                                                              other_exists: other /= Void
           close_enough: Result - Current < one
                                                                          ensure
   floor: INTEGER
                                                                              result_exists: Result /= Void
           -- Greatest integral value no greater than current
           object
                                                                      infix "/" (other: like Current): like Current
                                                                              -- Division by other
       ensure
                                                                              -- (From NUMERIC.)
           result_no_greater: Result <= Current
           close_enough: Current - Result < one
                                                                              other_exists: other /= Void
   rounded: INTEGER
                                                                              good_divisor: divisible (other)
           -- Rounded integral value
       ensure
                                                                          ensure
           definition: Result = sign * ((abs + 0.5).floor)
                                                                              result_exists: Result /= Void
```

```
infix "^" (other: like Current): like Current
           -- Current double to the power other
           -- (From NUMERIC.)
       require
           other_exists: other /= Void
           good_exponent: exponentiable (other)
       ensure
           result_exists: Result /= Void
   prefix "+": like Current
           -- Unary plus
           -- (From NUMERIC.)
       ensure
           result_exists: Result /= Void
   prefix "-": like Current
           -- Unary minus
           -- (From NUMERIC.)
       ensure
           result_exists: Result /= Void
feature -- Output
   out: STRING
           -- Printable representation of double value
           -- (From GENERAL.)
invariant
   irreflexive_comparison: not (Current < Current)</pre>
   neutral_addition: equal (Current + zero, Current)
   self_subtraction: equal (Current - Current, zero)
   neutral_multiplication: equal (Current * one, Current)
   self_division: divisible (Current) implies equal (Current/
     Current, one)
   sign_times_abs: equal (sign*abs, Current)
end
```

§5.11 CLASS POINTER 32

5.11 Class POINTER

5.12 Class ARRAY

```
frozen item (i: INTEGER): G
indexing
                                                                              -- Entry at index i, if in index interval
    description: "Sequences of values, all of the same type
      or of a conforming one, accessible through integer
      indices in a contiguous interval"
                                                                              good_key: valid_index (i)
class interface
                                                                      frozen infix "@" (i: INTEGER): G
                                                                              -- Entry at index i, if in index interval
   ARRAY[G]
                                                                          require
creation
                                                                              good_key: valid_index (i)
   make (minindex, maxindex: INTEGER)
                                                                  feature -- Measurement
           -- Allocate array: set index interval to
           -- minindex .. maxindex; set all values to default.
                                                                      count: INTEGER
           -- (Make array empty if minindex > maxindex.)
                                                                              -- Number of available indices
       ensure
                                                                      lower: INTEGER
           no_count: (minindex > maxindex) implies (count =
                                                                              -- Minimum index
                                                                      upper: INTEGER
           count_constraint: (minindex <= maxindex)</pre>
             implies (count = maxindex - minindex + 1)
                                                                              -- Maximum index
                                                                  feature -- Comparison
   make_from_array (a: ARRAY [G])
           -- Initialize from the items of a.
                                                                      is equal (other: like Current): BOOLEAN
           -- (Useful in proper descendants of class ARRAY,
                                                                              -- Is array made of the same items as other?
           -- to initialize an array-like object from a manifest
                                                                              -- (Redefined from GENERAL.)
           array.)
                                                                  feature -- Status report
feature -- Initialization
                                                                      valid_index (i: INTEGER): BOOLEAN
    make (minindex, maxindex: INTEGER)
                                                                              -- Is i within the bounds of the array?
           -- Set index interval to minindex .. maxindex
           -- reallocate if necessary; set all values to default.
                                                                  feature -- Element change
           -- (Make array empty if minindex > maxindex.)
                                                                      enter (v: G; i: INTEGER)
       ensure
                                                                              -- Replace i-th entry, if in index interval, by v.
           no_count: (minindex > maxindex) implies (count =
                                                                              -- (Redefinable synonym for put.)
           count_constraint: (minindex <= maxindex)</pre>
                                                                              good_key: valid_index (i)
             implies (count = maxindex - minindex + 1)
                                                                          ensure
   make_from_array(a: ARRAY[G])
                                                                              inserted: item (i) = v
           -- Initialize from the items of a; reallocate if
           -- necessary. (Useful in proper descendants of
                                                                      force (v: like item; i: INTEGER)
           -- class ARRAY, to initialize an array-like object
                                                                              -- Assign item v to i-th entry.
           -- from a manifest array.)
                                                                              -- Always applicable: resize the array if i falls out
feature -- Access
                                                                              of
                                                                              -- currently defined bounds; preserve existing
    entry (i: INTEGER): G
                                                                              items.
           -- Entry at index i, if in index interval
           -- (Redefinable synonym for item and infix "@".)
                                                                          ensure
                                                                              inserted: item (i) = v;
           good_key: valid_index (i)
                                                                              higher_count: count >= old count
```

§5.12 CLASS ARRAY 34

```
frozen put (v: like item; i: INTEGER)
           -- Replace i-th entry, if in index interval, by v.
           good_key: valid_index (i)
       ensure
           inserted: item(i) = v
feature -- Resizing
   resize (minindex, maxindex: INTEGER)
           -- Rearrange array so that it can accommodate
           -- indices down to minindex and up to maxindex.
           -- Do not lose any previously entered item.
       require
           good\_indices: minindex <= maxindex
       ensure
           no_low_lost: lower = minindex.min (old lower)
           no_high_lost: upper = maxindex.max (old upper)
feature -- Conversion
   to_c: POINTER
           -- Address of actual sequence of values,
           -- for passing to external (non-Eiffel) routines.
feature -- Duplication
   copy (other: like Current)
           -- Reinitialize by copying all the items of other.
           -- (This is also used by clone.)
           -- (From GENERAL.)
invariant
   consistent\_size: count = upper - lower + 1;
   non_negative_count: count >= 0
end
```

5.13 Class STRING

```
feature -- Access
indexing
   description: "Sequences of characters, accessible
                                                                      hash_code: INTEGER
      through integer indices in a contiguous range."
                                                                              -- Hash code value
                                                                              -- (From HASHABLE.)
class interface
    STRING
                                                                             good_hash_value: Result >= 0
creation
                                                                      index_of(c: CHARACTER; start: INTEGER): INTEGER
   frozen make (n: INTEGER)
                                                                              -- Position of first occurrence of c at or after start;
                                                                             -- 0 if none.
           -- Allocate space for at least n characters.
                                                                          require
       require
                                                                              start_large_enough: start >= 1
           non\_negative\_size: n >= 0
                                                                             start_small_enough: start <= count
       ensure
                                                                          ensure
           empty\_string: count = 0
                                                                             non negative result: Result >= 0
   make_from_string (s: STRING)
                                                                             at_this_position: Result > 0 implies item (Result)
           -- Initialize from the characters of s.
           -- (Useful in proper descendants of class STRING,
                                                                             -- none_before: For every i in start..Result, item (i)
           -- to initialize a string-like object from a manifest
           string.)
                                                                              -- zero_iff_absent:
                                                                              -- (Result = 0) = For every i in 1...count, item (i)
       require
                                                                             /=c
           string_exists: s/= Void
                                                                      item (i: INTEGER): CHARACTER
feature -- Initialization
                                                                              -- Character at position i
   from_c (c_string: POINTER)
                                                                          require
           -- Reset contents of string from contents of c_
                                                                             good_key: valid_index (i)
                                                                      substring_index (other: STRING; start: INTEGER):
           -- a string created by some external C function.
                                                                        INTEGER
                                                                              -- Position of first occurrence of other at or after
           C_string_exists: c_string /= Void
                                                                             start;
                                                                              -- 0 if none.
   frozen remake (n: INTEGER)
                                                                      infix "@" (i: INTEGER): CHARACTER
           -- Allocate space for at least n characters.
                                                                              -- Character at position i
       require
                                                                          require
           non\_negative\_size: n >= 0
                                                                              good_key: valid_index (i)
       ensure
                                                                  feature -- Measurement
           empty\_string: count = 0
                                                                      count: INTEGER
   make_from_string (s: STRING)
                                                                              -- Actual number of characters making up the
           -- Initialize from the characters of s.
                                                                             string
           -- (Useful in proper descendants of class STRING,
                                                                      occurrences (c: CHARACTER): INTEGER
           -- to initialize a string-like object from a manifest
           string.)
                                                                              -- Number of times c appears in the string
       require
           string_exists: s/= Void
                                                                             non_negative_occurrences: Result >= 0
```

§5.13 CLASS STRING 36

```
feature -- Comparison
                                                                      max (other: like Current): like Current)
                                                                             -- The greater of current object and other
   is_equal (other: like Current): BOOLEAN
                                                                             -- (From COMPARABLE.)
           -- Is string made of same character sequence as
                                                                          require
           other?
                                                                             other_exists: other /= Void
           -- (Redefined from GENERAL.)
                                                                          ensure
       require
                                                                             current_if_not_smaller: (Current >= other)
                                                                               implies (Result = Current)
           other_exists: other /= Void
                                                                             other_if_smaller: (Current < other) implies
   infix "<" (other: STRING): BOOLEAN
                                                                               (Result = other)
           -- Is string lexicographically lower than other?
                                                                      min (other: like Current): like Current)
           -- (False if other is void)
                                                                              -- The smaller of current object and other
           -- (From COMPARABLE.)
                                                                             -- (From COMPARABLE.)
                                                                          require
       require
                                                                             other_exists: other /= Void
           other_exists: other /= Void
                                                                         ensure
       ensure
                                                                             current_if_not_greater: (Current <= other)</pre>
           asymmetric: Result implies not (other < Current)
                                                                                implies (Result = Current)
                                                                             other_if_greater: (Current > other) implies
   infix "<=" (other: like Current): BOOLEAN
                                                                                (Result = other)
           -- Is current object less than or equal to other?
                                                                      three_way_comparison (other: like Current): INTEGER)
           -- (From COMPARABLE.)
                                                                             -- If current object equal to other, 0; if smaller,
                                                                             --1; if greater, 1.
       require
                                                                             -- (From COMPARABLE.)
           other exists: other /= Void
                                                                          require
                                                                             other_exists: other /= Void
       ensure
           definition: Result = (Current < other) or is_equal
             (other)
                                                                             equal\_zero: (Result = 0) = is\_equal (other)
                                                                             smaller: (Result = -1) = Current < other
   infix ">=" (other: like Current): BOOLEAN
                                                                             greater\_positive: (Result = 1) = Current > other
           -- Is current object greater than or equal to other?
                                                                  feature -- Status report
           -- (From COMPARABLE.)
                                                                      empty: BOOLEAN
       require
                                                                             -- Is string empty?
           other_exists: other /= Void
                                                                      valid_index (i: INTEGER): BOOLEAN
                                                                             -- Is i within the bounds of the string?
       ensure
                                                                  feature -- Element change
           definition: Result = (other <= Current)
                                                                      append_boolean (b: BOOLEAN)
   infix ">" (other: like Current): BOOLEAN
                                                                             -- Append the string representation of b at end.
           -- Is current object greater than other?
                                                                      append_character (c: CHARACTER)
           -- (From COMPARABLE.)
                                                                             -- Append c at end.
       require
                                                                         ensure
                                                                             item\_inserted: item (count) = c
           other_exists: other /= Void
                                                                             one\_more\_occurrence: occurrences(c) = old
       ensure
                                                                                (occurrences(c)) + 1
           definition: Result = (other < Current)
                                                                             item_inserted: has (c)
```

```
append_double (d: DOUBLE)
                                                                     put (c: CHARACTER; i: INTEGER)
                                                                             -- Replace character at position i by c.
        -- Append the string representation of d at end.
                                                                         require
append_integer (i: INTEGER)
                                                                             good_key: valid_index (i)
        -- Append the string representation of i at end.
                                                                         ensure
append_real (r: REAL)
                                                                             insertion\_done: item(i) = c
        -- Append the string representation of r at end.
                                                                     put_substring (s: like Current; start_pos, end_pos:
append_string (s: STRING)
                                                                        INTEGER)
        -- Append a copy of s, if not void, at end.
                                                                             -- Copy the characters of s to positions
    ensure
                                                                             -- start_pos .. end_pos.
       new\_count: count = old count + s \cdot count
                                                                         require
        -- appended: For every i in 1..s.count,
                                                                             string_exists: s /= Void;
             item (old count + i) = s \cdot item (i)
                                                                             index_small_enough: end_pos <= count;</pre>
fill (c: CHARACTER)
                                                                             order_respected: start_pos <= end_pos;</pre>
        -- Replace every character with c.
                                                                             index_large_enough: start_pos > 0
    ensure
                                                                         ensure
        -- allblank: For every i in 1..count, item (i) = Blank
                                                                             new\_count: count = old count + s \cdot count - end_
                                                                               pos + start\_pos - 1
head (n: INTEGER)
        -- Remove all characters except for the first n;
                                                                     right_adjust
        -- do nothing if n \ge count.
                                                                             -- Remove trailing white space.
    require
        non\_negative\_argument: n >= 0
                                                                             new_count: (count /= 0) implies (item (count) /= '
                                                                                ')
    ensure
                                                                     tail (n: INTEGER)
        new\_count: count = n \cdot min (old count)
                                                                             -- Remove all characters except for the last n;
        -- first_kept: For every i in 1..n, item (i) = old item
                                                                             -- do nothing if n \ge count.
                                                                         require
insert (s: like Current; i: INTEGER)
                                                                             non\_negative\_argument: n >= 0
        -- Add s to the left of position i.
    require
                                                                             new\_count: count = n \cdot min (old count)
        string_exists: s /= Void;
                                                                 feature -- Removal
        index\_small\_enough: i \le count;
        index\_large\_enough: i > 0
                                                                     remove (i: INTEGER)
                                                                             -- Remove i-th character.
    ensure
                                                                         require
       new\_count: count = old count + s \cdot count
                                                                             index_small_enough: i <= count;</pre>
insert character (c: CHARACTER; i: INTEGER)
                                                                             index\_large\_enough: i > 0
        -- Add c to the left of position i.
                                                                         ensure
                                                                             new\_count: count = old count - 1
       new\_count: count = old count + 1
                                                                     wipe_out
left_adjust
                                                                             -- Remove all characters.
        -- Remove leading white space.
                                                                         ensure
    ensure
                                                                             empty_string: count = 0
        new\_count: (count /= 0) implies (item (1) /= ' ')
                                                                             wiped_out: empty
```

§5.13 CLASS STRING

```
feature -- Resizing
                                                                    feature -- Output
                                                                        out: STRING
   resize (newsize: INTEGER)
                                                                               -- Printable representation
           -- Rearrange string so that it can accommodate
                                                                               -- (From GENERAL.)
           -- at least newsize characters.
                                                                           ensure
           -- Do not lose any previously entered character.
                                                                               result_not_void: Result /= Void
       require
                                                                   invariant
           new_size_non_negative: newsize >= 0
                                                                        irreflexive_comparison: not (Current < Current)</pre>
feature -- Conversion
                                                                       empty\_definition: empty = (count = 0);
   to_boolean: BOOLEAN
                                                                        non_negative_count: count >= 0
           -- Boolean value;
           -- "true" yields true, "false" yields false
                                                                   end
           -- (case-insensitive)
   to_double: DOUBLE
           -- "Double" value; for example, when applied to
           "123.0",
           -- will yield 123.0 (double)
   to_integer: INTEGER
           -- Integer value;
           -- for example, when applied to "123", will yield
   to_lower
           -- Convert to lower case.
   to_real: REAL
           -- Real value;
           -- for example, when applied to "123.0", will yield
           123.0
   to_upper
           -- Convert to upper case.
feature -- Duplication
   copy (other: like Current)
           -- Reinitialize by copying the characters of other.
           -- (This is also used by clone.)
           -- (From GENERAL.)
       ensure
           new_result_count: count = other.count
           -- same_characters: For every i in 1..count,
                item(i) = other \cdot item(i)
   substring (n1, n2: INTEGER): like Current
           -- Copy of substring containing all characters at indices
           -- between n1 and n2
       require
           meaningful\_origin: 1 \le n1;
           meaningful\_interval: n1 \le n2;
           meaningful\_end: n2 \le count
           new_result_count: Result_count = n2 - n1 + 1
           -- original_characters: For every i in 1..n2-n1,
                Result • item (i) = item (nl+i-1)
```

5.14 Class STD_FILES

indexing	put_new_line
description: "Commonly used input and output	Write line feed at end of default output.
mechanisms. This class may be used as either ancestor or supplier by classes needing its facilities."	put_real (r: REAL)
	Write <i>r</i> at end of default output.
class interface	<pre>put_string (s: STRING)</pre>
STD_FILES	Write s at end of default output.
feature Access	require
default_output: FILE	s /= Void
Default output.	set_error_default
error: FILE	Use standard error as default output.
Standard error file	set_output_default
input: FILE	Use standard output as default output.
Standard input file	feature Input
output: FILE Standard output file	•
•	read_character
standard_default: FILE Return the default_output or output	Read a new character from standard input.Make result available in <i>last_character</i>.
if default_output is Void.	read_double
feature Status report	Read a new double from standard input.
last_character: CHARACTER	Make result available in <i>last_double</i> .
Last character read by read_character	read_integer
last_double: DOUBLE	Read a new integer from standard input.
Last double read by read_double	Make result available in <i>last_integer</i> .
last_integer: INTEGER Last integer read by read_integer	read_line
	Read a line from standard input.
last_real: REAL Last real read by read_real	 Make result available in <i>last_string</i>. New line will be consumed but not part of <i>last_</i>
last_string: STRING	string.
Last string read by read_line,	read real
read_stream, or read_word	Read a new real from standard input.
feature Element change	Make result available in last_real.
put_boolean (b: BOOLEAN)	read_stream (nb_char: INTEGER)
Write <i>b</i> at end of default output.	Read a string of at most nb_char bound
<pre>put_character (c: CHARACTER)</pre>	characters
Write c at end of default output.	 from standard input. Make result available in last_string.
put_double (d: DOUBLE)	_
Write <i>d</i> at end of default output.	to_next_line Move to next input line on standard input.
put_integer (i: INTEGER)	
Write <i>i</i> at end of default output.	end

§5.15 CLASS FILE **40**

5.15 Class FILE

```
indexing
                                                                      make_open_read (fn: STRING)
                                                                              -- Create file object with fn as file name
   description: "Files viewed as persistent sequences of
                                                                              -- and open file in read mode.
      characters."
                                                                          require
class interface
                                                                              string_exists: fn /= Void;
                                                                              string_not_empty: not fn•empty
    FILE
                                                                          ensure
creation
                                                                              exists: exists;
   make (fn: STRING)
                                                                              open_read: is_open_read
           -- Create file object with fn as file name.
                                                                      make_open_read_write (fn: STRING)
       require
                                                                              -- Create file object with fn as file name
           string_exists: fn /= Void;
                                                                              -- and open file for both reading and writing.
           string_not_empty: not fn.empty
                                                                          require
                                                                              string_exists: fn /= Void;
       ensure
                                                                              string_not_empty: not fn.empty
           file_named: name • is_equal (fn);
                                                                          ensure
           file_closed: is_closed
                                                                              exists: exists;
   make_create_read_write (fn: STRING)
                                                                              open_read: is_open_read;
           -- Create file object with fn as file name
                                                                              open_write: is_open_write
           -- and open file for both reading and writing;
                                                                      make_open_write (fn: STRING)
           -- create it if it does not exist.
                                                                              -- Create file object with fn as file name
       require
                                                                              -- and open file for writing;
                                                                              -- create it if it does not exist.
           string_exists: fn /= Void;
                                                                          require
           string_not_empty: not fn.empty
                                                                              string_exists: fn /= Void;
       ensure
                                                                              string_not_empty: not fn•empty
           exists: exists;
                                                                          ensure
           open_read: is_open_read;
                                                                              exists: exists;
           open_write: is_open_write
                                                                              open_write: is_open_write
   make_open_append (fn: STRING)
                                                                   feature -- Access
           -- Create file object with fn as file name
                                                                       name: STRING
           -- and open file in append-only mode.
                                                                              -- File name
       require
                                                                   feature -- Measurement
           string_exists: fn /= Void;
                                                                      count: INTEGER
           string_not_empty: not fn•empty
                                                                              -- Size in bytes (0 if no associated physical file)
       ensure
                                                                  feature -- Status report
           exists: exists;
                                                                      empty: BOOLEAN
           open_append: is_open_append
                                                                              -- Is structure empty?
```

```
end_of_file: BOOLEAN
                                                                     open_read
                                                                             -- Open file in read-only mode.
          -- Has an EOF been detected?
                                                                         require
       require
                                                                             is_closed: is_closed
          opened: not is_closed
                                                                         ensure
   exists: BOOLEAN
                                                                             exists: exists
          -- Does physical file exist?
                                                                             open_read: is_open_read
   is closed: BOOLEAN
                                                                     open_read_append
          -- Is file closed?
                                                                             -- Open file in read and write-at-end mode;
   is_open_read: BOOLEAN
                                                                             -- create it if it does not exist.
          -- Is file open for reading?
                                                                         require
                                                                             is_closed: is_closed
   is open write: BOOLEAN
           -- Is file open for writing?
                                                                         ensure
                                                                             exists: exists
   is_plain_text: BOOLEAN
                                                                             open_read: is_open_read
          -- Is file reserved for text (character sequences)?
                                                                             open_append: is_open_append
   is_readable: BOOLEAN
                                                                     open_read_write
          -- Is file readable?
                                                                             -- Open file in read and write mode.
       require
                                                                         require
          handle_exists: exists
                                                                             is closed: is closed
   is writable: BOOLEAN
                                                                         ensure
          -- Is file writable?
                                                                             exists: exists
       require
                                                                             open_read: is_open_read
          handle_exists: exists
                                                                             open_write: is_open_write
   last character: CHARACTER
                                                                     open_write
          -- Last character read by read_character
                                                                             -- Open file in write-only mode;
                                                                             -- create it if it does not exist.
   last_double: DOUBLE
                                                                         ensure
          -- Last double read by read_double
                                                                             exists: exists
   last\_integer: INTEGER
                                                                             open_write: is_open_write
          -- Last integer read by read_integer
                                                                  feature -- Cursor movement
   last_real: REAL
                                                                     to next line
          -- Last real read by read_real
                                                                             -- Move to next input line.
   last_string: STRING
                                                                         require
          -- Last string read by read_line,
                                                                             readable: is readable
          -- read_stream, or read_word
                                                                  feature -- Element change
feature -- Status setting
                                                                     change_name (new_name: STRING)
   close
                                                                             -- Change file name to new_name
          -- Close file.
                                                                         require
       require
                                                                             not_new_name_void: new_name /= Void;
          medium_is_open: not is_closed
                                                                            file_exists: exists
       ensure
                                                                         ensure
           is_closed: is_closed
                                                                             name_changed: name • is_equal (new_name)
```

§5.15 CLASS FILE 42

feature Removal	feature Output
delete Remove link with physical file; delete physical file if no more link.require	<pre>put_boolean (b: BOOLEAN) Write ASCII value of b at current position. require extendible: extendible</pre>
exists: exists dispose Ensure this medium is closed when garbage-collected.	<pre>put_character (c: CHARACTER) Write c at current position. require extendible: extendible</pre>
feature Input	
read_character Read a new character Make result available in last_character.	<pre>put_double (d: DOUBLE) Write ASCII value of d at current position. require extendible: extendible</pre>
require readable: is_readable	put_integer (i: INTEGER)
read_double Read the ASCII representation of a new double from file. Make result available in last_double.	Write ASCII value of <i>i</i> at current position. require extendible: extendible
require readable: is_readable read_integer	<pre>put_real (r: REAL) Write ASCII value of r at current position. require</pre>
Read the ASCII representation of a new integer from file. Make result available in <i>last_integer</i> . require	extendible: extendible put_string (s: STRING) Write s at current position.
readable: is_readable	require
read_line Read a string until new line or end of file Make result available in laststring New line will be consumed but not part of last_	extendible: extendible invariant name_exists: name /= Void;
string.	name_not_empty: not name • empty;
require	writable_if_extendible: extendible implies is_writable
readable: is_readable read_real	end
Read the ASCII representation of a new real from file. Make result available in <i>last_real</i> .	
require	
readable: is_readable	
 read_stream (nb_char: INTEGER) Read a string of at most nb_char bound characters or until end of file. Make result available in last_string. 	
require readable: is_readable	
read_word Read a new word from standard input. Make result available in last_string.	

5.16 Class STORABLE

indexing description: "Objects that may be stored and retrieved along with all their dependents. This class may be used as ancestor by classes needing its facilities." class interface **STORABLE** feature -- Access retrieved (file: FILE): STORABLE -- Retrieved object structure, from external -- representation previously stored in *file*. -- To access resulting object under correct type, -- use assignment attempt. -- Will raise an exception (code Retrieve_ exception) -- if file content is not a STORABLE structure. require file_not_void: file /= Void; file_exists: file .exists; file_is_open_read: file .is_open_read file_not_plain_text: not file .is_plain_text ensure result_exists: Result /= Void feature -- Element change basic_store (file: FILE) -- Produce on file an external representation of the -- entire object structure reachable from current -- Retrievable within current system only. require file_not_void: file /= Void; file_exists: file.exists; file_is_open_write: file .is_open_write

file_not_plain_text: **not** file • is_plain_text

end

```
general_store (file: FILE)
        -- Produce on file an external representation of the
        -- entire object structure reachable from current
       object.
        -- Retrievable from other systems for same
       platform
        -- (machine architecture).
   require
       file_not_void: file /= Void;
       file_exists: file.exists;
       file_is_open_write: file • is_open_write
       file_not_plain_text: not file • is_plain_text
independent_store (file: FILE)
       -- Produce on file an external representation of the
        -- entire object structure reachable from current
        -- Retrievable from other systems for the same or
        -- platforms (machine architectures).
   require
       file_not_void: file /= Void;
       file_exists: file .exists;
       file_is_open_write: file • is_open_write
       file_not_plain_text: not file .is_plain_text
```

§5.17 CLASS MEMORY 44

5.17 Class MEMORY

indexing

description: "Facilities for tuning up the garbage collection mechanism. This class may be used as ancestor by classes needing its facilities."

class interface

MEMORY

feature -- Status report

collecting: BOOLEAN

-- Is garbage collection enabled?

feature -- Status setting

collection_off

-- Disable garbage collection.

collection_on

-- Enable garbage collection.

feature -- Removal

dispose

- -- Action to be executed just before garbage collection
- -- reclaims an object.
- -- Default version does nothing; redefine in descendants
- -- to perform specific dispose actions. Those actions
- -- should only take care of freeing external resources
- -- they should not perform remote calls on other objects
- -- since these may also be dead and reclaimed.

full_collect

- -- Force a full collection cycle if garbage
- -- collection is enabled; do nothing otherwise.

end

5.18 Class EXCEPTIONS

indexing

description: "Facilities for adapting the exception handling mechanism. This class may be used as ancestor by classes needing its facilities."

class interface

EXCEPTIONS

feature -- Access

developer_exception_name: STRING

-- Name of last developer-raised exception

require

applicable: is_developer_exception

feature -- Access

Check_instruction: INTEGER

-- Exception code for violated check

Class_invariant: INTEGER

-- Exception code for violated class invariant

Incorrect_inspect_value: INTEGER

-- Exception code for inspect value which is not

-- of the inspect constants, if there is no Else_part

Loop_invariant: INTEGER

-- Exception code for violated loop invariant

Loop_variant: INTEGER

-- Exception code for non-decreased loop variant

No more memory: INTEGER

-- Exception code for failed memory allocation

Postcondition: INTEGER

-- Exception code for violated postcondition

Precondition: INTEGER

-- Exception code for violated precondition

Routine_failure: INTEGER

-- Exception code for failed routine

Void_attached_to_expanded: INTEGER

-- Exception code for attachment of void value

-- to expanded entity

Void_call_target: INTEGER

-- Exception code for violated check

feature -- Status report

assertion_violation: BOOLEAN

-- Is last exception originally due to a violated

-- assertion or non-decreasing variant?

exception: INTEGER

-- Code of last exception that occurred

is_developer_exception: BOOLEAN

-- Is the last exception originally due to

-- a developer exception?

is_signal: BOOLEAN

-- Is last exception originally due to an external

-- event (operating system signal)?

feature -- Basic operations

die (code: INTEGER)

-- Terminate execution with exit status *code*,

-- without triggering an exception.

raise (name: STRING)

-- Raise a developer exception of name *name*.

end

§5.19 CLASS ARGUMENTS 46

5.19 Class ARGUMENTS

```
indexing
   description: "Access to command-line arguments. This
     class may be used as ancestor by classes needing its
     facilities."
class interface
   ARGUMENTS
feature -- Access
   argument (i: INTEGER): STRING
          -- i-th argument of command that started system
          execution
           -- (the command name if i = 0)
       require
          index\_large\_enough: i >= 0
          index\_small\_enough: i \le argument\_count
   command_name: STRING
          -- Name of command that started system execution
       ensure
          definition: Result = argument(0)
feature -- Measurement
   argument_count: INTEGER
          -- Number of arguments given to command that
          -- system execution (command name does not
          count)
       ensure
          Result >= 0
end
```

5.20 Class PLATFORM

```
indexing
                                                                 Minimum_character_code: INTEGER
                                                                        -- Smallest supported code for CHARACTER
   description: "Platform-dependent properties. This class
                                                                        values
     may be used as ancestor by classes needing its
                                                                    ensure
     facilities."
                                                                        meaningful: Result <= 0
class interface
                                                                 Minimum_integer: INTEGER
   PLATFORM
                                                                        -- Smallest supported value of type INTEGER
feature -- Access
                                                                        meaningful: Result <= 0
   Boolean_bits: INTEGER
                                                                 Pointer_bits: INTEGER
          -- Number of bits in a value of type BOOLEAN
                                                                        -- Number of bits in a value of type POINTER
       ensure
                                                                    ensure
          meaningful: Result >= 1
                                                                        meaningful: Result >= 1
   Character_bits: INTEGER
                                                                 Real_bits: INTEGER
          -- Number of bits in a value of type CHARACTER
                                                                        -- Number of bits in a value of type REAL
                                                                        meaningful: Result >= 1
          meaningful: Result >= 1
          large_enough: 2 ^ Result >= Maximum_
                                                             end
            character_code
   Double_bits: INTEGER
          -- Number of bits in a value of type DOUBLE
          meaningful: Result >= 1
          meaningful: Result >= Real_bits
   Integer_bits: INTEGER
          -- Number of bits in a value of type INTEGER
       ensure
          meaningful: Result >= 1
          large_enough: 2 ^ Result >= Maximum_integer
          large_enough_for_negative: 2 ^ Result >=
            -Minimum_integer
   Maximum_character_code: INTEGER
          -- Largest supported code for CHARACTER values
       ensure
          meaningful: Result >= 127
   Maximum_integer: INTEGER
          -- Largest supported value of type INTEGER.
       ensure
          meaningful: Result >= 0
```

5.21 Class BOOLEAN_REF

```
indexing
   description: "Reference class for BOOLEAN"
class interface
   BOOLEAN_REF
feature -- Access
   item: BOOLEAN
          -- Boolean value
   hash_code: INTEGER
          -- Hash code value
          -- (From HASHABLE.)
      ensure
          good_hash_value: Result >= 0
feature -- Element change
   set_item (b: BOOLEAN)
          -- Make b the associated boolean value.
       ensure
          item\_set: item = b
end
```

5.22 Class CHARACTER_REF

```
indexing
   description: "Reference class for CHARACTER"
class interface
   CHARACTER_REF
feature -- Access
   item: CHARACTER
          -- Character value
   hash_code: INTEGER
          -- Hash code value
          -- (From HASHABLE.)
          good_hash_value: Result >= 0
feature -- Element change
   set_item (c: CHARACTER)
          -- Make c the associated character value.
       ensure
          item\_set: item = c
end
```

5.23 Class DOUBLE_REF

```
indexing
   description: "Reference class for DOUBLE"
class interface
   DOUBLE_REF
feature -- Access
   item: DOUBLE
          -- Double value
   hash_code: INTEGER
          -- Hash code value
          -- (From HASHABLE.)
      ensure
          good_hash_value: Result >= 0
feature -- Element change
   set_item (d: DOUBLE)
          -- Make d the associated double value.
       ensure
          item\_set: item = d
end
```

5.24 Class INTEGER_REF

```
indexing
   description: "Reference class for INTEGER"
class interface
   INTEGER_REF
feature -- Access
   item: INTEGER
          -- Integer value
   hash_code: INTEGER
          -- Hash code value
          -- (From HASHABLE.)
          good_hash_value: Result >= 0
feature -- Element change
   set_item (i: INTEGER)
          -- Make i the associated integer value.
       ensure
          item\_set: item = i
end
```

5.25 Class POINTER_REF

```
indexing
   description: "Reference class for POINTER"
class interface
   POINTER_REF
feature -- Access
   item: POINTER
          -- Pointer value
   hash_code: INTEGER
          -- Hash code value
          -- (From HASHABLE.)
      ensure
          good_hash_value: Result >= 0
feature -- Element change
   set_item (p: POINTER)
          -- Make p the associated pointer value.
       ensure
          item\_set: item = p
end
```

5.26 Class REAL_REF

```
indexing
   description: "Reference class for REAL"
class interface
   REAL_REF
feature -- Access
   item: REAL
          -- Real value
   hash_code: INTEGER
          -- Hash code value
          -- (From HASHABLE.)
          good_hash_value: Result >= 0
feature -- Element change
   set_item (r: REAL)
          -- Make r the associated real value.
          item\_set: item = r
end
```

6 APPENDIX A: THE KERNEL STANDARDIZATION PROCESS

[This Appendix is not part of the Standard.]

6.1 Why plan a process?

The Eiffel Kernel Library cannot be specified for eternity. Ideas willcome up for new classes and features; ways will be found to do thingsbetter. The evolution process must be fast enough to enable Eiffel users to benefit from this flow of ideas and avoid technical obsolescence, but orderly enough to protect their existing investments and modes of operation.

6.2 Cycle time

A revision every ten to fifteen years, as has occurred for programming language standards (Fortran, C and Ada are examples) is not appropriate for the Eiffel Kernel Library. It would foster complacency most of the time, and major upheavals when a revision is finally brought into effect. A yearly process, similar to the upgrading of wines, car models and stable software products, provides the right pace of change.

6.3 Vintages

Each revision of this Standard describes a **vintage** of the Eiffel Library Kernel Standard. The present version is vintage 1995.

6.4 Yearly schedule

The following deadlines apply to year *year*:

- 1 January: Vintage *year* takes effect.
- 1 April: first permitted date for starting discussions on Vintage *year+1* in NICE's Library Committee. (1 January to 31 March is acooling-off period.)
- 6.4.3 1 May: first permitted date for submitting formal proposals to the Library Committee for Vintage year + 1.
- 1 July: last permitted date for submitting initial proposals for Vintage *year* + *1*.
- 6.4.5 1 September: last permitted date for submitting final proposals (which may result from merging of several proposals) for Vintage year + 1.
- 1 October: last date by which the Committee may have defined Vintage year +1.

This schedule is applicable starting with vintage 96. For the present vintage (95), the first, the date of applicability is 1 July 1995.

6.8

6.9

6.10 6.11

6.5 Intermediate corrections

During the time when a vintage is in effect, minor corrections may prove necessary, due for example to typographical errors in the current version of this Standard or to inconsistencies discovered by users or implementors of Eiffel. In such a case the chairman of the Library Committee of NICE may, at his discretion, submit a motion covering one or more revisions. To be approved, such motions shall require a unanimous vote of the Library Committee, with the possible exception of any member who has notified the chairman of an absence of more than one month. If approved, such a revision shall receive a revision level and shall give rise to a modified Kernel Library Standard, identified as "Vintage *year* Level *revision_level*". The modifications shall be integrated into the following year's vintage.

6.6 Eiffel Kernel Supplier requirements

Any provider of an Eiffel environment must make the following information available to any NICE member:

- Vintage and revision level currently supported.
 - Any features not supported. (It is not permitted to have a non-supported class.)
 - List of classes needed by kernel classes, but not in the kernel, hereafter referred to as para-kernel classes.
 - Full inheritance hierarchy of kernel and para-kernel classes.
 - List of names of features (immediate or inherited) that appear in the provider's kernel classes but not in this Standard.

7.2

7.3

7.4

7.5

7.6

7.7

7.8

7 APPENDIX B: DIFFERENCES UP TO ELKS 95

[This Appendix is not part of the Standard.]

The following differences exist between this Standard and earlierpresentations of the Kernel Library:

- Addition to *GENERAL* of a query *default* which returns the default value of the type of the current object. This also addresses the need to obtain the default value for type *POINTER*; for convenience, since *POINTER* has no manifest constant, a query *default_pointer* has also been included. (See page 14.)
 - Adaptation of the semantics of *copy* and equality features (*equal*, *is_equal* and their *standard_* versions) so that the result is true if and only if the objects are truly identical, and in particular have the same type. This implies a language change too; the previous definition was non-symmetric so that *a.copy* (*b*) and *equal* (*a, b*) only applied to the fields corresponding to the attributes of *a*'s type. The earlier effect can still be achieved through function *stripped*, as explained next in 7.5. (See page 13.)
 - Addition to *GENERAL* of a frozen feature *same_type* which specifies conformance both ways. Addition of the requirement that *conforms_to* is frozen too. (See page 13.)
 - Addition of a number of assertion clauses to the features of *GENERAL*, in particular to specify more precisely the semantics of equality, copying, cloning and conformance.
 - Addition to GENERAL of a function stripped such that stripped (a) is a clone of the current object limited to the fields that apply to a's dynamic type. As a result, the old semantics of copying and equality mentioned in 7.2 may now be achieved through calls such as a.copy (b.stripped (a)) and equal (a, b.stripped (a)). (See page 13.)
 - Addition to *GENERAL* of *object_id* and *id_object* to allow unique identification of objects. (See page 13.)
 - In class *PLATFORM*, removal of the assumption that *Character_bits*, *Integer_bits*, *Real_bits* and *Double_bits* are constants. This does not introduce any incompatibility with earlier uses except if they relied on the specific numerical values. (See page 47.)
 - Removal of *PLATFORM* from the universal inheritance hierarchy; *PLATFORM* is no longer a parent of *ANY* and hence an ancestor of every

	class, and has no particular language-defined role; classes that need its facilities must name it explicitly among their proper ancestors. This is actually a language change. (See section 4.)
7.9	• Addition to <i>PLATFORM</i> of features <i>Maximum_integer</i> , <i>Minimum_integer</i> , <i>Maximum_character_code</i> and <i>Minimum_character_code</i> . (See page 47.)
7.10	• Addition to <i>COMPARABLE</i> of <i>min</i> and <i>max</i> functions and of a three-way comparison function, <i>three_way_comparison</i> , which returns 0, -1 or 1. (See page 16.)
7.11	• Addition to the arithmetic basic classes of functions <i>abs</i> and <i>sign</i> (the latter defined in terms of <i>three_way_comparison</i>). Addition to <i>REAL</i> and <i>DOUBLE</i> of <i>floor</i> , <i>ceiling</i> , <i>rounded</i> and <i>integer_part</i> . Addition to <i>DOUBLE</i> of <i>real_part</i> . (See page 24 and following.)
7.12	• Addition of inheritance links making all basic classes (<i>INTEGER</i> and so on) heirs of <i>HASHABLE</i> , so that it is now possible to hash any object. (See section 4.) Removal of function <i>is_hashable</i> and the corresponding preconditions.
7.13	• Addition to <i>ARRAY</i> of features <i>enter</i> and <i>entry</i> as redefinable synonyms to <i>put</i> and <i>item</i> (or <i>infix</i> "@"), the latter becoming frozen. (See page 33.)
7.14	• Addition to <i>STORABLE</i> of a procedure <i>independent_store</i> which produces machine-independent representations of object structures. (See page 43.)
7.15	• Addition of a few features to class <i>FILE</i> describing file opening and opening modes (such as read-only or read-write). In earlier presentations the corresponding class was <i>UNIX_FILE</i> . The new class is very similar but removes any Unix-specific aspect. (See section 5.15.)
7.16	• Changes of names in class <i>STD_FILES</i> and <i>FILE</i> : for consistency with the usual Eiffel naming style, underscores were added and abbrevations were expanded. In the following list (which uses the order of appearance of the features in <i>STD_FILES</i>), the added underscores appear as * and the added letters appear in <i>bold italics</i> : last*character, last*double, last*real, last*integer, last*string, put*boolean, put*character, put*double, put*integer, put_new*line, put*real, put*string, read*character, read*double, read*integer, read*line, read*real, read*stream, read*word, to_next*line. (See sections 5.14 and 5.15.)
7.17	• Addition to <i>EXCEPTIONS</i> of a procedure <i>die</i> that terminates the execution cleanly with a given exit status, without triggering an exception. (See page 45.)
7.18	• In class <i>STRING</i> , replacement of the <i>adapt</i> function by a more convenient procedure <i>make_from_string</i> which descendants of the class can use to

initialize a string-like object from a manifest string, as in !! t.make_from_

7.20

string ("THIS STRING"), where the type of *t* is a descendant of STRING. (See page 35.)

- Similarly, addition to *ARRAY* of a procedure *make_from_array* allowing initialization from a manifest array, as in !! *a.make_from_array* (<<*a*, *b*, *c*, *d*>>).
 - Removal from *STRING* of a number of features which some committee members judged too specialized: *mirror*, *mirrored*, *share*, *shared_with*, *item_code*, *has*, *prepend*, *set*, *prune*, *prune_all*. Renaming of *replace_substring* to *put_substring*. Removal of *fill_blanks*, replaced by *fill* (applying to an arbitrary character). Change of the result type of *out* to *STRING* (rather than *like Current*).

8 INDEX

[This Index is not part of the Standard.]

8.1

This Index indicates the page of definition of every class and feature appearing in the Required Flatshort Forms of section 5

8.2

Following the standard Eiffel conventions, feature names appear in lower-case italics and class names, when making up index entries, in **UPPER-CASE ITALICS**. Operator functions appear under **prefix** and **infix**; for example division appears under **infix** "/". This also applies to boolean operators, which appear under **infix** "and", **infix** "and then" and so on.

8.3

In a class entry, the class appears in *UPPER-CASE ITALICS*. Each reference to a feature name is followed by the name of the class or classes in which it is available, each with the corresponding page. To avoid any confusion with occurrences of the class name in its other role – as an index entry pointing to the beginning of the class specification – the class name in this case appears in UPPER-CASE ROMAN.

(ELKS 98 note: FrameMaker 5.5 acts so strange the font conventions don't hold any more. I have no idea what's going on and have written to Frame customer support in the hope it is not yet another bug of the new release but something stupid I am doing.)

	append_character
abs	STRING 36
DOUBLE 30	append_double
INTEGER 24	STRING 37
REAL 27	append_integer
ANY 15	STRING 37
append_boolean	append_real
STRING 36	STRING 37

§8 INDEX 60

append_string STRING 37	collecting MEMORY 44
argument ARGUMENTS 46	collection_off MEMORY 44
ARGUMENTS 46 argument_count	collection_on MEMORY 44
ARGUMENTS 46 ARRAY 33	command_name ARGUMENTS 46
	COMPARABLE 16
assertion_violation EXCEPTIONS 45	conforms_to GENERAL 13
basic_store STORABLE 43	consistent_size ARRAY 34
BOOLEAN 20 Boolean_bits PLATFORM 47	copy ARRAY 34 GENERAL 14 STRING 38
BOOLEAN_REF 48 ceiling DOUBLE 30 REAL 27	count ARRAY 33 FILE 40 STRING 35
change_name FILE 41	deep_clone GENERAL 14
CHARACTER 21	deep_equal
Character_bits	GENERAL 13
PLATFORM 47 CHARACTER_REF 49	default GENERAL 14
Check_instruction EXCEPTIONS 45	default_output STD_FILES 39
Class_invariant EXCEPTIONS 45	default_pointer GENERAL 14
clone GENERAL 14	default_rescue GENERAL 14
close FILE 41	delete FILE 42
code CHARACTER 21	developer_exception_name EXCEPTIONS 45

61 INDEX §8

die EXCEPTIONS 45 dispose FILE 42 MEMORY 44	exponentiable DOUBLE 30 INTEGER 24 NUMERIC 18 REAL 27
divisible DOUBLE 30 INTEGER 24 NUMERIC 18 REAL 27 DOUBLE 29 Double_bits PLATFORM 47 DOUBLE_REF 50 do_nothing	fill STRING 37 floor DOUBLE 30 REAL 27 force ARRAY 33 from_c STRING 35
empty FILE 40 STRING 36 empty_definition STRING 38 end_of_file FILE 41	full_collect MEMORY 44 GENERAL 13 general_store STORABLE 43 generating_type GENERAL 13
enter ARRAY 33	generator GENERAL 13
entry ARRAY 33 equal GENERAL 13	HASHABLE 17 hash_code BOOLEAN 20 BOOLEAN_REF 48
error STD_FILES 39 exception EXCEPTIONS 45 EXCEPTIONS 45 exists FILE 41	CHARACTER 21 CHARACTER_REF 49 DOUBLE 29 DOUBLE_REF 50 HASHABLE 17 INTEGER 23 INTEGER_REF 51 POINTER 32

§8 INDEX **62**

POINTER_REF 52 REAL_26 REAL_REF 53	infix "+" DOUBLE 30 INTEGER 24
STRING 35 head STRING 37	NUMERIC 18 REAL 27 infix "/"
id_object GENERAL 13	DOUBLE 30 INTEGER 24 NUMERIC 18 REAL 27
Incorrect_inspect_value EXCEPTIONS 45	infix "//" INTEGER 24
independent_store STORABLE 43	infix "<" CHARACTER 21
index_of STRING 35	COMPARABLE 16 DOUBLE 29
infix "and then" BOOLEAN 20	INTEGER 23 REAL 26
infix "and" BOOLEAN 20	STRING 36 infix "<="
infix "implies" BOOLEAN 20	CHARACTER 21 COMPARABLE 16 DOUBLE 29
infix "or else" BOOLEAN 20	INTEGER 23 REAL 26
infix "or" BOOLEAN 20	STRING 36 infix ">"
infix "xor" BOOLEAN 20	CHARACTER 21 COMPARABLE 16 DOUBLE 29
infix "-" DOUBLE 30 INTEGER 24	INTEGER 23 REAL 26 STRING 36
NUMERIC 18 REAL 27	infix ">=" CHARACTER 21
infix "*" DOUBLE 30 INTEGER 24 NUMERIC 18 REAL 27	COMPARABLE 16 DOUBLE 29 INTEGER 23 REAL 26 STRING 36

63 INDEX §8

infix "@"	is_signal
STRING 35	EXCEPTIONS 45
infix "\" INTEGER 24	is_writable FILE 41
infix "^" DOUBLE 31 INTEGER 24 NUMERIC 18 REAL 28 input STD_FILES 39	item BOOLEAN_REF 48 CHARACTER_REF 49 DOUBLE_REF 50 INTEGER_REF 51 POINTER_REF 52 REAL_REF 53
insert	item
STRING 37 insert_character STRING 37	ARRAY 33 STRING 35
INTEGER 23 Integer_bits PLATFORM 47	last_character FILE 41 STD_FILES 39
INTEGER_REF 51 io GENERAL 14	last_double FILE 41 STD_FILES 39
is_closed FILE 41	last_integer FILE 41 STD_FILES 39
is_developer_exception EXCEPTIONS 45 is_equal	last_real FILE 41 STD FILES 39
ARRAY 33 COMPARABLE 16 GENERAL 13 STRING 36	last_string FILE 41 STD_FILES 39
is_open_read FILE 41	left_adjust STRING 37
is_open_write FILE 41	Loop_invariant EXCEPTIONS 45
is_plain_text FILE 41	Loop_variant EXCEPTIONS 45
is_readable FILE 41	lower ARRAY 33

§8 INDEX **64**

	Minimum_character_code PLATFORM 47
make ARRAY 33 FILE 40 STRING 35	Minimum_integer PLATFORM 47
make_create_read_write FILE 40	name FILE 40
make_from_array ARRAY 33	name_exists FILE 42
make_from_string STRING 35	name_not_empty FILE 42
make_open_append FILE 40	non_negative_count ARRAY 34 STRING 38
make_open_read FILE 40	No_more_memory EXCEPTIONS 45
make_open_read_write FILE 40	NUMERIC 18
make_open_write FILE 40	object_id GENERAL 13
CHARACTER 21 COMPARABLE 16 DOUBLE 29 INTEGER 23 REAL 26 STRING 36	occurrences STRING 35 one DOUBLE 29 INTEGER 23 NUMERIC 18
Maximum_character_code PLATFORM 47	REAL 26 open_read
Maximum_integer PLATFORM 47	FILE 41 open_read_append FILE 41
MEMORY 44 min	open_read_write FILE 41
CHARACTER 21 COMPARABLE 16 DOUBLE 29 INTEGER 23	open_write FILE 41 out
REAL 26 STRING 36	BOOLEAN 20 CHARACTER 22

65 INDEX §8

DOUBLE 31 GENERAL 14 INTEGER 25 POINTER 32 REAL 28 STRING 38	put_character FILE 42 STD_FILES 39 put_double FILE 42 STD_FILES 39
output STD_FILES 39	put_integer FILE 42 STD_FILES 39
PLATFORM 47	put_new_line
POINTER 32	STD_FILES 39
Pointer_bits PLATFORM 47	put_real FILE 42
POINTER_REF 52	STD_FILES 39
Postcondition EXCEPTIONS 45	put_string FILE 42
Precondition EXCEPTIONS 45	STD_FILES 39
prefix "not" BOOLEAN 20	put_substring STRING 37
prefix "-" DOUBLE 31 INTEGER 25 NUMERIC 18 REAL 28	raise EXCEPTIONS 45 read_character FILE 42 STD_FILES 39
prefix "+"	read_double
DOUBLE 31 INTEGER 25 NUMERIC 18	FILE 42 STD_FILES 39
REAL 28	read_integer
print GENERAL 14	FILE 42 STD_FILES 39
put ARRAY 34 STRING 37	read_line FILE 42 STD_FILES 39
STRING 31	SID_FILES 39
put_boolean FILE 42 STD_FILES 39	read_real FILE 42 STD_FILES 39

§8 INDEX **66**

read_stream FILE 42 STD_FILES 39	set_output_default STD_FILES 39 sign
read_word FILE 42	DOUBLE 29 INTEGER 23 REAL 26
REAL 26	
Real_bits PLATFORM 47	standard_clone GENERAL 14
REAL_REF 53	standard_copy GENERAL 14
remake STRING 35	standard_default STD_FILES 39
remove STRING 37	standard_equal GENERAL 14
resize ARRAY 34 STRING 38	standard_is_equal GENERAL 14
retrieved	STD_FILES 39
STORABLE 43	STORABLE 43
right_adjust	STRING 35
STRING 37 rounded	stripped GENERAL 13
DOUBLE 30 REAL 27	substring STRING 38
Routine_failure EXCEPTIONS 45	substring_index STRING 35
same_type GENERAL 13	tagged_out GENERAL 14
set_error_default STD_FILES 39	tail STRING 37
set_item BOOLEAN_REF 48 CHARACTER_REF 49 DOUBLE_REF 50 INTEGER_REF 51 POINTER_REF 52 REAL_REF 53	three_way_comparison CHARACTER 21 COMPARABLE 16 DOUBLE 30 INTEGER 24 REAL 27 STRING 36

INDEX §8

to boolean writable_if_extendible STRING 38 FILE 42 to_c ARRAY 34 zero DOUBLE 29 to_double INTEGER 23 STRING 38 NUMERIC 18 to_integer REAL 26 STRING 38 Symbols to_lower "@" STRING 38 ARRAY 33 to_next_line FILE 41 STD_FILES 39 to_real STRING 38 to_upper STRING 38 truncated_to_integer DOUBLE 30 REAL 27 truncated_to_real DOUBLE 30 upper ARRAY 33 valid_index ARRAY 33 STRING 36 Void GENERAL 14 Void_attached_to_expanded **EXCEPTIONS 45** Void_call_target **EXCEPTIONS 45** wipe_out

STRING 37

§8 INDEX 68