class ARRAY [E_]

+

Summary

top

Class invariant

top

- **valid_bounds:** lower <= upper + 1
- capacity >= upper - lower + 1

-

Overview

top

creation features

- **make (min_index: INTEGER_32, max_index: INTEGER_32)**

  Prepare the array to hold values for indexes in range [min_index .. max_index].

- **with_capacity (needed_capacity: INTEGER_32, low: INTEGER_32)**

  *Create an empty array with **capacity** initialized at least to needed_capacity and **lower** set to low.*

- **from_collection (model: TRAVERSABLE[E_])**

  Initialize the current object with the contents of model.

exported features

- **lower: INTEGER_32**

  Lower index bound.

Creation and Modification:

- **make (min_index: INTEGER_32, max_index: INTEGER_32)**

  Prepare the array to hold values for indexes in range [min_index .. max_index].

- **with_capacity (needed_capacity: INTEGER_32, low: INTEGER_32)**

  *Create an empty array with **capacity** initialized at least to needed_capacity and **lower** set to low.*

Modification:

- **resize (min_index: INTEGER_32, max_index: INTEGER_32)**

  Resize to bounds min_index and max_index.

- **reindex (new_lower: INTEGER_32)**

  Change indexing to take in account the expected new_lower index.

Implementation of deferred:

- **count: INTEGER_32**

  Number of available indices.

- **is_empty: BOOLEAN**

  Is collection empty ?

  See also count.

- **subarray (min: INTEGER_32, max: INTEGER_32): ARRAY [E_]**

  New collection consisting of items at indexes in [min .. max].

- **item (i: INTEGER_32): E_**

  Item at the corresponding index i.

- **put (element: E_, i: INTEGER_32)**

  Make element the item at index i.

- **force (element: E_, index: INTEGER_32)**

  Make element the item at index, enlarging the collection if necessary (new bounds except index are initialized with default values).

- **copy (other: ARRAY [E_])**

  Reinitialize by copying all the items of other.

- **set_all_with (v: E_)**

  Set all items with value v.

- **remove_first**

  *Remove the **first** element of the collection.*

- **remove_head (n: INTEGER_32)**

  Remove the n elements of the collection.

- **remove (index: INTEGER_32)**

  Remove the item at position index.

- **clear_count**

  *Discard all items (**is_empty** is True after that call).*

- **clear_count_and_capacity**

  *Discard all items (**is_empty** is True after that call).*

- **add_first (element: E_)**

  *Add a new item in first position : **count** is increased by one and all other items are shifted right.*

- **add_last (element: E_)**

  *Add a new item at the end : **count** is increased by one.*

- **from_collection (model: TRAVERSABLE[E_])**

  Initialize the current object with the contents of model.

- **all_default: BOOLEAN**

  Do all items have their type's default value?

- **occurrences (element: E_): INTEGER_32**

  *Number of occurrences of element using **is_equal** for comparison.*

- **fast_occurrences (element: E_): INTEGER_32**

  Number of occurrences of element using basic = for comparison.

- **first_index_of (element: E_): INTEGER_32**

*Give the index of the first occurrence of element using **is_equal** for comparison.*

- **index_of (element: E_, start_index: INTEGER_32): INTEGER_32**

  *Using **is_equal** for comparison, gives the index of the first occurrence of element at or after start_index.*

- **reverse_index_of (element: E_, start_index: INTEGER_32): INTEGER_32**

  *Using **is_equal** for comparison, gives the index of the first occurrence of element at or before start_index.*

- **fast_first_index_of (element: E_): INTEGER_32**

  Give the index of the first occurrence of element using basic = for comparison.

- **fast_index_of (element: E_, start_index: INTEGER_32): INTEGER_32**

  Using basic = for comparison, gives the index of the first occurrence of element at or after start_index.

- **fast_reverse_index_of (element: E_, start_index: INTEGER_32): INTEGER_32**

  Using basic = comparison, gives the index of the first occurrence of element at or before start_index.

- **is_equal (other: ARRAY [E_]): BOOLEAN**

  *Do both collections have the same **lower**, **upper**, and items?*

- **is_equal_map (other: ARRAY [E_]): BOOLEAN**

  *Do both collections have the same **lower**, **upper**, and items?*

- **slice (min: INTEGER_32, max: INTEGER_32): ARRAY [E_]**

  New collection consisting of items at indexes in [min..max].

- **get_new_iterator: ITERATOR[E_]**

Accessing:

- **infix "@" (i: INTEGER_32): E_**

  *The infix notation which is actually just a synonym for **item**.*

Writing:

- **swap (i1: INTEGER_32, i2: INTEGER_32)**

  Swap item at index i1 with item at index i2.

- **set_slice_with (v: E_, lower_index: INTEGER_32, upper_index: INTEGER_32)**

  Set all items in range [lower_index .. upper_index] with v.

- **clear_all**

  Set every item to its default value.

Adding:

- **add (element: E_, index: INTEGER_32)**

  *Add a new element at rank index : **count** is increased by one and range [index .. **upper**] is shifted right by one position.*

- **append_collection (other: COLLECTION[E_])**

  Append other to Current.

Removing:

- **remove_last**

  *Remove the **last** item.*

- **remove_tail (n: INTEGER_32)**

  Remove the last n item(s).

Looking and Searching:

- **has (x: E_): BOOLEAN**

  *Look for x using **is_equal** for comparison.*

- **fast_has (x: E_): BOOLEAN**

  Look for x using basic = for comparison.

- **last_index_of (element: E_): INTEGER_32**

  *Using **is_equal** for comparison, gives the index of the last occurrence of element at or before **upper**.*

- **fast_last_index_of (element: E_): INTEGER_32**

  *Using basic = for comparison, gives the index of the last occurrence of element at or before **upper**.*

Looking and comparison:

- **same_items (other: COLLECTION[E_]): BOOLEAN**

  Do both collections have the same items?

Printing:

- **fill_tagged_out_memory**

  *Append a viewable information in **tagged_out_memory** in order to affect the behavior of **out**, **tagged_out**, etc.*

Agents based features:

- **do_all (action: ROUTINE[TUPLE[TUPLE 1[E_]]])**

  Apply action to every item of Current.

- **for_all (test: FUNCTION[TUPLE[TUPLE 1[E_]]]): BOOLEAN**

  Do all items satisfy test?

- **exists (test: FUNCTION[TUPLE[TUPLE 1[E_]]]): BOOLEAN**

  Does at least one item satisfy test?

Other features:

- **replace_all (old_value: E_, new_value: E_)**

  *Replace all occurrences of the element old_value by new_value using **is_equal** for comparison.*

- **fast_replace_all (old_value: E_, new_value: E_)**

  Replace all occurrences of the element old_value by new_value using basic = for comparison.

- **move (lower_index: INTEGER_32, upper_index: INTEGER_32, distance: INTEGER_32)**

  Move range lower_index ..

- **reverse**

Reverse the order of the elements.

Indexing:

- **upper: INTEGER_32**

  Maximum index.

- **valid_index (i: INTEGER_32): BOOLEAN**

  True when i is valid (i.e., inside actual bounds).

Accessing:

- **first: E_**

  The very first item.

- **last: E_**

  The last item.

- **capacity: INTEGER_32**

  Internal storage capacity in number of item.

Interfacing with C:

- **to_external: POINTER**

  Gives C access into the internal storage of the ARRAY.

-

lower: INTEGER_32

writable attribute

top

Lower index bound.

-

make (min_index: INTEGER_32, max_index: INTEGER_32)

effective procedure

top

Prepare the array to hold values for indexes in range [min_index .. max_index].
Set all values to default. When max_index = min_index - 1, the array is_empty.
require

- **valid_bounds:** min_index <= max_index + 1

ensure

- **lower_set:** lower = min_index
- **upper_set:** upper = max_index
- **items_set:** all_default

-

with_capacity (needed_capacity: INTEGER_32, low: INTEGER_32)

effective procedure

top

Create an empty array with capacity initialized at least
to needed_capacity and lower set to low.
require

- needed_capacity >= 0

ensure

- is_empty
- needed_capacity <= capacity
- lower = low

-

resize (min_index: INTEGER_32, max_index: INTEGER_32)

effective procedure

top

Resize to bounds min_index and max_index.
Do not lose any item whose index is in both [lower .. upper] and
[min_index .. max_index]. New positions if any are initialized with the appropriate
default value.
require

- min_index <= max_index + 1

ensure

- lower = min_index
- upper = max_index

-

reindex (new_lower: INTEGER_32)

effective procedure

top

Change indexing to take in account the expected new_lower index.
The upper index is translated accordingly.
ensure

- lower = new_lower
- count = old count

-

count: INTEGER_32

effective function

top

Number of available indices.
See also is_empty, lower, upper.
ensure

- **definition:** Result = upper - lower + 1

-

is_empty: BOOLEAN

effective function

top

Is collection empty ?
See also count.
ensure

- **definition:** Result = count = 0

-

subarray (min: INTEGER_32, max: INTEGER_32): ARRAY [E_]

effective function

top

New collection consisting of items at indexes in [min .. max].
Result has the same dynamic type as Current. See also slice.
require

- lower <= min
- max <= upper
- min <= max + 1

ensure

- Result.lower = min
- same_dynamic_type(Result)
- Result.count = max - min + 1
- Result.lower = min or Result.lower = 0

-

item (i: INTEGER_32): E_

effective function

top

Item at the corresponding index i.
See also lower, upper, valid_index.
require

- valid_index(i)

-

put (element: E_, i: INTEGER_32)

effective procedure

top

Make element the item at index i.
See also lower, upper, valid_index, item, swap, force.
require

- valid_index(i)

ensure

- item(i) = element
- count = old count

-

force (element: E_, index: INTEGER_32)

effective procedure

top

Make element the item at index, enlarging the collection if necessary (new bounds except index are initialized with default values).
See also put, item, swap.
require

- True
- index >= lower

ensure

- lower = index.min(old lower)
- upper = index.max(old upper)
- item(index) = element

-

copy (other: ARRAY [E_])

effective procedure

top

Reinitialize by copying all the items of other.
require

- same_dynamic_type(other)

ensure

- is_equal(other)

-

set_all_with (v: E_)

effective procedure

top

Set all items with value v.
See also set_slice_with.
ensure

- count = old count

-

remove_first

effective procedure

top

Remove the first element of the collection.
See also remove_last, remove, remove_head.
require

- not is_empty

ensure

- upper = old upper
- count = old count - 1
- lower = old lower + 1 xor upper = old upper - 1

-

remove_head (n: INTEGER_32)

effective procedure

top

Remove the n elements of the collection.
See also remove_tail, remove, remove_first.
require

- n > 0 and n <= count

ensure

- upper = old upper
- count = old count - n
- lower = old lower + n xor upper = old upper - n

-

remove (index: INTEGER_32)

effective procedure

top

Remove the item at position index.
Followings items are shifted left by one position.
See also remove_first, remove_head, remove_tail, remove_last.
require

- valid_index(index)

ensure

- count = old count - 1
- upper = old upper - 1

-

clear_count

effective procedure

top

Discard all items (is_empty is True after that call).
If possible, the actual implementation is supposed to keep its internal storage area in order to refill Current in an efficient way.
See also clear_count_and_capacity.
ensure

- capacity = old capacity
- **is_empty:** count = 0

-

clear_count_and_capacity

effective procedure

top

Discard all items (is_empty is True after that call).
If possible, the actual implementation is supposed to release its internal storage area for this memory to be used by other objects.
See also clear_count.
ensure

- capacity = old capacity
- **is_empty:** count = 0

-

add_first (element: E_)

effective procedure

top

Add a new item in first position : count is increased by one and all other items are shifted right.
See also add_last, first, last, add.
ensure

- first = element
- count = 1 + old count
- lower = old lower
- upper = 1 + old upper

-

add_last (element: E_)
effective procedure

top

Add a new item at the end : count is increased by one.
See also add_first, last, first, add.
ensure

- last = element
- count = 1 + old count
- lower = old lower
- upper = 1 + old upper

-

from_collection (model: TRAVERSABLE[E_])
effective procedure

top

Initialize the current object with the contents of model.
require

- model /= Void
- **useful_work:** model /= Current

ensure

- lower = model.lower
- upper = model.upper
- count = model.count

-

all_default: BOOLEAN
effective function

top

Do all items have their type's default value?
Note: for non Void items, the test is performed with the is_default predicate.
See also clear_all.

-

occurrences (element: E_): INTEGER_32
effective function

top

Number of occurrences of element using is_equal for comparison.
See also fast_occurrences, index_of.
ensure

- Result >= 0

-

fast_occurrences (element: E_): INTEGER_32
effective function

top

Number of occurrences of element using basic = for comparison.
See also occurrences, index_of.
ensure

- Result >= 0

-

first_index_of (element: E_): INTEGER_32
effective function

top

Give the index of the first occurrence of element using is_equal for comparison.
Answer upper + 1 when element is not inside.
See also fast_first_index_of, index_of, last_index_of, reverse_index_of.
ensure

- **definition:** Result = index_of(element, lower)

-

index_of (element: E_, start_index: INTEGER_32): INTEGER_32
effective function

top

Using is_equal for comparison, gives the index of the first occurrence of element at or after start_index.
Answer upper + 1 when element when the search fail.
See also fast_index_of, reverse_index_of, first_index_of.
ensure

- Result.in_range(start_index, upper + 1)
- valid_index(Result) implies (create {SAFE_EQUAL}).test(element, item(Result))

-

reverse_index_of (element: E_, start_index: INTEGER_32): INTEGER_32
effective function

top

Using is_equal for comparison, gives the index of the first occurrence of element at or before start_index.
Search is done in reverse direction, which means from the start_index down to the lower index . Answer lower -1 when the search fail.
See also fast_reverse_index_of, last_index_of, index_of.
require

- valid_index(start_index)

ensure

- Result.in_range(lower - 1, start_index)
- valid_index(Result) implies item(Result).is_equal(element)

-

fast_first_index_of (element: E_): INTEGER_32
effective function

top

Give the index of the first occurrence of element using basic = for comparison.
Answer upper + 1 when element is not inside.
See also first_index_of, last_index_of, fast_last_index_of.
ensure

- **definition:** Result = fast_index_of(element, lower)

-

fast_index_of (element: E_, start_index: INTEGER_32): INTEGER_32
effective function

top

Using basic = for comparison, gives the index of the first occurrence of element at or after start_index.
Answer upper + 1 when element when the search fail.
See also index_of, fast_reverse_index_of, fast_first_index_of.
ensure

- Result.in_range(start_index, upper + 1)
- valid_index(Result) implies element = item(Result)

-

fast_reverse_index_of (element: E_, start_index: INTEGER_32): INTEGER_32
effective function

top

Using basic = comparison, gives the index of the first occurrence of element at or before start_index.
Search is done in reverse direction, which means from the start_index down to the lower index . Answer lower -1 when the search fail.
See also reverse_index_of, fast_index_of, fast_last_index_of.
require

- valid_index(start_index)

ensure

- Result.in_range(lower - 1, start_index)
- valid_index(Result) implies item(Result) = element

-

is_equal (other: ARRAY [E_]): BOOLEAN
effective function

top

Do both collections have the same lower, upper, and items?
The basic = is used for comparison of items.
See also is_equal_map, same_items.
require

- other /= Void

ensure

- **commutative:** generating_type = other.generating_type implies Result = other.is_equal(Current)
- Result implies lower = other.lower and upper = other.upper

-

is_equal_map (other: ARRAY [E_]): BOOLEAN

effective function

top

Do both collections have the same lower, upper, and items?
Feature is_equal is used for comparison of items.
See also is_equal, same_items.
ensure

- Result implies lower = other.lower and upper = other.upper

-

slice (min: INTEGER_32, max: INTEGER_32): ARRAY [E_]

effective function

top

New collection consisting of items at indexes in [min..max].
Result has the same dynamic type as Current. The lower index of the Result is the same as lower.
See also from_collection, move, replace_all.
require

- lower <= min
- max <= upper
- min <= max + 1

ensure

- same_dynamic_type(Result)
- Result.count = max - min + 1
- Result.lower = lower

-

get_new_iterator: ITERATOR[E_]

effective function

top

ensure

- Result /= Void

+

infix "@" (i: INTEGER_32): E_

frozen
effective function

top

The infix notation which is actually just a synonym for item.

-

swap (i1: INTEGER_32, i2: INTEGER_32)

effective procedure

top

Swap item at index i1 with item at index i2.
See also item, put.
require

- valid_index(i1)
- valid_index(i2)

ensure

- item(i1) = old item(i2)
- item(i2) = old item(i1)

- count = old count

-

set_slice_with (v: E_, lower_index: INTEGER_32, upper_index: INTEGER_32)

effective procedure

top

Set all items in range [lower_index .. upper_index] with v.
See also set_all_with.
require

- lower_index <= upper_index
- valid_index(lower_index)
- valid_index(upper_index)

ensure

- count = old count

-

clear_all

effective procedure

top

Set every item to its default value.
The count is not affected.
See also clear, all_default.
ensure

- **stable_upper:** upper = old upper
- **stable_lower:** lower = old lower
- all_default

-

add (element: E_, index: INTEGER_32)

deferred procedure

top

Add a new element at rank index : count is increased by one and range [index .. upper] is shifted right by one position.
See also add_first, add_last, append_collection.
require

- index.in_range(lower, upper + 1)

ensure

- item(index) = element
- count = 1 + old count
- upper = 1 + old upper

-

append_collection (other: COLLECTION[E_])

effective procedure

top

Append other to Current.
See also add_last, add_first, add.
require

- other /= Void

ensure

- count = other.count + old count

-

remove_last

deferred procedure

top

Remove the last item.
See also remove_first, remove, remove_tail.
require

- not is_empty

ensure

- count = old count - 1
- upper = old upper - 1

-

remove_tail (n: INTEGER_32)

deferred procedure

top

Remove the last n item(s).
See also remove_head, remove, remove_last.
require

- n > 0 and n <= count

ensure

- count = old count - n
- upper = old upper - n

-

has (x: E_): BOOLEAN

effective function

top

Look for x using is_equal for comparison.
See also fast_has, index_of, fast_index_of.

-

fast_has (x: E_): BOOLEAN

effective function

top

Look for x using basic = for comparison.
See also has, fast_index_of, index_of.

-

last_index_of (element: E_): INTEGER_32

effective function

top

Using is_equal for comparison, gives the index of the last occurrence of element at or before upper.
Search is done in reverse direction, which means from the upper down to
the lower index . Answer lower -1 when the search fail.
See also fast_last_index_of, reverse_index_of, index_of.
ensure

- **definition:** Result = reverse_index_of(element, upper)

-

fast_last_index_of (element: E_): INTEGER_32

effective function

top

Using basic = for comparison, gives the index of the last occurrence of element at or before upper.
Search is done in reverse direction, which means from the upper down to
the lower index . Answer lower -1 when the search fail.
See also fast_reverse_index_of, last_index_of.
ensure

- **definition:** Result = fast_reverse_index_of(element, upper)

-

same_items (other: COLLECTION[E_]): BOOLEAN

effective function

top

Do both collections have the same items?
The basic = is used for comparison of items and indices are not considered (for example this routine may yeld True with Current indexed in range [1..2]
and other indexed in range [2..3]).
See also is_equal_map, is_equal.
require

- other /= Void

ensure

- Result implies count = other.count

-

fill_tagged_out_memory

frozen
effective procedure

top

Append a viewable information in tagged_out_memory in order to affect the behavior of out, tagged_out, etc.

-

do_all (action: ROUTINE[TUPLE[TUPLE 1[E_]]])

effective procedure

top

Apply action to every item of Current.
See also for_all, exists.
require

- action /= Void

-

for_all (test: FUNCTION[TUPLE[TUPLE 1[E_]]]): BOOLEAN

effective function

top

Do all items satisfy test?
See also do_all, exists.
require

- test /= Void

-

exists (test: FUNCTION[TUPLE[TUPLE 1[E_]]]): BOOLEAN

effective function

top

Does at least one item satisfy test?
See also do_all, for_all.
require

- test /= Void

-

replace_all (old_value: E_, new_value: E_)

deferred procedure

top

Replace all occurrences of the element old_value by new_value using is_equal for comparison.
See also fast_replace_all, move.
ensure

- count = old count
- not (create {SAFE_EQUAL}).test(old_value, new_value)
  implies occurrences(old_value) = 0

-

fast_replace_all (old_value: E_, new_value: E_)

deferred procedure

top

Replace all occurrences of the element old_value by new_value using basic = for comparison.
See also replace_all, move.
ensure

- count = old count
- old_value /= new_value implies fast_occurrences(old_value) = 0

-

move (lower_index: INTEGER_32, upper_index: INTEGER_32, distance: INTEGER_32)

effective procedure

top

Move range lower_index ..
upper_index by distance positions. Negative distance moves towards lower indices.
Free places get default values.
See also slice, replace_all.
require

- lower_index <= upper_index
- valid_index(lower_index)
- valid_index(lower_index + distance)
- valid_index(upper_index)

- valid_index(upper_index + distance)

ensure

- count = old count

-

reverse

deferred procedure

top

Reverse the order of the elements.
ensure

- count = old count

-

upper: INTEGER_32

deferred function

top

Maximum index.
See also lower, valid_index, item.

-

valid_index (i: INTEGER_32): BOOLEAN

effective function

top

True when i is valid (i.e., inside actual bounds).
See also lower, upper, item.
ensure

- **definition:** Result = lower <= i and i <= upper

-

first: E_

deferred function

top

The very first item.
See also last, item.
require

- not is_empty

ensure

- **definition:** Result = item(lower)

-

last: E_

deferred function

top

The last item.
See also first, item.
require

- not is_empty

ensure

- **definition:** Result = item(upper)

-

capacity: INTEGER_32

writable attribute

top

Internal storage capacity in number of item.

-

to_external: POINTER

effective function

top

Gives C access into the internal storage of the `ARRAY`.
Result is pointing the element at index lower.
NOTE: do not free/realloc the Result. Resizing of the array

```
      can makes this pointer invalid.
```

require

- not is_empty

ensure

- Result.is_not_null