MATRIX

feature -- { **NONE** }-- Implementation

imp : ARRAY2[ENTRY]

-- row numbers start with 1
-- column numbers start with 1

feature -- commands
make (nor : INTEGER; noc : INTEGER)
-- Create 2d with nor rows and noc columns
require
no_precond: True
ensure
number_of_rows_initialized:
current.number_of_rows ~ nor

number_of_columns_initialized:
current.number_of_columns ~ noc

make_from (lol: LINKED_LIST[LINKED_LIST[ENTRY]]
require
non_empty_list_of_lists:
∀x : 1 ≤ x ≤ lol.count : lol[x].count > 0

rectangle_shape :
∀x : 1 ≤ x ≤ lol.count : lol[x].count ~ lol[1].count
ensure
number_of_rows_initialized:
current.number_of_rows = lol.count

number_of_columns_initialized:
current.number_of_columns = lol[1].count

correct_contents:
∀x∀y: 1 ≤ x ≤ current.number_of_rows :∧ 1 ≤ y ≤ current.number_of_columns :
current.get_entry(x,y).sv ~ current[x][y].sv ∧ current[x][y] ~ lol[x][y].sv

feature -- Queries
scalar_multiply (scalar : INTEGER) : MATRIX
--Obtain a new matrix by applying a scalar multiplication to the current matrix
require
no_precond: True
ensure
current_matrix_unchanged:
∀x ∀y : 1 ≤ x ≤ current.number_of_rows ∧ 1 ≤ y ≤ current.number_of_columns : current[x][y] ~ (old current).deep_twin[x][y]

same_dimension_sizes:
result.number_of_columns ~ current.number_of_columns ∧ result.number_of_rows ~ current.number_of_rows

iv_of_each_entry_scaled:
∀x ∀y : 1 ≤ x ≤ current.number_of_rows ∧ 1 ≤ y ≤ current.number_of_columns : result[x][y].iv ~ current[x][y].iv * scalar

sv_of_each_entry_same:
∀x ∀y : 1 ≤ x ≤ current.number_of_rows ∧ 1 ≤ y ≤ current.number_of_columns : result[x][y].sv ~ current[x][y].sv

transpose : MATRIX
require
True
ensure
current_matrix_unchanged:
∀x ∀y : 1 ≤ x ≤ current.number_of_rows ∧ 1 ≤ y ≤ current.number_of_columns : current[x][y] ~ (old current.deep_twin)[x][y]

corresponding_dimensions:
result.number_of_columns ~ current.number_of_rows ∧ result.number_of_rows ~ current.number_of_columns

corresponding_cells:
∀x ∀y : 1 ≤ x ≤ current.number_of_rows ∧ 1 ≤ y ≤ current.number_of_columns : current[x][y] ~ result[x][y]

feature -- Queries
number_of_rows: INTEGER
require
True
ensure
True

number_of_columns: INTEGER
require
True
ensure
True

number_of_entries: INTEGER
require
True
ensure
correct_result:
result ~ current.number_of_columns * current.number_of_rows

set_entry (e: ENTRY; row, column : INTEGER)
require
valid_row:
row ≥ 1 ∧ row ≤ current.number_of_rows

valid_column:
column ≥ 1 ∧ column ≤ current.number_of_columns
ensure
designated_cell_changed:
current[row][column] /~ (old current.deep_twin)[row][column]

other_cells_unchanged:
∀x ∀y : 1 ≤ x ≤ current.number_of_rows ∧ 1 ≤ y ≤ current.number_of_columns :
(old current)[x][y] ~ current[x][y] : (x = row ∧ y = column)

get_entry (row, column : INTEGER) :ENTRY
require
valid_row:
row ≥ 1 ∧ row ≤ current.number_of_rows

valid_column:
column ≥ 1 ∧ column ≤ current.number_of_columns
ensure
True

get_row(i : INTEGER) : ARRAY[ENTRY]
require
valid_row:
row ≥ 1 ∧ row ≤ current.number_of_rows
ensure
return_value_constraint:
result.lower = 1

correct_result:
∀x : 1 ≤ x ≤ current.number_of_columns : result[x].sv ~ current[i][x].sv ∧ result[x].iv ~ current[i][x]

get_column (i : INTEGER) : LINKED_LIST[ENTRY]
require
valid_column:
column ≥ 1 ∧ column ≤ current.number_of_columns
ensure
correct_result:
∀y : 1 ≤ y ≤ current.number_of_rows : result[y].sv ~ current[y][i] ∧ result[y] ~ current[y][i]

feature -- Equality
is_equal (other: like Current) : Boolean
require
True
ensure
equal_means_same_dimension_sizes:
current.number_of_columns ~ other.number_of_columns ∧ current.number_of_rows ~ other.number_of_rows

equal_means_corresponding_entries_equal:
∀x ∀y : 1 ≤ x ≤ current.number_of_rows ∧ 1 ≤ y ≤ current.number_of_columns : current[x][y].sv ~ other[x][y].sv ∧
current[x][y].iv ~ other[x][y].iv

invariant
implementation_constrant:
imp.lower = 1

atleast_1_by_1:
imp.width * imp.height > 1