1 OPIS BIBLIOTEKI

Biblioteka «matrix» ma cel przedstawić użytkownikowi funkcji do posługiwania się na macierzach .

Funkcji tej biblioteki faktycznie przejmują znaczenia wejściowych macierzy przez wartość. To pozwala bezpiecznie używać jako wejściowy i wyjściowy parametr tą samą macierz bez wycieków pamięci.

Do tworzenia macierzy trzeba korzystać z funkcji **make_matrix**.

Każdą macierz po wykorzystaniu trzeba zniszczyć za pomocą funkcji **delete_matrix**.

2 OPIS DANYCH

Typ danych	Pola	Opis typu
matrix	Struktura danych, realizująca abstrakcję macierzy.	
	double **array	Macierz dwuwymiarowa typu double. Dostęp do elementów odbywa się jak do zwyczajnej dwuwymiarowej tablicy: array[i][j].
	int n_rows	Liczba wierszy macierzy. Wartość zostaje nadawana przez funkcji biblioteki. Ręczne nadawanie temu polowi wartości spowoduje nieokreślone zachowanie się programu.
	int n_columns	Liczba kolumn macierzy. Wartość zostaje nadawana przez funkcji biblioteki. Ręczne nadawanie temu polowi wartości spowoduje nieokreślone zachowanie się programu.
	int n_permutations	Zawiera liczbę permutacji, które zostały zrobione nad macierzą.
*init_user	Wskaźnik na funkcję o prototypie double foo (int, int). Ta funkcja wykorzystuje się przy inicjalizacji macierzy przez funkcję **init_matrix_by_function**. Każdemu elementowi macierzy array[i][j] zostaje przepisana wartość foo(int, int), gdzie jako parametry zostają podstawione indeksy bieżącego elementu.	

3 OPIS FUNKCJI

Prototyp funkcji	Opis funkcji
double **determinant** (IN const matrix*)	Zwraca wartość wyznacznika macierzy, podanej jako parametr. W przypadku błędu zwraca 0, ponieważ zerowy wyznacznik oznacza, że macierz nie jest kwadratowa.
int **column_mult_on_const** (double *factor*, int *i_column*, MODIFIED matrix*)	Mnoży kolumnę *i_column* macierzy MODIFIED przez wartość *factor*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **columns_sub** (double *factor*, int *i_subtracted_column*, int *i_subtracting_column*, MODIFIED matrix*)	Odejmuje od kolumny *i_subtracted_column* macierzy MODIFIED kolumnę *i_subtracting_column*, mnożoną przez *factor*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **columns_swap** (int *i_col_1*, int *i_col_2*, MODIFIED matrix*)	Zamienia miejscami kolumny *i_col_1* i *i_col_2* macierzy MODIFIED. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **copy_column_to_other_matrix** (int *i_source*, int *i_receiver*, IN const matrix **in_matrix*, OUT matrix **out_matrix*)	Kopiuje kolumnę *i_source* macierzy *in_matrix* do kolumny *i_ receiver* macierzy *out_matrix*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **copy_matrix** (IN const matrix*, OUT matrix*)	Kopiuje macierz IN do macierzy OUT, zwracając 1 w przypadku sukcesu i 0 w przeciwnym przypadku.
int **copy_row_to_other_matrix** (int *i_source*, int *i_receiver*, IN cons matrix **in_matrix*, OUT matrix **out_matrix*)	Kopiuje wiersz *i_source* macierzy *in_matrix* do wierszu *i_ receiver* macierzy *out_matrix*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **delete_matrix** (matrix*)	Wyzwala pamięć dynamiczną, zajętą macierzą, podaną jako parametr. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **init_matrix_as_unit** (MODIFIED matrix*)	Inicjalizuje macierz MODIFIED jako jedynkową, to znaczy 1 na głównej przekątnej i 0 na pozostałych miejscach. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **init_matrix_by_function** (MODIFIED matrix*, init_user)	Inicjalizuje każdy element macierzy MODIFIED, funkcją **init_user**. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **init_matrix_by_random** (MODIFIED matrix*, int32_t *down*, int32_t *up*)	Inicjalizuje macierz MODIFIED wartościami losowymi typu int w zakresie od *down* do *up*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.

int **init_matrix** (MODIFIED matrix*)	Inicjalizuje macierz, podaną w parametrze wartościami, odczytywanymi ze standardowego wejścia. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **inverse_matrix** (IN const matrix *_in_matrix_, OUT matrix *_out_matrix_)	Zapisuje do macierzy *out_matrix* macierz odwrotną do macierzy *in_matrix*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **multiplex_matrices** (IN const matrix* *in_matrix_1*, IN const matrix* *in_matrix_2*, OUT matrix*)	Zapisuje do macierzy OUT iloczyn macierzy *in_matrix_1* przez macierz *in_matrix_2*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **rank** (IN const matrix*)	Zwraca rang podanej macierzy.
int **row_mult_on_const** (double *factor*, int *i_row*, MODIFIED matrix*)	Mnoży wiersz *i_row* macierzy MODIFIED przez wartość *factor*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **rows_sub** (double *factor*, int *i_subtracted_row*, int *i_subtracting_row*, MODYFIED matrix*)	Odejmuje od wiersza *i_subtracted_row* macierzy MODYFIED wiersz *i_subtracting_row*, mnożony przez *factor*. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **rows_swap** (int *i_row_1*, int *i_row_2*, MODIFIED matrix*)	Zamienia miejscami wierszy *i_row_1* i *i_row_2* macierzy MODIFIED. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **show_matrix** (IN const matrix*)	Wypisuje macierz IN w konsoli. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **transpose** (IN const matrix*, OUT matrix*)	Zapisuje do macierzy OUT transponowaną macierz IN. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
int **triangle_form** (IN const matrix*, OUT matrix*)	Zapisuje do macierzy OUT macierz IN w postaci schodkowej. W przypadku sukcesu zwraca 1, w przeciwnym przypadku 0.
matrix **make_matrix** (int *n_rows*, int *n_columns*)	Funkcja tworzy nową macierz o *n_rows* wierszach i *n_columns* kolumnach.