

1 DESCRIPTION OF LIBRARY

Library «matrix» gives user functions for operations on matrices.

Some functions of this library actually receive parameters by value. It allows use the same matrix safely as input and as output parameter without memory leaks.

Use [make_matrix](#) function for creating matrix.

Every matrix must be deleted by [delete_matrix](#) function.

2 DATA DESCRIPTION

Data type	Fields	Description
matrix	Structure, which implements abstraction of matrix.	
	double **array	Two-dimensional array of double type. Accessing for elements released as array[i][j].
	int n_rows	Number of rows of matrix. This field is modified only by library internal functions. Changing this parameter by user causes undefined behavior.
	int n_columns	Number of columns of matrix. This field is modified only by library internal functions. Changing this parameter by user causes undefined behavior.
	int n_permutations	Number of permutations witch have done on this matrix. Changing this parameter by user causes undefined behavior.
*init_user	Pointer to function with prototype of double foo (int , int). This function is used in the initialization matrix by init_matrix_by_function . Every element of matrix array[i][j] is filled by foo(int , int) value, where current element's indexes are transmitted as parameters.	

3 DESCRIPTION OF FUNCTIONS

Function's prototype	Description
double determinant (IN const matrix*)	Returns value of determinant of matrix, witch has given as parameter. In error case returns 0.
int add_matrices (IN const matrix* , IN const matrix* , OUT matrix*)	Write sum of two input matrices into OUT matrix. Returns 1 if success, otherwise returns 0.
int add_matrices (IN const matrix* , IN const matrix* , OUT matrix*)	Write difference of first and second input matrices into OUT matrix. Returns 1 if success, otherwise returns 0.
int column_mult_on_const (double factor , int i_column , MODIFIED matrix*)	Multiplex column <i>i_column</i> of matrix MODIFIED by <i>factor</i> . Returns 1 if success, otherwise returns 0.
int columns_sub (double factor , int i_subtracted_column , int i_subtracting_column , MODIFIED matrix*)	Subtract from column <i>i_subtracted_column</i> of matrix MODIFIED column <i>i_subtracting_column</i> , multiplexed by <i>factor</i> . Returns 1 if success, otherwise returns 0.
int columns_swap (int i_col_1 , int i_col_2 , MODIFIED matrix*)	Swap <i>i_col_1</i> and <i>i_col_2</i> columns of matrix MODIFIED. Returns 1 if success, otherwise returns 0.
int copy_column_to_other_matrix (int i_source , int i_receiver , IN const matrix *in_matrix , OUT matrix *out_matrix)	Copy column <i>i_source</i> of matrix <i>in_matrix</i> to column <i>i_receiver</i> of matrix <i>out_matrix</i> . Returns 1 if success, otherwise returns 0.
int copy_matrix (IN const matrix* , OUT matrix*)	Copy matrix IN to matrix OUT. Returns 1 if success, otherwise returns 0.
int copy_row_to_other_matrix (int i_source , int i_receiver , IN const matrix *in_matrix , OUT matrix *out_matrix)	Copy row <i>i_source</i> of matrix <i>in_matrix</i> to row <i>i_receiver</i> of matrix <i>out_matrix</i> . Returns 1 if success, otherwise returns 0.
int delete_matrix (matrix*)	Free dynamically allocated by matrix memory. Returns 1 if success, otherwise returns 0.
int init_matrix_as_unit (MODIFIED matrix*)	Initialize matrix MODIFIED as unit matrix, i. e. 1 on the main diagonal and 0 in other places. Returns 1 if success, otherwise returns 0.
int init_matrix_by_function (MODIFIED matrix* , init_user)	Initialize every element of matrix MODIFIED, by function init_user . Returns 1 if success, otherwise returns 0.
int init_matrix_by_random (MODIFIED matrix* , int down , int up)	Initialize matrix MODIFIED by random values of type of int in range from <i>down</i> to <i>up</i> . Returns 1 if success, otherwise returns 0.

int init_matrix (MODIFIED matrix *)	Initialize matrix, given as parameter by scanned from standard input stream values. Returns 1 if success, otherwise returns 0.
int inverse_matrix (IN const matrix * <i>in_matrix</i> , OUT matrix * <i>out_matrix</i>)	Writes to <i>out_matrix</i> inverted matrix <i>in_matrix</i> . Returns 1 if success, otherwise returns 0.
int multiplex_matrices (IN const matrix * <i>in_matrix_1</i> , IN const matrix * <i>in_matrix_2</i> , OUT matrix *)	Writes to the matrix OUT product of matrices <i>in_matrix_1</i> and <i>in_matrix_2</i> . Returns 1 if success, otherwise returns 0.
int rank (IN const matrix *)	Returns rank of given matrix.
int row_mult_on_const (double <i>factor</i> , int <i>i_row</i> , MODIFIED matrix *)	Multiplexes row <i>i_row</i> of matrix MODIFIED by <i>factor</i> . Returns rank of given matrix.
int rows_sub (double <i>factor</i> , int <i>i_subtracted_row</i> , int <i>i_subtracting_row</i> , MODIFIED matrix *)	Subtract from row <i>i_subtracted_row</i> of matrix MODIFIED row <i>i_subtracting_row</i> , multiplexed by <i>factor</i> . Returns 1 if success, otherwise returns 0.
int rows_swap (int <i>i_row_1</i> , int <i>i_row_2</i> , MODIFIED matrix *)	Swap <i>i_row_1</i> and <i>i_row_2</i> rows of matrix MODIFIED. Returns 1 if success, otherwise returns 0.
int show_matrix (IN const matrix *)	Print IN matrix into standard output stream. Returns 1 if success, otherwise returns 0.
int transpose (IN const matrix *, OUT matrix *)	Writes to the matrix OUT transposed matrix IN. Returns 1 if success, otherwise returns 0.
int triangle_form (IN const matrix *, OUT matrix *)	Writes to the matrix OUT matrix IN in triangle form. Returns 1 if success, otherwise returns 0.
matrix make_matrix (int <i>n_rows</i> , int <i>n_columns</i>)	Creates matrix with <i>n_rows</i> rows and <i>n_columns</i> columns. Returns created matrix.