Matrix Calculator

Generated by Doxygen 1.9.3

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 cmdArgs Struct Reference

```
#include <Structures.h>
```

Public Attributes

- std::string inputAFile
- std::string inputBFile
- std::string outputFile
- std::string OperationType

3.1.1 Detailed Description

A structure that stores the variables extracted from the console

3.1.2 Member Data Documentation

3.1.2.1 inputAFile

```
std::string cmdArgs::inputAFile
```

3.1.2.2 inputBFile

```
std::string cmdArgs::inputBFile
```

3.1.2.3 OperationType

```
std::string cmdArgs::OperationType
```

3.1.2.4 outputFile

```
std::string cmdArgs::outputFile
```

The documentation for this struct was generated from the following files:

- Structures.cpp
- Structures.h

Chapter 4

File Documentation

4.1 LoadingMatrix.cpp File Reference

```
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>
#include <cstddef>
#include <vector>
#include <cstdlib>
#include <random>
#include <chrono>
#include <algorithm>
#include <iomanip>
#include "Manipulators.h"
```

Include dependency graph for LoadingMatrix.cpp:

4.2 LoadingMatrix.h File Reference

```
#include <vector>
```

Include dependency graph for LoadingMatrix.h: This graph shows which files directly or indirectly include this file:

Macros

- #define LOADINGMATRIX

Functions

- int main (int argc, char ∗argv[ ])
- std::vector< std::vector< double > > read_matrix (std::string source)
- void debugMatrix (const std::vector< std::vector< double > > vect)
- void save_solution (std::vector< std::vector< double > > solution, std::string &outputFile)
- void save_solution (double solution, std::string &outputFile)
- void save_solution (std::string input, std::string &outputFile)
- void Create_matrix (std::string &file_name, int nrows, int ncols)
- const double random_value ()

4.2.1 Macro Definition Documentation

4.2.1.1 LOADINGMATRIX

```
#define LOADINGMATRIX
```

4.2.2 Function Documentation

4.2.2.1 Create_matrix()

```
void Create_matrix (
             std::string & file_name,
             int nrows,
             int ncols )
```

print generates and writes a matrix of random values into a file

Parameters

const	std::string& file_name pointer to the file been written into
cconst	int nrows number of rows the matrix been written into file should have
cconst	int ncols number of columns the matrix been written into file should have

4.2.2.2 debugMatrix()

```
void debugMatrix (
             const std::vector< std::vector< double > > vect )
```

The function prints a matrix on console

Parameters

const	std::vector<std::vector<double>> vect vector of vector double printed to the output

4.2.2.3 main()

```
int main (
             int argc,
             char * argv[] )
```

This is the main function of the program int argc The number of entries in the argv array. argv An array of pointers to strings that contain the arguments to the program.

4.2.2.4 random_value()

```
const double random_value ( )
```

The function generates random values between 50 and 1

Returns

double distro(engine) returns a random value of type double

4.2.2.5 read_matrix()

```
std::vector< std::vector< double > > read_matrix (
            std::string source )
```

The function reads a matrix of random number from a file

Parameters

const	std::string& file_name pointer to the file been read from

Returns

std::vector< std::vector<double>> read_matrix returns a vector of vector matrix containing the read matrix

4.2.2.6 save_solution() [1/3]

```
void save_solution (
            double solution,
            std::string & outputFile )
```

The function saves the solution form a matrix operation to a file

Parameters

double	solution double with value to the printed to file
std::string&	outputFile pointer to the file been saved to

4.2.2.7 save_solution() [2/3]

```
void save_solution (
            std::string input,
            std::string & outputFile )
```

The function saves the solution form a matrix operation to a file

Parameters

std::string	file_name string with error message to the printed to file
std::string	&outputFile pointer to the file been saved to

4.2.2.8 save_solution() [3/3]

```
void save_solution (
            std::vector< std::vector< double > > solution,
            std::string & outputFile )
```

The function saves the solution form a matrix operation to a file

Parameters

const	std::string& file_name pointer to the file been saved to
const	std::vector< std::vector<double>> solution vector of vector matrix containing the matrix to be saved to file

4.3 LoadingMatrix.h

Go to the documentation of this file.

```
#ifndef LOADINGMATRIX_H
#define LOADINGMATRIX

#include <vector>

int main(int argc, char* argv[]);

std::vector< std::vector<double» read_matrix(std::string source);

void debugMatrix(const std::vector<std::vector<double» vect);

void save_solution(std::vector<std::vector<double» solution, std::string& outputFile);

void save_solution(double solution, std::string& outputFile);

void save_solution(std::string input, std::string& outputFile);

void Create_matrix(std::string& file_name, int nrows, int ncols);

const double random_value();

//void LOG(std::string message);

#endif
```

4.4 main.cpp File Reference

```
#include <iostream>
#include <vector>
#include "Matrix.h"
#include "LoadingMatrix.h"
#include "Structures.h"
#include "Manipulators.h"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char ∗argv[ ])

4.4.1 Function Documentation

4.4.1.1 main()

```
int main (
            int argc,
            char * argv[ ] )
```

This is the main function of the program int argc The number of entries in the argv array. argv An array of pointers to strings that contain the arguments to the program.

4.5 Manipulators.cpp File Reference

```
#include <sstream>
#include <algorithm>
#include <iomanip>
#include "Manipulators.h"
```

Include dependency graph for Manipulators.cpp:

Functions

- double roundDouble (double &solution, int &precise)
- std::string Removehash (std::string &line, std::string &hash)
- std::string Replace (std::string &line, char &delim)

4.5.1 Function Documentation

4.5.1.1 Removehash()

```
std::string Removehash (
            std::string & line,
            std::string & hash )
```

The function evaluates a line and extraxts line if the variable in the memory location @hash is encountered

Parameters

std::string	&line ampersand to address of line to be evaluated
std::string	&hash variable of character that triggers an extraction of line

Returns

std::string Line returns a string with substring from the line

4.5.1.2 Replace()

```
std::string Replace (
             std::string & line,
             char & delim )
```

The function evaluates a line and replaces delimeters or separators with white space

Parameters

std::string&	line ampersand to address of line to be evaluated

Returns

std::string Line returns a string with the edited line

4.5.1.3 roundDouble()

```
double roundDouble (
             double & solution,
             int & precise )
```

The function rounds up a double to a precise double

Parameters

double	&solution ampersand to address of double been rounded
int	&precise int of numbers the function rounds to

Returns

double returns a more precise double

4.6 Manipulators.h File Reference

This graph shows which files directly or indirectly include this file:

Functions

- std::string Replace (std::string &line, char &delim)
- std::string Removehash (std::string &line, std::string &hash)
- double roundDouble (double &solution, int &precise)

4.6.1 Function Documentation

4.6.1.1 Removehash()

```
std::string Removehash (
              std::string & line,
              std::string & hash )
```

The function evaluates a line and extraxts line if the variable in the memory location @hash is encountered

Parameters

std::string	&line ampersand to address of line to be evaluated
std::string	&hash variable of character that triggers an extraction of line

Returns

std::string Line returns a string with substring from the line

4.6.1.2 Replace()

```
std::string Replace (
              std::string & line,
              char & delim )
```

The function evaluates a line and replaces delimeters or separators with white space

Parameters

std::string&	line ampersand to address of line to be evaluated

Returns

std::string Line returns a string with the edited line

4.6.1.3 roundDouble()

```
double roundDouble (
              double & solution,
              int & precise )
```

The function rounds up a double to a precise double

Parameters

double	&solution ampersand to address of double been rounded
int	&precise int of numbers the function rounds to

Returns

double returns a more precise double

4.7 Manipulators.h

Go to the documentation of this file.

```
#ifndef MANIPULATORS_H
#define  MANIPULATORS_H

std::string Replace(std::string& line, char& delim);

std::string Removehash(std::string& line, std::string& hash);

double roundDouble(double& solution, int& precise);

#endif

```

4.8 Matrix.cpp File Reference

```
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>
#include <vector>
#include "Matrix.h"
```

Include dependency graph for Matrix.cpp:

Functions

- double Determinant (std::vector< std::vector< double > > A)
- std::vector< std::vector< double > > Add (std::vector< std::vector< double > > A, std::vector< std↩ ::vector< double > > B)
- std::vector< std::vector< double > > Subtract (std::vector< std::vector< double > > A, std::vector< std↩ ::vector< double > >B)
- std::vector< std::vector< double > > Multiply (std::vector< std::vector< double > > A, std::vector< std↩ ::vector< double > > B)
- std::vector< std::vector< double > > Transpose (std::vector< std::vector< double > > A)
- std::vector< std::vector< double > > Cofactor (std::vector< std::vector< double > > A)
- std::vector< std::vector< double > > Inverse (std::vector< std::vector< double > > A)

4.8.1 Function Documentation

4.8.1.1 Add()

```
std::vector< std::vector< double > > Add (
            std::vector< std::vector< double > > A,
            std::vector< std::vector< double > > B )
```

The function calculates the addition of two matrix and returns a solution to the file

Parameters

const	std::vector<std::vector<int>>A first matrix from file
const	std::vector<std::vector<int>> B second matrix from file

Returns

std::vector<std::vector<int>> solution solution from the add operation

4.8.1.2 Cofactor()

```
std::vector< std::vector< double > > Cofactor (
            std::vector< std::vector< double > > A )
```

The function takes a matrix and returns the cofactor of the matrix

Parameters

const	std::vector<std::vector<int>> matrix The input matrix to be transposed

Returns

std::vector<std::vector<int>> solution returns the cofactor matrix

4.8.1.3 Determinant()

```
double Determinant (
            std::vector< std::vector< double > > A )
```

The function calculates the determinate of a matrix and returns a solution

Parameters

std::vector<std::vector<double>>	A a vector of vector matrix containing the values read from file

Returns

int Determinant returns the value of the determinant

4.8.1.4 Inverse()

```
std::vector< std::vector< double > > Inverse (
            std::vector< std::vector< double > > A )
```

The function calculates the inverse of a matrix and returns a matrix of vector containing the solution

Parameters

const	std::vector<std::vector<int>> matrix The input matrix
std::string&	outputFile pointer to the file where output solution from the operation is saved

4.8.1.5 Multiply()

```
std::vector< std::vector< double > > Multiply (
            std::vector< std::vector< double > > A,
            std::vector< std::vector< double > > B )
```

The function multiplies two matrix and returns a matrix of vector containing the solution to the file

Parameters

const	std::vector<std::vector<int>> A first Vector of matrix to be multiplied
const	std::vector<std::vector<int>> B second vector of matrix to be multiplied
std::string&	outputFile pointer to the file where output solution from the operation is saved

4.8.1.6 Subtract()

```
std::vector< std::vector< double > > Subtract (
            std::vector< std::vector< double > > A,
            std::vector< std::vector< double > > B )
```

The function calculates the difference of two matrix and returns a solution to the file

Parameters

const	std::vector<std::vector<int>> matrix1 first matrix from file
const	std::vector<std::vector<int>> matrix2 second matrix from file
std::string&	outputFile pointer to the file where output solution from the operation is saved

4.8.1.7 Transpose()

```
std::vector< std::vector< double > > Transpose (
            std::vector< std::vector< double > > A )
```

The function takes a matrix and returns the matrix transpose to the file

Parameters

const	std::vector<std::vector<int>> matrix The input matrix to be transposed

Returns

std::vector<std::vector<int>> solution returns the matrix transpose

4.9 Matrix.h File Reference

```
#include <vector>
```

Include dependency graph for Matrix.h: This graph shows which files directly or indirectly include this file:

Functions

- double Determinant (std::vector< std::vector< double > > A)
- std::vector< std::vector< double > > Add (std::vector< std::vector< double > > A, std::vector< std::vector< double > > B)
- std::vector< std::vector< double > > Subtract (std::vector< std::vector< double > > A, std::vector< std::vector< double > > B)
- std::vector< std::vector< double > > Multiply (std::vector< std::vector< double > > A, std::vector< std::vector< double > > B)
- std::vector< std::vector< double > > Transpose (std::vector< std::vector< double > > A)
- std::vector< std::vector< double > > Cofactor (std::vector< std::vector< double > >A)
- std::vector< std::vector< double > > Inverse (std::vector< std::vector< double > > A)

4.9.1 Function Documentation

4.9.1.1 Add()

```
std::vector< std::vector< double > > Add (
            std::vector< std::vector< double > > A,
            std::vector< std::vector< double > > B )
```

The function calculates the addition of two matrix and returns a solution to the file

Parameters

const	std::vector<std::vector<int>>A first matrix from file
const	std::vector<std::vector<int>> B second matrix from file

Returns

std::vector<std::vector<int>> solution solution from the add operation

4.9.1.2 Cofactor()

```
std::vector< std::vector< double > > Cofactor (
            std::vector< std::vector< double > > A )
```

The function takes a matrix and returns the cofactor of the matrix

Parameters

const	std::vector<std::vector<int>> matrix The input matrix to be transposed

Returns

std::vector<std::vector<int>> solution returns the cofactor matrix

4.9.1.3 Determinant()

```
double Determinant (
            std::vector< std::vector< double > > A )
```

The function calculates the determinate of a matrix and returns a solution

Parameters

std::vector<std::vector<double>>	A a vector of vector matrix containing the values read from file

Returns

int Determinant returns the value of the determinant

4.9.1.4 Inverse()

```
std::vector< std::vector< double > > Inverse (
            std::vector< std::vector< double > > A )
```

The function calculates the inverse of a matrix and returns a matrix of vector containing the solution

Parameters

const	std::vector<std::vector<int>> matrix The input matrix
std::string&	outputFile pointer to the file where output solution from the operation is saved

4.9.1.5 Multiply()

```
std::vector< std::vector< double > > Multiply (
            std::vector< std::vector< double > > A,
            std::vector< std::vector< double > > B )
```

The function multiplies two matrix and returns a matrix of vector containing the solution to the file

Parameters

const	std::vector<std::vector<int>> A first Vector of matrix to be multiplied
const	std::vector<std::vector<int>> B second vector of matrix to be multiplied
std::string&	outputFile pointer to the file where output solution from the operation is saved

4.9.1.6 Subtract()

```
std::vector< std::vector< double > > Subtract (
            std::vector< std::vector< double > > A,
            std::vector< std::vector< double > > B )
```

The function calculates the difference of two matrix and returns a solution to the file

Parameters

const	std::vector<std::vector<int>> matrix1 first matrix from file
const	std::vector<std::vector<int>> matrix2 second matrix from file
std::string&	outputFile pointer to the file where output solution from the operation is saved

4.9.1.7 Transpose()

```
std::vector< std::vector< double > > Transpose (
              std::vector< std::vector< double > > A )
```

The function takes a matrix and returns the matrix transpose to the file

Parameters

const	std::vector<std::vector<int>> matrix The input matrix to be transposed

Returns

std::vector<std::vector<int>> solution returns the matrix transpose

4.10 Matrix.h

Go to the documentation of this file.

```
#ifndef MATRIX_H
#define MATRIX_H

#include <vector>

double Determinant(std::vector<std::vector<double» A);

std::vector<std::vector<double» Add( std::vector<std::vector<double» A, std::vector<std::vector<double»
    B);

std::vector<std::vector<double» Subtract( std::vector<std::vector<double» A,
    std::vector<std::vector<double» B);

std::vector<std::vector<double» Multiply(std::vector<std::vector<double» A,
    std::vector<std::vector<double» B);

std::vector<std::vector<double» Transpose( std::vector<std::vector<double» A);

std::vector<std::vector<double» Cofactor(std::vector<std::vector<double»A);

std::vector<std::vector<double» Inverse(std::vector<std::vector<double» A);

#endif

```

4.11 Sample data .cpp File Reference

4.12 Structures.cpp File Reference

```
#include <iostream>
#include <vector>
#include "Structures.h"
```

Include dependency graph for Structures.cpp:

Classes

- struct cmdArgs

Enumerations

- enum class operation {
  add = 0 , subtract , multiply , determinant ,
  inverse , cofactor , transpose , not_supported ,
  add = 0 , subtract , multiply , determinant ,
  inverse , cofactor , transpose , not_supported }

Functions

- cmdArgs ReadcmdArgs (int argc, char ∗argv[ ], cmdArgs &cmdArgs)
- operation convert (std::string &Operation)

4.12.1 Enumeration Type Documentation

4.12.1.1 operation

```
enum class operation  [strong]
```

Enumerator

add	
subtract	
multiply	
determinant	
inverse	
cofactor	
transpose	
not_supported	
add	
subtract	
multiply	
determinant	
inverse	
cofactor	
transpose	
not_supported	

4.12.2 Function Documentation

4.12.2.1 convert()

```
operation convert (
            std::string & Operation )
```

The function reads the string input(-operation) from user and allocates the correct enum type

Parameters

std,↩ :	string &Operation memory address holding string of the operation to be performed

Returns

operation r returns an enum type with a specific operation to be performed

4.12.2.2 ReadcmdArgs()

```
cmdArgs ReadcmdArgs (
            int argc,
            char * argv[ ],
            cmdArgs & cmdArgs )
```

The function reads the string input(-operation) from user and allocates the correct enum type

Parameters

int	arg memory address holding string of the operation to be performed
*char**	argv[] pointer to an array of strings holding values inputted from the console switches @cmdArgs& cmdArgs ampersand with the memory address to struct for storage of extracted values

Returns

cmdArgs Returns a struct with assigned values

4.13 Structures.h File Reference

`#include <vector>`
Include dependency graph for Structures.h: This graph shows which files directly or indirectly include this file:

Classes

- struct cmdArgs

Enumerations

- enum class operation {
  add = 0 , subtract , multiply , determinant ,
  inverse , cofactor , transpose , not_supported ,
  add = 0 , subtract , multiply , determinant ,
  inverse , cofactor , transpose , not_supported }

Functions

- operation convert (std::string &Operation)
- cmdArgs ReadcmdArgs (int argc, char ∗argv[ ], cmdArgs &cmdArgs)

4.13.1 Enumeration Type Documentation

4.13.1.1 operation

```
enum class operation  [strong]
```

An enum of operations for function execution

Enumerator

add	
subtract	
multiply	
determinant	
inverse	
cofactor	
transpose	
not_supported	
add	
subtract	
multiply	
determinant	
inverse	
cofactor	
transpose	
not_supported	

4.13.2 Function Documentation

4.13.2.1 convert()

```
operation convert (
             std::string & Operation )
```

The function reads the string input(-operation) from user and allocates the correct enum type

Parameters

std,↩ :	string &Operation memory address holding string of the operation to be performed

Returns

operation r returns an enum type with a specific operation to be performed

4.13.2.2 ReadcmdArgs()

```
cmdArgs ReadcmdArgs (
             int argc,
             char * argv[ ],
             cmdArgs & cmdArgs )
```

The function reads the string input(-operation) from user and allocates the correct enum type

Parameters

int	arg memory address holding string of the operation to be performed
char∗	argv[] pointer to an array of strings holding values inputted from the console switches @cmdArgs& cmdArgs ampersand with the memory address to struct for storage of extracted values

Returns

cmdArgs Returns a struct with assigned values

4.14 Structures.h

Go to the documentation of this file.

```
#ifndef STRUCTURES
#define STRUCTURES

#include <vector>

enum class operation { add = 0, subtract, multiply, determinant, inverse, cofactor, transpose,
      not_supported };

struct cmdArgs
{
    std::string inputAFile;
    std::string inputBFile;
    std::string outputFile;
    std::string OperationType;
};

operation convert(std::string& Operation);

cmdArgs ReadcmdArgs(int argc, char* argv[], cmdArgs& cmdArgs);

#endif
```

Index