ELKS: The Eiffel Library
Kernel Standard

A.1 OVERVIEW

A.1.1 Purpose
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 as the Kernel Library.

A.1.2 Application
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 A.3.2 — 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.

A.1.3 Process
The Eiffel Library standardization process 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 2005, valid for the calendar years 2005-2006.

A.1.4 Copyright status
This Standard is appendix A of the book Eiffel: The Language by Bertrand Meyer (Prentice Hall, 2002) and the copyright belongs to the author. Electronic or paper reproduction of this Standard is permitted provided the reproduction includes the entire text of the Standard, including the present copyright notice and the mention that the latest version, up-to-date with any error corrections, may be found at http://eiffel.com.

A.2 CONTENTS OF THIS STANDARD

A.2.1 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 A.2 to A.6 of the present appendix, with the exception of elements appearing in black between square brackets […] which are comments.

A.2.2 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”.

A.2.3 Other documents

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 in this Standard and those of the other chapters of Eiffel: The Language, this Standard shall take precedence.
A.3 COMPARABILITY CONDITIONS

A.3.1 Definitions

A.3.1.1 Required Classes
In this Standard, the phrase “Required Classes” denotes a set of classes whose names are those listed in section A.4.

A.3.1.2 Required Flatshort Form
In this Standard, the phrase “Required Flatshort Forms” denotes the flatshort forms given for the Required Classes in section A.4.

A.3.1.3 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 A.3 of this Standard. [The term “Ancestry” is used rather than “Inheritance” because the required links may be implemented by indirect rather than direct inheritance.]

A.3.2 Kernel compatibility
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:

A.3.2.1 • For each of the Required Classes, the implementation includes a class with the same name.

A.3.2.1.1 • All the Required Ancestry Links are present between these classes.

A.3.2.1.2 • The flatshort form of each one of these classes is Flatshort-Compatible with the corresponding Required Flatshort Form.

A.3.2.1.3 • All the dependents of the Required Classes in the implementation are also included in the implementation.

A.3.2.1.4 • 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 A.3.2.1.3 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.]
A.3.3.1.9 • In case of precondition redeclaration, the successive preconditions shall appear as a single Precondition clause, separated by semicolons.
A.3.3.1.10 • In case of postcondition redeclaration, the successive preconditions shall appear as a single Postcondition clause, separated by and then.

A.3.4 Flatshort Compatibility

A.3.4.1 Definition

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 A.6, where both flatshort forms follow the conventions of section A.3.3, belongs to one of the following eleven categories:

A.3.4.1.1 • 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.)
A.3.4.1.2 • An invariant clause that appears in \(ic\) but not in \(sc\).
A.3.4.1.3 • For a feature that appears in both \(ic\) and \(sc\), a postcondition clause that appears in \(ic\) but not in \(sc\).
A.3.4.1.4 • 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.
A.3.4.1.5 • 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.3.4.1.6 • A difference between the Tag_mark of an Assertion_clause in \(ic\) and its counterpart in \(sc\).
A.3.4.1.7 • 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.
A.3.4.1.8 • For a feature that appears in both \(ic\) and \(sc\), a result type in \(ic\) that is different from the corresponding type in \(sc\) but conforms to it.
A.3.4.1.9 • 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 \(sc\).
A.3.4.1.10 • A Note_entry that appears in \(ic\) but not in \(sc\).
A.3.4.1.11 • 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 A.3.4.1.11, the division of classes into one Feature_clause or more, and the labels of these classes, appear in this document for the sole purpose of readability and ease of of reference, but are not part of this Standard.][33]
[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.]}

A.4 REQUIRED CLASSES

The Required Classes are the following thirty classes [ordered from the general to the specific, as in section A.6]:

A.4.1 • ANY [flatshort form in section A.6.1].
A.4.2 • TYPE [flatshort form in section A.6.2].
A.4.3 • PART_COMPATARABLE [flatshort form in section A.6.3].
A.4.4 • COMPARABLE [flatshort form in section A.6.4].
A.4.5 • HASHABLE [flatshort form in section A.6.5].
A.4.6 • NUMERIC [flatshort form in section A.6.6].
A.4.7 • INTERVAL [flatshort form in section A.6.7].
A.4.8 • BOOLEAN [flatshort form in section ].
A.4.9 • CHARACTER [flat short form in section A.6.9].
A.4.10 • INTEGER_GENERAL [flatshort form in A.6.10].
A.4.11 • INTEGER [flatshort form in section A.6.11].
A.4.12 • INTEGER_8 [flatshort form in section A.6.12].
A.4.13 • INTEGER_16 [flatshort form in section A.6.13].
A.4.14 • INTEGER_64 [flatshort form in section A.6.14].
A.4.15 • REAL_GENERAL [flatshort form in A.6.15].
A.4.16 • REAL [flatshort form in section A.6.16].
A.4.17 • POINTER [flatshort form in section A.6.18].
A.4.18 • ARRAY [flatshort form in section A.6.19].
A.4.19 • ANONYMOUS [flatshort form in section A.6.20].
A.4.20 • STRING [flatshort form in section A.6.21].
A.4.21 • STD_FILES [flatshort form in section A.6.22].
A.4.22 • FILE [flatshort form in section A.6.23].
A.5 REQUIRED ANCESTORY LINKS

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

A.5.1 • Every Required Class is a descendant of ANY.
A.5.2 • COMPARABLE is a proper descendant of PART_COMPARABLE.
A.5.3 • TYPE is a proper descendant of PART_COMPARABLE.
A.5.4 • BOOLEAN is a proper descendant of HASHABLE.
A.5.5 • CHARACTER is a proper descendant of COMPARABLE.
A.5.6 • CHARACTER is a proper descendant of HASHABLE.
A.5.7 • FILE is a proper descendant of MEMORY.
A.5.8 • FUNCTION [BASE, OPEN_ARGS, RESULT_TYPE] is a proper descendant of ROUTINE [BASE, OPEN_ARGS].
A.5.9 • INTEGER is a proper descendant of INTEGER_GENERAL.
A.5.10 • INTEGER_8 is a proper descendant of INTEGER_GENERAL.
A.5.11 • INTEGER_16 is a proper descendant of INTEGER_GENERAL.
A.5.12 • INTEGER_64 is a proper descendant of INTEGER_GENERAL.
A.5.13 • INTEGER_GENERAL is a proper descendant of COMPARABLE.
A.5.14 • INTEGER_GENERAL is a proper descendant of HASHABLE.
A.5.15 • INTEGER_GENERAL is a proper descendant of NUMERIC.
A.5.16 • POINTER is a proper descendant of HASHABLE.
A.5.17 • PREDICATE [BASE, OPEN_ARGS] is a proper descendant of FUNCTION [BASE, OPEN_ARGS, BOOLEAN].
A.5.18 • PROCEDURE [BASE, OPEN_ARGS] is a proper descendant of ROUTINE [BASE, OPEN_ARGS].
A.5.19 • REAL_GENERAL is a proper descendant of COMPARABLE.
A.5.20 • REAL_GENERAL is a proper descendant of HASHABLE.
A.5.21 • REAL_GENERAL is a proper descendant of COMPARABLE.
A.5.22 • REAL is a proper descendant of REAL_GENERAL.
A.5.23 • STRING is a proper descendant of COMPARABLE.
A.5.24 • STRING is a proper descendant of HASHABLE.
A.5.25 • STRING is a proper descendant of HASHABLE.
A.5.26 • STRING is a proper descendant of HASHABLE.

["Proper descendant" is a transitive relation, so that for example INTEGER_8 is a descendant of COMPARABLE as a result of A.5.10 and A.5.13.]

A.6 SHORT FORMS OF REQUIRED CLASSES

The following pages (sections A.6.1 to A.6.33) contain the short forms of the required classes as defined in preceding sections.
A.6.1 CLASS ANY

note

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

class interface

ANY

feature -- Access

type: TYPE [like Current]
-- Generating type of current object
-- (type of which it is a direct instance)
one: ONCE_MANAGER
-- Handle on the state of the system’s once routines

feature -- Comparison

is_equal (other: like Current): BOOLEAN
-- Is other attached to an object considered equal
-- to current object?
-- The object comparison operator ~ relies on this function.
ensure
same_type: Result implies same_type (other)
symmetric: Result = other.is_equal (Current)
consistent: default_is_equal (other) implies Result

frozen default_is_equal (other: ? ANY): BOOLEAN
-- Are some and other attached to isomorphic
-- structures made of field-by-field equal objects?
ensure
shallow_implies_deep: default_is_equal (other)
implies Result
only_if_same_type: Result implies same_type (other)
symmetric: Result implies other.is_deep_equal (Current)

feature {NONE} -- Duplication

frozen cloned: like Current
-- New object equal to current one.
ensure
equal: Result ~ Current

frozen default_cloned: like Current
-- New object field-by-field identical to current object
ensure
identical_result: default_is_equal (Result)

frozen default_copy (other: like Current)
-- Copy every field of other onto corresponding field
-- of current object.
require
type_identity: same_type (other)
ensure
made_identical: default_is_equal (other)

frozen deep_cloned: like Current
-- New object structure recursively duplicated from
-- current object
ensure
deep_equal: deep_is_equal (Result)

feature -- Basic operations

default_rescue
-- Handle exception if no Rescue clause.
-- (Default: do nothing.)
frozen do_nothing
-- Execute a null action.

feature -- Output

io: STD_FILES
-- Handle to standard file setup
out: STRING
-- New string containing terse printable
-- representation of current object

invariant
reflexive_default_equality: default_is_equal (Current)
reflexive_equality: Current ~ Current

end
A.6.2 CLASS TYPE

note

description: "[Object]

    Objects describing types conforming to G.

    ]"

class interface

    TYPE [G]

feature -- Access

    adapted alias "]" (x: G) : G
    -- Value of x, adapted if necessary to type G
    -- through conformance or conversion
    ensure
        consistent: Result = x

class_name: STRING
    -- Human-readable form of name of base class
    -- (newly created result for every call)

default: G
    -- Default value of this type
    ensure
        consistent: Result.type ~ Current

hash_code: INTEGER
    -- Hash code value
    ensure
        good_hash_value: Result >= 0

name: STRING
    -- Human-readable form of this type's name
    -- (newly created result for every call)

up_to alias ".." (other: TYPE [ANY]) : INTERVAL [TYPE [ANY]]
    -- Interval containing all types t in system such that
    -- Current <= t and t <= other

feature -- Comparison

    conforms_to alias "<" (other: TYPE [ANY]) : BOOLEAN
    -- Does current type conform to other?

    is_equal (other: TYPE [ANY]) : BOOLEAN
    -- Is current type identical to other?
    -- The object comparison operator ~ relies on this function.
    ensure
        conformance_both_ways:
            Result = conforms_to (other) and
            other.conforms_to (Current)
        yes_if_both_empty_regardless_of_bounds:
            is_empty and other.is_empty imply Result

end
A.6.3 CLASS PART_COMPARABLE

note

description: "[Objects that may be compared according to a partial order relation]
"math: "The model is a partial order relation."

code: "The basic operation is "<" (less than); others are defined in terms of this operation and is_equal."

defined in terms of this operation and is_equal.

deferred class interface

PART_COMPARABLE

feature -- Access

up_to alias "." (other: PART_COMPARABLE): INTERVAL [PART_COMPARABLE]
-- Interval containing all values \(t\), if any, such that \(Current \leq t\) and \(t \leq other\)

feature -- Comparison

is_comparable "(other: like Current): BOOLEAN
-- Do current object and other figure in the relation?

defined in terms of this operation and is_equal.

definition: Result = (Current < other) or (Current = other) or (Current > other)
symmetric: Result = other.is_comparable (Current)

is_less alias "<" (other: like Current): BOOLEAN
-- Is current object less than other?

definition: Result = (Current < other)
only_if_comparable: Result implies is_comparable (other)

is_less_equal alias "\leq\" (other: like Current): BOOLEAN
-- Is current object less than or equal to other?

definition: Result = (Current < other) or (Current = other)
only_if_comparable: Result implies is_comparable (other)

is_greater_equal alias ">=\" (other: like Current): BOOLEAN
-- Is current object greater than or equal to other?

definition: Result = (other \leq Current)

is_greater alias ">\" (other: like Current): BOOLEAN
-- Is current object greater than other?

definition: Result = (other < Current)
only_if_comparable: Result implies is_comparable (other)

is_equal (other: like Current): BOOLEAN
-- Is other attached to an object considered equal
-- to current object?
-- The object comparison operator ~ relies on this function.

definition: Result implies other.is_equal (Current)
consistent: default_is_equal (other) implies Result

max (other: like Current): like Current
-- The greater of current object and other

require
comparable: is_comparable (other)

ensure

current_if_not_smaller: (Current >= other) implies (Result = Current)
other_if_smaller: (Current < other) implies (Result = other)

min (other: like Current): like Current
-- The smaller of current object and other

require
comparable: is_comparable (other)

ensure

current_if_not_greater: (Current <= other) implies (Result = Current)
other_if_greater: (Current > other) implies (Result = other)

three_way_comparison (other: like Current): INTEGER
-- If current object equal to other, 0;
-- if smaller, -1; if greater, 1.

require
comparable: is_comparable (other)

ensure

equal_zero: (Result = 0) = (Current = other)
smaller_negative: (Result = -1) = (Current < other)
greater_positive: (Result = 1) = (Current > other)

invariant

irreflexive_comparison: not (Current < Current)
A.6.4 CLASS COMPARABLE

**note**

description: "
Objects such that any two can be compared through to a total order relation 
"

math: "The model is a total order relation."

comment: ["The basic operation is "<" (less than); others are defined in terms of this operation and is_equal.]

**deferred class interface**

**COMPAREABLE**

**feature** -- Access

up_to alias "." (other: COMPARABLE): INTERVAL [COMPAREABLE]
-- Interval containing all values \( t \), if any, such that
-- \( \text{Current} \leq t \) and \( t \leq \text{other} \)
-- Empty if \( \text{Current} > \text{other} \)

**feature** -- Comparison

is_comparable " (other: like Current): BOOLEAN
-- Do current object and other figure in the relation?
-- (From PART_COMPARABLE); here always true
-- for a total order

ensure

  total_order: Result = True

is_less alias "<" (other: like Current): BOOLEAN
-- Is current object less than other?

**deferred**

ensure

  asymmetric: Result implies not (other < Current)

is_less_equal alias "\leq\" (other: like Current): BOOLEAN
-- Is current object less than or equal to other?

ensure

  definition: Result = ((Current < other) or (Current = other))

is_greater_equal alias "\geq\" (other: like Current): BOOLEAN
-- Is current object greater than or equal to other?

ensure

  definition: Result = (other <= Current)

is_greater alias ">" (other: like Current): BOOLEAN
-- Is current object greater than other?

ensure

  definition: Result = (other < Current)

is_equal (other: like Current): BOOLEAN
-- Is other attached to an object considered equal to current object?
-- The object comparison operator \( \sim \) relies on this function.

**ensure**

  symmetric: Result implies other. is_equal (Current)
  consistent: default_is_equal (other) implies Result
  trichotomy: Result = (not (Current < other) and not (other < Current))

max (other: like Current): like Current
-- The greater of current object and other

ensure

  current_if_not_smaller: (Current >= other) implies (Result = Current)
  other_if_smaller: (Current < other) implies (Result = other)

min (other: like Current): like Current
-- The smaller of current object and other

ensure

  current_if_not_greater: (Current <= other) implies (Result = Current)
  other_if_greater: (Current > other) implies (Result = other)

three_way_comparison (other: like Current): INTEGER
-- If current object equal to other, 0;
-- if smaller, -1; if greater, 1.

ensure

  equal_zero: (Result = 0) = (Current \sim other)
  smaller_negative: (Result = -1) = (Current < other)
  greater_positive: (Result = 1) = (Current > other)

**invariant**

irreflexive_comparison: not (Current < Current)

end
A.6.5 CLASS HASHABLE

note

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

defered class interface

HASHABLE

feature -- Access

hash_code: INTEGER

-- Hash code value

defered

ensure
good_hash_value: Result >= 0

end
A.6.6 CLASS NUMERIC

note
description: "[Objects to which numerical operations are applicable]

math: "The model is a commutative ring."

defered class interface

NUMERIC

feature -- Access

one: like Current
-- Neutral element for "*" and "/"
defered

zero: like Current
-- Neutral element for "+" and "-"
defered

feature -- Status report

divisible (other: like Current): BOOLEAN
-- May current object be divided by other?
defered

exponentiable (other: NUMERIC): BOOLEAN
-- May current object be elevated to the power other?
defered

feature -- Basic operations

plus alias "+" (other: like Current): like Current
-- Sum with other (commutative).
defered

ensure

commutative: equal (Result, other + Current)

minus alias "-" (other: like Current): like Current
-- Result of subtracting other
defered

ensure

consistent: Result + other = Current

product alias "*" (other: like Current): like Current
-- Product by other
defered

divided alias "/" (other: like Current): like Current
-- Division by other

require

good_divisor: divisible (other)
defered

power alias "^" (other: NUMERIC): NUMERIC
-- Current object to the power other

require

good_exponent: exponentiable (other)
defered

identity alias "+": like Current
-- Unary plus
defered

negated alias "-": like Current
-- Unary minus
defered

invariant

neutral_addition: equal (Current + zero, Current)
self_subtraction: equal (Current – Current, zero)
normal_multiplication: equal (Current * one, Current)
self_division: divisible (Current) implies equal
(Current / Current, one)
end
A.6.7 CLASS INTERVAL

note

description: "Sets of values, from a partially or totally ordered set \( G \), all between two given bounds."

class interface

\( \text{INTERVAL} [G \rightarrow \text{PART_COMPARABLE}] \)

create

\( \text{make} (l, u; G) \)

-- Set bounds to \( l \) and \( u \); make interval empty if \( l > u \).

require

comparable: \( l, \text{is_comparable} (u) \)

ensure

\( \text{lower_set: lower} = l \)
\( \text{lower_set: upper} = u \)

feature -- Initialization

\( \text{make} (l, u; G) \)

-- Set bounds to \( l \) and \( u \); make interval empty if \( l > u \).

require

comparable: \( l, \text{is_comparable} (u) \)

ensure

\( \text{lower_set: lower} = l \)
\( \text{lower_set: upper} = u \)

feature -- Access

\( \text{lower: G} \)

-- Lower bound

\( \text{upper: G} \)

-- Upper bound

feature -- Comparison

\( \text{is_comparable "" (other: like Current): BOOLEAN} \)

-- Is either one of current interval and \( other \)

-- strictly contained in the other?

ensure

definition: \( \text{Result} = (\text{Current} < \text{other}) \text{ or } ((\text{Current} \sim \text{other}) \text{ or } (\text{Current} > \text{other}) \)

\( \text{is_subinterval alias "<" (other: like Current): BOOLEAN} \)

-- Is current interval strictly included in \( other \)?

defered

ensure

definition: \( \text{Result} = \text{lower} > \text{other}.\text{lower and upper} < \text{other}.\text{upper} \)

\( \text{is_superinterval alias ">" (other: like Current): BOOLEAN} \)

-- Does current interval strictly include \( other \)?

ensure

definition: \( \text{Result} = (\text{other} < \text{Current}) \)

… OTHER COMPARISON FEATURES

AS IN CLASS \( \text{PART_COMPARABLE} \) …

feature -- Status report

\( \text{is_empty: BOOLEAN} \)

-- Does interval contain no values?

invariant

consistent: \( \text{lower.is_comparable (upper)} \)
empty_if_no_values: \( \text{is_empty} = (\text{lower} > \text{upper}) \)

end
A.6.8 CLASS BOOLEAN

note

description: “Truth values with boolean operations”

expanded class interface

BOOLEAN

feature -- Access

hash_code: INTEGER
--- Hash code value
--- (From HASHABLE.)

ensure
good_hash_value: Result >= 0

feature -- Basic operations

conjuncted alias "and" (other: BOOLEAN): BOOLEAN
--- Boolean conjunction with other

ensure
de_morgan: Result = not (not Current or (not other))
commutative: Result = (other and Current)
consistent_with_semi_strict: Result implies (Current and then other)

conjuncted_semistrict alias "and then" (other: BOOLEAN): BOOLEAN
--- Boolean semi-strict conjunction with other

ensure
de_morgan: Result = not (not Current or else (not other))

implication alias "implies" (other: BOOLEAN): BOOLEAN
--- Boolean implication of other
--- (semi-strict)

ensure
definition: Result = (not Current or else other)

negated alias "not" : BOOLEAN
--- Negation.

disjuncted alias "or" (other: BOOLEAN): BOOLEAN
--- Boolean disjunction with other

ensure
de_morgan: Result = not (not Current and (not other))
commutative: Result = (other or Current)
consistent_with_semi_strict: Result implies (Current or else other)

disjuncted_semistrict alias "or else" (other: BOOLEAN): BOOLEAN
--- Boolean semi-strict disjunction with other

ensure
de_morgan: Result = not (not Current and then (not other))

disjuncted_exclusive alias "xor" (other: BOOLEAN): BOOLEAN
--- Boolean exclusive or with other

ensure
definition: Result = ((Current or other) and not (Current and other))

feature -- Output

out: STRING
--- Printable representation of boolean

invariant

involutive_negation: Current ~ (not (not Current))
non_contradiction: not (Current and (not Current))
excluded_middle: Current or (not Current)

end
A.6.9 CLASS CHARACTER

note
description: "[Characters, with comparison operations and an ASCII code]"

expanded class interface

CHARACTER

feature -- Access
code: INTEGER
-- Associated integer value
hash_code: INTEGER
-- Hash code value
-- (From HASHABLE.)
ensure
good_hash_value: Result >= 0

up_to alias ".." (other: CHARACTER):
INTERVAL [CHARACTER]
-- Interval containing all characters c, if any, such that
-- Current <= c and c <= other
-- Empty if Current > other

feature -- Comparison
is_less alias "<" (other: like Current): BOOLEAN
-- Is current object less than other?
-- (From COMPARABLE.)
ensure
asymmetric: Result implies not (other < Current)

is_less_equal alias "<=" (other: like Current):
BOOLEAN
-- Is current object less than or equal to other?
-- (From COMPARABLE.)
ensure
definition: Result = (Current < other) or (Current = other)

is_greater_equal alias ">=" (other: like Current):
BOOLEAN
-- Is current object greater than or equal to other?
-- (From COMPARABLE.)
ensure
definition: Result = (other <= Current)

is_greater alias ">" (other: like Current): BOOLEAN
-- Is current object greater than other?
-- (From COMPARABLE.)
ensure
definition: Result = (other < Current)

max (other: like Current): like Current
-- The greater of current object and other
-- (From COMPARABLE.)
ensure
current_if_not_smaller: (Current >= other) implies (Result = Current)
other_if_smaller: (Current < other) implies (Result = other)

min (other: like Current): like Current
-- The smaller of current object and other
-- (From COMPARABLE.)
ensure
current_if_not_greater: (Current <= other) implies (Result = Current)
other_if_greater: (Current > other) implies (Result = other)

three_way_comparison (other: like Current):
INTEGER
-- If current object equal to other, 0;
-- if smaller, -1; if greater, 1.
-- (From COMPARABLE.)
ensure
equal_zero: (Result = 0) = (Current ~ other)
smaller: (Result = -1) = Current < other
greater_positive: (Result = 1) = Current > other

feature -- Output
out: STRING
-- Printable representation of character
-- (From ANY.)

invariant
irreflexive_comparison: not (Current < Current)

end
A.6.10 CLASS INTEGER_GENERAL

note
description: "Integer values of set size"

class interface
    INTEGER_GENERAL
create
    make (b: INTEGER)
        -- Initialize with bit size b.
        -- (No effect on expanded targets.)
        require
            positive: b > 0
        ensure
            bit_size_set: bit_size = b
    default_create
        -- Initialize with default bit size: 32.
        ensure
            bit_size_set: bit_size = Default_bit_size
from_integer convert (other: INTEGER_GENERAL)
    -- Initialize from other; do not lose any precision.
    ensure
        bit_size_set: bit_size = Default_bit_size
feature -- Access
    bit_size: INTEGER
        -- Number of bits in representation
    Default_bit_size: INTEGER
        -- Number of bits in representation
    hash_code: INTEGER
        -- Hash code value
        -- (From HASHABLE.)
    ensure
        good_hash_value: Result >= 0
    one: like Current
        -- Neutral element for "+" and "−"
        -- (From NUMERIC.)
    value: Result = 1
    sign: INTEGER
        -- Sign value (0, −1 or 1)
    ensure
        three_way: Result = three_way_comparison (zero)
    up_to alias ".." (other: INTEGER_GENERAL) : INTERVAL [INTEGER_GENERAL]
        -- Interval containing all integers i, if any, such that
        -- Current ≤ i and i ≤ other
        -- Empty if Current > other
    zero: like Current
        -- Neutral element for "+" and "−"
        -- (From NUMERIC.)
    value: Result = 0
feature -- Comparison
    is_less alias "<" (other: like Current): BOOLEAN
        -- Is other greater than current integer?
        -- (From COMPARABLE.)
    ensure
        asymmetric: Result implies not (other < Current)
    is_less_equal alias "≤" (other: like Current): BOOLEAN
        -- Is current object less than or equal to other?
        -- (From COMPARABLE.)
    ensure
        definition: Result = (Current < other) or (Current = other)
    is_greater_equal alias "≥" (other: like Current): BOOLEAN
        -- Is current object greater than or equal to other?
        -- (From COMPARABLE.)
    ensure
        definition: Result = (other <= Current)
    is_greater alias ">" (other: like Current): BOOLEAN
        -- Is current object greater than other?
        -- (From COMPARABLE.)
    ensure
        definition: Result = (other < Current)
    max (other: like Current): like Current
        -- The greater of current object and other
        -- (From COMPARABLE.)
    ensure
        current_if_not_smaller: (Current >= other) implies (Result = Current)
        other_if_smaller: (Current < other) implies (Result = other)
    min (other: like Current): like Current
        -- The smaller of current object and other
        -- (From COMPARABLE.)
    ensure
        current_if_not_greater: (Current <= other) implies (Result = Current)
        other_if_greater: (Current > other) implies (Result = other)
three_way_comparison (other, like Current):
  INTEGER
  -- If current object equal to other, 0;
  -- if smaller, -1; if greater, 1.
  -- (From COMPARABLE.)
ensure
equal_zero: (Result = 0) = (Current ~ other)
smaller: (Result = 1) = Current < other
greater_positive: (Result = -1) = Current > other

feature -- Basic operations

abs (like Current)
  -- Absolute value
ensure
  non_negative: Result >= 0
  same_absolute_value: (Result = Current) or (Result = -Current)

product alias "*" (other, like Current)
  -- Product by other
  -- (From NUMERIC.)

plus alias "+" (other, like Current)
  -- Sum with other
  -- (From NUMERIC.)

divide alias "/" (other, like Current)
  -- Division by other
ensure
  commutative: equal (Result, other + Current)

feature -- Status report

divisible (other, like Current): BOOLEAN
  -- May current object be divided by other?
  -- (From NUMERIC.)
ensure
  value: Result = (other /= 0)

exponentiable (other, NUMERIC): BOOLEAN
  -- May current object be elevated to the power other?
  -- (From NUMERIC.)
ensure
  safe_values: (other, conforms_to (Current) or (other, conforms_to (0, 0) and (Current >= 0)))
  implies Result

bit_one (n, INTEGER): BOOLEAN
  -- Is n-th bit (from left, in binary representation)
  -- a one?
require
  at_most_size: n <= bit_size
  at_least_one: n > 0

feature -- Element change

bit_shift (n, INTEGER): like Current
  -- Bit-shift n positions, to right if positive,
  -- left otherwise.
require
  at_most_size: n <= bit_size
  at_least_minus_size: n >= -bit_size

bit_shift_left (n, INTEGER): like Current
  -- Bit-shift n positions to left.
require
  non_negative: n >= 0
  at_most_size: n <= bit_size

bit_shift_right (n, INTEGER): like Current
  -- Bit-shift n positions to right.
require
  non_negative: n >= 0
  at_most_size: n <= bit_size

bit_or (i, like Current): like Current
  -- Bitwise or with i.

instantiation alias "+" (like Current)
  -- Unary plus
  -- (From NUMERIC.)

negated alias "-" (like Current)
  -- Unary minus
  -- (From NUMERIC.)

power alias "^" (other, NUMERIC): REAL
  -- Integer power of Current by other
  -- (From "^" in NUMERIC.)
require
  good_divisor: divisible (other)

quotient alias "/" (other, like Current): like Current
  -- Integer division of Current by other
  -- (From "/" in NUMERIC.)
require
  good_divisor: divisible (other)

remainder alias "\" (other, like Current): like Current
  -- Remainder of integer division of Current by other
  -- (From "\" in NUMERIC.)
require
  good_divisor: divisible (other)
bit_xor (i: like Current): like Current
-- Bitwise exclusive or with i.

bit_not: like Current
-- One’s complement.

feature -- Output
  out: STRING
    -- Printable representation of current object
    -- (From ANY.)

invariant
  bit_size_positive: bit_size > 0
  default_bit_size_positive: default_bit_size > 0
  irreflexive_comparison: not (Current < Current)
  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
A.6.11 CLASS INTEGER

note
description: "32-bit integer values"
expanded class interface
   INTEGER
create
default_create
   -- Initialize with default bit size: 32.
   ensure
      bit_size_set: bit_size = 32
from_integer convert (b: INTEGER_GENERAL)
   -- Initialize from other, losing leftmost part if
   -- other is of smaller bit size.
   ensure
      bit_size_set: bit_size = Default_bit_size
feature
   … SAME FEATURE SPECIFICATIONS
AS CLASS INTEGER_GENERAL, …
invariant
   … SAME INVARIANT CLAUSES
AS CLASS INTEGER_GENERAL, PLUS:

   bit_size_definition: bit_size = 32
end
A.6.12 CLASS INTEGER_8

note
description: "8-bit integer values"

expanded class interface
  INTEGER_8

create
default_create
  -- Initialize with default bit size: 8.
  ensure
    bit_size_set: bit_size = 8
from_integer (other: INTEGER_GENERAL)
  -- Initialize from other, losing leftmost part if
  -- other is of smaller bit size.
  ensure
    bit_size_set: bit_size = Default_bit_size

feature
  … SAME FEATURE SPECIFICATIONS
  AS CLASS INTEGER_GENERAL …

invariant
  … SAME INVARIANT CLAUSES
  AS CLASS INTEGER_GENERAL, PLUS:
  bit_size_definition: bit_size = 8
A.6.13 CLASS INTEGER_16

note
description: "16-bit integer values"

expanded class interface
INTEGER_16

create
default_create
  -- Initialize with default bit size: 16.
ensure
  bit_size_set: bit_size = 16

from_integer convert (other: INTEGER_GENERAL)
  -- Initialize from other, losing leftmost part if
  -- other is of smaller bit size.
ensure
  bit_size_set: bit_size = Default_bit_size

feature
  … SAME FEATURE SPECIFICATIONS
  AS CLASS INTEGER_GENERAL, …

invariant
  … SAME INVARIANT CLAUSES
  AS CLASS INTEGER_GENERAL, PLUS:
  bit_size_definition: bit_size = 16

end
A.6.14 CLASS INTEGER_64

note
description: "64-bit integer values"

expanded class interface
  INTEGER_64

create
  default_create
    -- Initialize with default bit size: 64.
  ensure
    bit_size_set: bit_size = 64

from_integer convert (other: INTEGER_GENERAL)
  -- Initialize from other, losing leftmost part if
  -- other is of smaller bit size.
  ensure
    bit_size_set: bit_size = Default_bit_size

feature
  ... SAME FEATURE SPECIFICATIONS
  AS CLASS INTEGER_GENERAL ...

invariant
  ... SAME INVARIANT CLAUSES
  AS CLASS INTEGER_GENERAL, PLUS:
    bit_size_definition: bit_size = 64

end
A.6.15 CLASS REAL_GENERAL

A.6.15 CLASS REAL_GENERAL

note
description: "Real values, single precision"

expanded class interface

REAL

feature -- Access

hash_code: INTEGER
-- Hash code value
-- (From HASHABLE.)

ensure
good_hash_value: Result >= 0

one: like Current
-- Neutral element for "*" and "j"
-- (From NUMERIC.)

ensure
value: Result = 1.0

sign: INTEGER
-- Sign value (0, –1 or 1)

ensure
three_way: Result = three_way_comparison (zero)

up_to alias "." (other: REAL_GENERAL):
INTERVAL [REAL_GENERAL]
-- Interval containing all reals r, if any, such that
-- Current <= r and r <= other
Empty if Current > other

zero: like Current
-- Neutral element for "*" and "j"
-- (From NUMERIC.)

ensure
value: Result = 0.0

feature -- Comparison

is_less alias "<" (other: like Current): BOOLEAN
-- Is other greater than current real?
-- (From COMPARABLE.)

ensure
asymmetric: Result implies not (other < Current)

is_less_equal alias "<=" (other: like Current):
BOOLEAN
-- Is current object less than or equal to other?
-- (From COMPARABLE.)

ensure
definition: Result = (Current < other) or (Current = other)

is_greater_equal alias ">=" (other: like Current):
BOOLEAN
-- Is current object greater than or equal to other?
-- (From COMPARABLE.)

ensure
definition: Result = (other <= Current)

is_greater alias ">" (other: like Current): BOOLEAN
-- Is current object greater than other?
-- (From COMPARABLE.)

ensure
definition: Result = (other < Current)

max (other: like Current): like Current
-- The greater of current object and other
-- (From COMPARABLE.)

ensure
current_if_not_smaller: (Current >= other) implies (Result = Current)
other_if_smaller: (Current < other) implies (Result = other)

min (other: like Current): like Current
-- The smaller of current object and other
-- (From COMPARABLE.)

ensure
current_if_not_greater: (Current <= other) implies (Result = Current)
other_if_greater: (Current > other) implies (Result = other)

three_way_comparison (other: like Current):
INTEGER
-- If current object equal to other, 0;
-- if smaller, –1; if greater, 1.
-- (From COMPARABLE.)

ensure
equal_zero: (Result = 0) = (Current ~ other)
greater_positive: (Result = 1) = Current > other

feature -- Status report

divisible (other: like Current): BOOLEAN
-- May current object be divided by other?
-- (From NUMERIC.)

ensure
not_exact_zero: Result implies (other /= 0 , 0)

exponentiable (other: NUMERIC): BOOLEAN
-- May current object be elevated to the power other?
-- (From NUMERIC.)

ensure
safe_values: (other, conforms_to (0) or (other, conforms_to (0) and (Current >= 0 , 0)) implies Result
feature -- Conversion

ceiling: INTEGER
  -- Smallest integral value no smaller than
  -- current object
ensure
  result_no_smaller: Result >= Current
  close_enough: Result – Current < one

floor: INTEGER
  -- Greatest integral value no greater than
  -- current object
ensure
  result_no_greater: Result <= Current
  close_enough: Current – Result < one

rounded: INTEGER
  -- Rounded integral value
ensure
  definition: Result = sign * ((abs + 0.5) . floor)

feature -- Basic operations

abs: like Current
  -- Absolute value
ensure
  non_negative: Result >= 0
  same_absolute_value: (Result = Current) or (Result = –Current)

product alias "*" (other: like Current): like Current
  -- Product by other
  -- (From NUMERIC.)

plus alias "+" (other: like Current): like Current
  -- Sum with other
  -- (From NUMERIC.)
ensure
  commutative: equal (Result, other + Current)

minus alias "-" (other: like Current): like Current
  -- Result of subtracting other
  -- (From NUMERIC.)
ensure
  consistent: Result + other = Current

divided alias "/" (other: like Current): like Current
  -- Division by other
  -- (From NUMERIC.)
require
  good_divisor: divisible (other)

power alias "^" (other: NUMERIC): REAL
  -- Current real to the power other
  -- (From NUMERIC.)
require
  good_exponent: exponentiable (other)

identity alias "+": like Current
  -- Unary plus
  -- (From NUMERIC.)

negated alias "-": like Current
  -- Unary minus
  -- (From NUMERIC.)

feature -- Output

out: STRING
  -- Printable representation of real value
  -- (From ANY.)

invariant
  irreflexive_comparison: not (Current < Current)
  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)
A.6.16 CLASS *REAL*

**note**
- description: "32-bit real values"

**expanded class interface**

REAL

**feature**

- … SAME FEATURE SPECIFICATIONS
- AS CLASS *REAL GENERAL* …

**end**
A.6.17 CLASS TYPED_POINTER
A.6.18 CLASS POINTER

note
description: 
References to objects meant to be exchanged with non-Eiffel software

expanded class interface
POINTER

feature -- Access
hash_code: INTEGER
-- Hash code value
-- (From HASHABLE.)
ensure
good_hash_value: Result \geq 0

feature -- Basid operations
plus alias "+" (offset: INTEGER): POINTER
-- Pointer to address at current position plus
-- offset bytes

feature -- Output
out: STRING
-- Printable representation of pointer value
-- (From ANY.)

end
A.6.19 CLASS ARRAY

**note**

description: "Sequences of values, all of the same type or of a conforming one, accessible through integer indices in a contiguous interval"

**class interface**

ARRAY [G]

**create**

make (minindex, maxindex: INTEGER)
-- Allocate array; set index interval to
-- minindex .. maxindex; set all values to default.
-- (Make array empty if minindex > maxindex.)

**ensure**

empty_if_bounds_dont_fit: (minindex > maxindex)
implies (count = 0)

bounds_set: (minindex <= maxindex) implies
((lower = minindex) and (upper = maxindex))

from_interval (int: INTERVAL [INTEGER])
-- Allocate array; set index interval to int;
-- set all values to default.
-- (Make array empty if interval is empty.)

**ensure**

empty_if_bounds_dont_fit: (int.is_empty)
implies (count = 0)

bounds_set: not (int.is_empty) implies
((lower = int.lower) and (upper = int.upper))

**feature** -- Access

item alias "[i]" assign "put" (i: INTEGER): G
-- Entry at index i

**require**

good_key: valid_index (i)

**feature** -- Measurement

bounds: INTERVAL [INTEGER]
-- Integer interval for indices

count: INTEGER
-- Number of available indices

lower: INTEGER
-- Minimum index

upper: INTEGER
-- Maximum index

**feature** -- Status report

valid_index (i: INTEGER): BOOLEAN
-- Is i within the bounds of the array?

**feature** -- Element change

force (v: like item; i: INTEGER)
-- Assign item v to i-th entry.
-- Always applicable: resize the array if i falls out of
-- currently defined bounds; preserve existing items.

**ensure**

inserted: item (i) = v

higher_count: count >= old count

put (v: like item; i: INTEGER)
-- Replace i-th entry, if in index interval, by v.

**require**

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

**invariant**

consistent_size: count = upper – lower + 1

non_negative_count: count >= 0

interval_consistent: bounds ~ lower..upper

end
note
description: "[Tuples: finite sequences of values, each of a specified type]
"

class interface

ANONYMOUS

feature -- Access

item: ANY
    -- i-th element of tuple

require
good_key: valid_index (i)

hash_code: INTEGER
    -- Hash code value
    -- (From HASHABLE.)

ensure
good_hash_value: Result >= 0

feature -- Measurement

count: INTEGER
    -- Minimum member of items in tuple

feature -- Status report

valid_index (i: INTEGER): BOOLEAN
    -- Is i within the bounds of the array?

ensure
ok_if_between_one_and_count:
    ((i >= 1) and (i <= count)) implies Result

feature -- Element change

put (v: ANY; i: INTEGER)
    -- Replace i-th item by v.

require
good_key: valid_index (i)

ensure
replaced: item (i) = v

end
A.6.21 CLASS STRING

note
description: "[Sequences of characters, accessible through integer indices in a contiguous range. ]"

class interface
STRING
create
frozen make (n: INTEGER)
-- Allocate space for at least n characters.
require
non_negative_size: n >= 0
ensure
empty_string: count = 0
from_string (s: STRING)
-- Initialize from the characters of s.
-- (Useful in proper descendants of class STRING,
-- to initialize a string-like object from a manifest string.)
feature -- Initialization
from_c (c_string: POINTER)
-- Reset contents of string from contents of c_string,
-- a string created by some external C function.
frozen remake (n: INTEGER)
-- Allocate space for at least n characters.
require
non_negative_size: n >= 0
ensure
empty_string: count = 0
from_string (s: STRING)
-- Initialize from the characters of s.
-- (Useful in proper descendants of class STRING,
-- to initialize a string-like object from a manifest string.)
feature -- Access
hash_code: INTEGER
-- Hash code value
ensure
good_hash_value: Result >= 0
is_less_equal alias "<=" (other: like Current): BOOLEAN
-- Is current object less than or equal to other?
ensure
definition: Result = (Current < other) or (Current = other)
feature -- Measurement
count: INTEGER
-- Actual number of characters making up the string
occurrences (c: CHARACTER): INTEGER
-- Number of times c appears in the string
ensure
non_negative_occurrences: Result >= 0
feature -- Comparison
is_equal (other: like Current): BOOLEAN
-- Is string made of same character sequence as other?
-- The object comparison operator ~ relies on this function.
is_less alias "<" (other: STRING): BOOLEAN
-- Is string lexicographically lower than other?
-- (From COMPARABLE.)
ensure
asymmetric: Result implies not (other < Current)
other
is_greater_equal alias ">" (other: like Current): BOOLEAN
-- Is current object greater than or equal to other?
-- (From COMPARABLE.)
ensure
definition: Result = (other <= Current)
is_greater alias ">" (other: like Current): BOOLEAN
-- Is current object greater than other?
-- (From COMPARABLE.)
ensure
definition: Result = (other < Current)
max (other: like Current): like Current)
-- The greater of current object and other
-- (From COMPARABLE.)
ensure
current_if_not_smaller: (Current >= other) implies (Result = Current)
other_if_smaller: (Current < other) implies (Result = other)
min (other: like Current): like Current)
-- The smaller of current object and other
-- (From COMPARABLE.)
ensure
current_if_not_greater: (Current <= other) implies (Result = Current)
other_if_greater: (Current > other) implies (Result = other)
three_way_comparison (other: like Current): INTEGER
-- If current object equal to other, 0;
-- if smaller, -1; if greater, 1.
-- (From COMPARABLE.)
ensure
equal_zero: (Result = 0) = (Current ~ other)
smaller: (Result = -1) = Current < other
greater_positive: (Result = 1) = Current > other

feature -- Status report
is_empty: BOOLEAN
-- Does string contain no characters?
valid_index (i: INTEGER): BOOLEAN
-- Is i within the bounds of the string?

feature -- Element change
append_boolean (b: BOOLEAN)
-- Append the string representation of b at end.

put (c: CHARACTER; i: INTEGER)
-- Replace character at position i by c.
require
good_key: valid_index (i)
ensure
append_character (c: CHARACTER)
-- Append c at end.
ensure
item_inserted: item (count) = c
one_more_occurrence: occurrences (c) = old
(occurrences (c)) + 1
item_inserted: has (c)
append_integer (i: INTEGER)
-- Append the string representation of i at end.
append_real (r: REAL)
-- Append the string representation of r at end.
append_string (s: STRING)
-- Append a copy of s at end.
ensure
new_count: count = old count + s.count
-- appended: For every i in 1..s.count,
-- item (old count + i) = s.item (i)
fill (c: CHARACTER)
-- Replace every character with c.
ensure
allblank: For every i in 1..count, item (i) = Blank
head (n: INTEGER)
-- Remove all characters except for the first n;
-- do nothing if n >= count.
require
non_negative_argument: n >= 0
ensure
new_count: count = n, min (old count)
-- first_kept: For every i in 1..n, item (i)=old item (i)
insert (s: like Current; i: INTEGER)
-- Add s to the left of position i.
require
index_small_enough: i <= count
index_large_enough: i > 0
ensure
new_count: count = old count + s.count
insert_character (c: CHARACTER; i: INTEGER)
-- Add c to the left of position i.
ensure
new_count: count = old count + 1
left_adjust
-- Remove leading white space.
ensure
new_count: (count /= 0) implies (item (1) /= ' ')
insertion_done: item (i) = c
put_substring (s: like Current; start_pos, end_pos: INTEGER)
-- Copy the characters of s to positions
-- start_pos .. end_pos.

require
index_small_enough: end_pos <= count
order_respected: start_pos <= end_pos
index_large_enough: start_pos > 0
ensure
new_count: count = old count + s.count – end_pos + start_pos – 1

right_adjust
-- Remove trailing white space.
ensure
new_count: (count /= 0) implies (item (count) /= ' ')

tail (n: INTEGER)
-- Remove all characters except for the last n;
-- do nothing if n >= count.
require
non_negative_argument: n >= 0
ensure
new_count: count = n.min (old count)

feature -- Removal
remove (i: INTEGER)
-- Remove i-th character.
require
index_small_enough: i <= count
index_large_enough: i > 0
ensure
new_count: count = old count – 1

wipe_out
-- Remove all characters.
ensure
empty_string: count = 0
wiped_out: is_empty

feature -- Resizing
resize (newsize: INTEGER)
-- Rearrange string so that it can accommodate
-- at least newsize characters.
-- Do not lose any previously entered character.
require
new_size_non_negative: newsize >= 0

feature -- Conversion
to_integer: INTEGER
-- Integer value;
-- for example, when applied to "123", will yield 123
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 ANY.)
require
new_result_count: count = other.count
ensure
same_characters: For every i in 1..count,
-- item (i) = other.item (i)

substring (n1, n2: INTEGER): like Current
-- Copy of substring containing all characters at indices
-- between n1 and n2
require
meaningful_origin: 1 <= n1
meaningful_interval: n1 <= n2
meaningful_end: n2 <= count
ensure
new_result_count: Result.count = n2 – n1 + 1
original_characters: For every i in 1..n2-n1,
-- Result.item (i) = item (n1+i-1)

feature -- Output
out: STRING
-- Printable representation
-- (From ANY.)

invariant
irreflexive_comparison: not (Current < Current)
empty_definition: is_empty = (count = 0)
non_negative_count: count >= 0

end
CLASS STD_FILES

**note**

- **description**: "Commonly used input and output mechanisms. This class may be used as either ancestor or supplier by classes needing its facilities."]"

**class interface**

`STD_FILES`

**feature** -- Access

- **default_output**: ? FILE
  -- Default output.
- **error**: FILE
  -- Standard error file
- **input**: FILE
  -- Standard input file
- **output**: FILE
  -- Standard output file
- **standard_default**: FILE
  -- default_output if not void,
  -- otherwise output.

**feature** -- Status report

- **last_character**: CHARACTER
  -- Last character read by `read_character`
- **last_integer**: INTEGER
  -- Last integer read by `read_integer`
- **last_real**: REAL
  -- Last real read by `read_real`
- **last_string**: STRING
  -- Last string read by `read_line`
  -- `read_stream`, or `read_word`

**feature** -- Element change

- **put_boolean** (b: BOOLEAN)
  -- Write b at end of default output.
- **put_character** (c: CHARACTER)
  -- Write c at end of default output.
- **put_integer** (i: INTEGER)
  -- Write i at end of default output.
- **put_new_line**
  -- Write line feed at end of default output.
- **put_real** (r: REAL)
  -- Write r at end of default output.
- **put_string** (s: STRING)
  -- Write s at end of default output.

- **set_error_default**
  -- Use standard error as default output.
- **set_output_default**
  -- Use standard output as default output.

**feature** -- Input

- **read_character**
  -- Read a new character from standard input.
  -- Make result available in `last_character`.
- **read_integer**
  -- Read a new integer from standard input.
  -- Make result available in `last_integer`.
- **read_line**
  -- Read a line from standard input.
  -- Make result available in `last_string`.
  -- New line will be consumed but not part of `last_string`.
- **read_real**
  -- Read a new real from standard input.
  -- Make result available in `last_real`.
- **read_stream** (nb_char: INTEGER)
  -- Read a string of at most nb_char bound characters
  -- from standard input.
  -- Make result available in `last_string`.
- **to_next_line**
  -- Move to next input line on standard input.

**end**
A.6.23 CLASS FILE

note
description: "[Files viewed as persistent sequences of characters ]"
class interface
FILE
create
make (fn: STRING)
-- Create file object with fn as file name.
require
string_not_empty: not fn.is_empty
ensure
file_named: name ~ n
file_closed: is_closed
create_read_write (fn: STRING)
-- Create file object with fn as file name
-- and open file for both reading and writing;
-- create it if it does not exist.
require
string_not_empty: not fn.is_empty
ensure
exists: exists
open_read: is_open_read
open_write: is_open_write
open_write (fn: STRING)
-- Create file object with fn as file name
-- and open file for writing;
-- create it if it does not exist.
require
string_not_empty: not fn.is_empty
ensure
exists: exists
open_write: is_open_write
open_read (fn: STRING)
-- Create file object with fn as file name
-- and open file in read mode.
require
string_not_empty: not fn.is_empty
ensure
exists: exists
open_read: is_open_read
open_read_write (fn: STRING)
-- Create file object with fn as file name
-- and open file for both reading and writing.
require
string_not_empty: not fn.is_empty
ensure
exists: exists
open_read: is_open_read
open_write: is_open_write
open_append (fn: STRING)
-- Create file object with fn as file name
-- and open file in append-only mode.
require
string_not_empty: not fn.is_empty
ensure
exists: exists
open_append: is_open_append
open_read (fn: STRING)
-- Create file object with fn as file name
-- and open file in read mode.
require
string_not_empty: not fn.is_empty
ensure
exists: exists
open_read: is_open_read
is_writable: BOOLEAN
   -- Is file writable?
   require
      handle_exists: exists
last_character: CHARACTER
   -- Last character read by read_character
last_integer: INTEGER
   -- Last integer read by read_integer
last_real: REAL
   -- Last real read by read_real
last_string: STRING
   -- Last string read by read_line,
      -- or read_stream, or read_word

feature -- Status setting
   close
      -- Close file.
      require
         medium_is_open: not is_closed
   ensure
      is_closed: is_closed

open_read
   -- Open file in read-only mode.
   require
      is_closed: is_closed
   ensure
      exists: exists
      open_read: is_open_read

open_read_append
   -- Open file in read and write-at-end mode;
      -- create it if it does not exist.
   require
      is_closed: is_closed
   ensure
      exists: exists
      open_read: is_open_read
      open_append: is_open_append

open_read_write
   -- Open file in read and write mode.
   require
      is_closed: is_closed
   ensure
      exists: exists
      open_read: is_open_read
      open_write: is_open_write
      open_write
         -- Open file in write-only mode;
         -- create it if it does not exist.
   ensure
      exists: exists
      open_write: is_open_write

feature -- Cursor movement
   to_next_line
      -- Move to next input line.
      require
         readable: is_readable

feature -- Element change
   change_name (new_name: STRING)
      -- Change file name to new_name
   require
      file_exists: exists
   ensure
      name_changed: name ~ new_name

feature -- Removal
   delete
      -- Remove link with physical file; delete physical
         -- file if no more link.
   require
      exists: exists
      dispose
         -- Ensure this medium is closed when
            -- garbage-collected.

feature -- Input
   read_character
      -- Read a new character.
      -- Make result available in last_character.
   require
      readable: is_readable
   ensure
      readable: is_readable

read_integer
   -- Read the ASCII representation of a new integer
      -- from file. Make result available in last_integer.
   require
      readable: is_readable

read_line
   -- Read a string until new line or end of file.
      -- Make result available in last_string.
      -- New line will be consumed but not part of
         last_string.
   require
      readable: is_readable
read_real
   -- Read the ASCII representation of a new real
   -- from file. Make result available in last_real.

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.

feature -- Output

put_boolean (b: BOOLEAN)
   -- Write ASCII value of b at current position.

require
   extendible: extendible

put_character (c: CHARACTER)
   -- Write c at current position.

require
   extendible: extendible

put_integer (i: INTEGER)
   -- Write ASCII value of i at current position.

require
   extendible: extendible

put_real (r: REAL)
   -- Write ASCII value of r at current position.

require
   extendible: extendible

put_string (s: STRING)
   -- Write s at current position.

require
   extendible: extendible

invariant

name_not_empty: not name.is_empty

writable_if_extendible: extendible implies is_writable

end
A.6.24 CLASS STORABLE

note

description: "[Objects that may be stored and retrieved along with all their dependents]"
usage: "[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_exists: file.exists
file_is_open_read: file.is_open_read
file_not_plain_text: not file.is_plain_text

feature -- Element change

basic_store (file: FILE)
-- Produce on file an external representation of entire
-- object structure reachable from current object.
-- Retrievable within current system only.

require
file_exists: file.exists
file_is_open_write: file.is_open_write
file_not_plain_text: not file.is_plain_text

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_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 object.
-- Retrievable from other systems for the same or other
-- platforms (machine architectures).

require
file_exists: file.exists
file_is_open_write: file.is_open_write
file_not_plain_text: not file.is_plain_text
A.6.25 CLASS MEMORY

note
description: "[
Facilities for tuning up the garbage collection
mechanism
]
usage: "[
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
A.6.26 CLASS EXCEPTIONS

note
description: "[Facilities for adapting the exception handling mechanism]"
usage: "[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 one
      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 feature call on void reference

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

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 -- Status report
assertion_violation: BOOLEAN
   -- Is last exception originally due to a violated
      assertion or non-decreasing variant?
A.6.27 CLASS ARGUMENTS

note
description: "Access to command-line arguments"
usage: "[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 <= 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 started system execution (command name does not count)
ensure
non_negative: Result >= 0

end
A.6.28 CLASS PLATFORM

note

description: "Platform-dependent properties"
usage: "[This class may be used as ancestor by classes needing its facilities.
]"

class interface

PLATFORM

feature -- Access

Boolean_bits: INTEGER
-- Number of bits in a value of type BOOLEAN

ensure
meaningful: Result >= 1

Character_bits: INTEGER
-- Number of bits in a value of type CHARACTER

ensure
meaningful: Result >= 1
large_enough: 2 ^ Result >= Maximum_character_code

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

Minimum_character_code: INTEGER
-- Smallest supported code for CHARACTER values

ensure
meaningful: Result <= 0

Minimum_integer: INTEGER
-- Smallest supported value of type INTEGER

ensure
meaningful: Result <= 0

Pointer_bits: INTEGER
-- Number of bits in a value of type POINTER

ensure
meaningful: Result >= 1

Real_bits: INTEGER
-- Number of bits in a value of type REAL

ensure
meaningful: Result >= 1
A.6.29 CLASS ONCE_MANAGER

note
description: "[
    Controller of keyed once routines
    ]"
usage: "[
    See feature onces in class ANY.
    ]"
class interface
    ONCE_MANAGER
feature -- Status report
    fresh (key: STRING): BOOLEAN
        -- Will the presence of key among a once routine’s
        -- once keys cause execution of the routine’s body?
feature -- Element change
    refresh (key: STRING)
        -- Reset all once routines that use key as once key.
        ensure
            refreshed: fresh (key)
    refresh_all
        -- Reset all once routines.
    refresh_all_except (keys: ARRAY [STRING])
        -- Reset all once routines except those using
        -- any of the items of keys as once keys.
    refresh_some (keys: ARRAY [STRING])
        -- Reset all once routines that use any
        -- of the items of keys as once keys.
end
A.6.30 CLASS ROUTINE

note
description: "[Objects representing delayed calls to a routine, with some operands possibly still open]"
defered class interface
ROUTINE [BASE_TYPE, OPEN_ARGS -> TUPLE]

feature -- Initialization
adapt (other: ROUTINE [ANY, OPEN_ARGS])
-- Initialize from other.
-- Useful in descendants.

feature -- Access
operands: OPEN_ARGS
-- Open operands
target: ANY
-- Target of call

open_operand_type (i: INTEGER): INTEGER
-- Type of i-th open operand.
require
positive: i >= 1
within_bounds: i <= open_count

hash_code: INTEGER
-- Hash code value

precondition (args: like operands) BOOLEAN
-- Do args satisfy routine's precondition
-- in present state?

postcondition (args: like operands) BOOLEAN
-- Does current state satisfy routine's
-- postcondition for args?

feature -- Status report
callable: BOOLEAN
-- Can routine be called on current object?

is_equal (other: like Current): BOOLEAN
-- Is associated routine the same as the one
-- associated with other?
-- The object comparison operator ~ relies on this function.

valid_operands (args: OPEN_ARGS): BOOLEAN
-- Are args valid operands for this routine?

feature -- Measurement
open_count: INTEGER
-- Number of open parameters.

feature -- Element change
set_operands (args: OPEN_ARGS)
-- Use args as operands for next call.
require
valid_operands: valid_operands (args)

feature -- Duplication
copy (other: like Current)
-- Use same routine as other.

feature -- Basic operations
call (args: OPEN_ARGS)
-- Call routine with operands args.
require
valid_operands: valid_operands (args)
callable: callable
apply is
-- Call routine with operands as last set.
require
valid_operands: valid_operands (operands)
callable: callable
defered
end
A.6.31 CLASS PROCEDURE

note
description: "[
Objects representing delayed calls to a procedure,
with some operands possibly still open
]

comment: "[
Features are the same as those of ROUTINE,
with apply made effective, and no further
redefinition of is_equal and copy.
]

class interface
PROCEDURE [BASE_TYPE, OPEN_ARGS -> TUPLE]

feature -- Access
operands: OPEN_ARGS
-- Open operands

target: ANY
-- Target of call

open_operand_type (i: INTEGER): INTEGER
-- Type of i-th open operand.

require
positive : i >= 1
within_bounds: i <= open_count

hash_code: INTEGER
-- Hash code value

feature -- Status report
callable: BOOLEAN
-- Can procedure be called on current object?

is_equal (other: like Current): BOOLEAN
-- Is associated procedure the same as the one
-- associated with other?
-- The object comparison operator ~ relies on this function.

valid_operands (args: OPEN_ARGS): BOOLEAN
-- Are args valid operands for this procedure?

precondition (args: like OPEN_ARGS): BOOLEAN
-- Do args satisfy procedure’s precondition
-- in present state?

postcondition (args: like operands): BOOLEAN
-- Does current state satisfy procedure’s
-- postcondition/or args?

feature -- Measurement
open_count: INTEGER
-- Number of open parameters.

feature -- Element change
set_operands (args: OPEN_ARGS)
-- Use args as operands for next call.

require
valid_operands: valid_operands (args)

feature -- Duplication
copy (other: like Current)
-- Use same procedure as other.

feature -- Basic operations
call (args: OPEN_ARGS)
-- Call procedure with operands args.

require
valid_operands: valid_operands (args)
callable: callable

apply is
-- Call procedure with operands as last set.
require
valid_operands: valid_operands (operands)
callable: callable

end
A.6.32 CLASS FUNCTION

note

description: "[
Objects representing delayed calls to a function,
with some operands possibly still open
]"

comment: "[
Features are the same as those of ROUTINE,
with apply made effective, and the addition of last_result and item.
]"

class interface

FUNCTION [BASE_TYPE, OPEN_ARGS -> TUPLE, RESULT_TYPE]

feature -- Access

last_result: RESULT_TYPE
-- Result of last call, if any.

require
valid_operands: valid_operands (args)
callable: callable

operands: OPEN_ARGS
-- Open operands

target: ANY
-- Target of call

open_operand_type (i: INTEGER): INTEGER
-- Type of i-th open operand.

require
positive : i >= 1
within_bounds: i <= open_count

hash_code: INTEGER
-- Hash code value

precondition (args: like operands): BOOLEAN
-- Do args satisfy function’s precondition
-- in present state?

postcondition (args: like operands): BOOLEAN
-- Does current state satisfy function’s
-- postconditionfor args?

feature -- Status report

callable: BOOLEAN
-- Can function be called on current object?

is_equal (other: like Current): BOOLEAN
-- Is associated function the same as the one
-- associated with other?
-- The object comparison operator ~ relies on this function.

valid_operands (args: OPEN_ARGS): BOOLEAN
-- Are args valid operands for this function?

feature -- Measurement

open_count: INTEGER
-- Number of open parameters.

feature -- Element change

set_operands (args: OPEN_ARGS)
-- Use args as operands for next call.

require
valid_operands: valid_operands (args)

feature -- Duplication

copy (other: like Current)
-- Use same function as other.

feature -- Basic operations

call (args: OPEN_ARGS)
-- Call function with operands args.

require
valid_operands: valid_operands (args)
callable: callable

apply is
-- Call function with operands as last set.

require
valid_operands: valid_operands (operands)
callable: callable

item (args: like operands)
-- Result of calling function with args as operands

require
valid_operands: valid_operands (operands)
callable: callable

ensure
set_by_call: Result = last_result

end
A.6.33 CLASS PREDICATE

**note**

description: "[Objects representing delayed calls to boolean-valued function, with some operands possibly still open ]"

inheritance: "[This class inherits (see section A.5.17) from FUNCTION [BASE_TYPE, OPEN_ARGS, BOOLEAN] ]"

comment: "[Features are the same as those of FUNCTION, with RESULT_TYPE replaced by BOOLEAN, and no further redefinition of is_equal and copy. ]"

**class interface**

\[
\text{PREDICATE [BASE_TYPE, OPEN_ARGS \rightarrow TUPLE]}
\]

**feature -- Access**

*last_result: RESULT_TYPE*
-- Result of last call, if any.

require
valid_operands: valid_operands (args)
callable: callable

*operands: OPEN_ARGS*
-- Open operands

target: ANY
-- Target of call

open_operand_type (i: INTEGER): INTEGER
-- Type of i-th open operand.

require
positive: i >= 1
within_bounds: i <= open_count

*hash_code: INTEGER*
-- Hash code value

precondition (args: like operands): BOOLEAN
-- Do args satisfy function’s precondition
-- in present state?

postcondition (args: like operands): BOOLEAN
-- Does current state satisfy function’s
-- postcondition/or args?

**feature -- Status report**

*callable: BOOLEAN*
-- Can function be called on current object?

*is_equal (other: like Current): BOOLEAN*
-- Is associated function the same as the one
-- associated with other?
-- The object comparison operator ~ relies on this function.

*valid_operands (args: OPEN_ARGS): BOOLEAN*
-- Are args valid operands for this function?

**feature -- Measurement**

*open_count: INTEGER*
-- Number of open parameters.

**feature -- Element change**

*set_operands (args: OPEN_ARGS)*
-- Use args as operands for next call.

require
valid_operands: valid_operands (args)

**feature -- Duplication**

*copy (other: like Current)*
-- Use same function as other.

**feature -- Basic operations**

*call (args: OPEN_ARGS)*
-- Call function with operands args.

require
valid_operands: valid_operands (args)
callable: callable

*apply is*
-- Call function with operands as last set.

require
valid_operands: valid_operands (operands)
callable: callable

*item (args: like operands)*
-- Result of calling function with args as operands

require
valid_operands: valid_operands (operands)
callable: callable

**ensure**

*set_by_call: Result = last_result*