

## Matrix Calculator

Generated by Doxygen 1.9.3



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 cmdArgs Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 inputAFile	5
3.1.2.2 inputBFile	5
3.1.2.3 OperationType	6
3.1.2.4 outputFile	6
<b>4 File Documentation</b>	<b>7</b>
4.1 LoadingMatrix.cpp File Reference	7
4.2 LoadingMatrix.h File Reference	7
4.2.1 Macro Definition Documentation	8
4.2.1.1 LOADINGMATRIX	8
4.2.2 Function Documentation	8
4.2.2.1 Create_matrix()	8
4.2.2.2 debugMatrix()	8
4.2.2.3 main()	8
4.2.2.4 random_value()	9
4.2.2.5 read_matrix()	9
4.2.2.6 save_solution() [1/3]	9
4.2.2.7 save_solution() [2/3]	10
4.2.2.8 save_solution() [3/3]	10
4.3 LoadingMatrix.h	10
4.4 main.cpp File Reference	11
4.4.1 Function Documentation	11
4.4.1.1 main()	11
4.5 Manipulators.cpp File Reference	11
4.5.1 Function Documentation	11
4.5.1.1 Removehash()	11
4.5.1.2 Replace()	12
4.5.1.3 roundDouble()	12
4.6 Manipulators.h File Reference	13
4.6.1 Function Documentation	13
4.6.1.1 Removehash()	13
4.6.1.2 Replace()	13
4.6.1.3 roundDouble()	14

4.7 Manipulators.h . . . . .	14
4.8 Matrix.cpp File Reference . . . . .	14
4.8.1 Function Documentation . . . . .	15
4.8.1.1 Add() . . . . .	15
4.8.1.2 Cofactor() . . . . .	15
4.8.1.3 Determinant() . . . . .	16
4.8.1.4 Inverse() . . . . .	16
4.8.1.5 Multiply() . . . . .	16
4.8.1.6 Subtract() . . . . .	17
4.8.1.7 Transpose() . . . . .	17
4.9 Matrix.h File Reference . . . . .	17
4.9.1 Function Documentation . . . . .	18
4.9.1.1 Add() . . . . .	18
4.9.1.2 Cofactor() . . . . .	18
4.9.1.3 Determinant() . . . . .	18
4.9.1.4 Inverse() . . . . .	19
4.9.1.5 Multiply() . . . . .	19
4.9.1.6 Subtract() . . . . .	19
4.9.1.7 Transpose() . . . . .	20
4.10 Matrix.h . . . . .	20
4.11 Sample data .cpp File Reference . . . . .	20
4.12 Structures.cpp File Reference . . . . .	20
4.12.1 Enumeration Type Documentation . . . . .	21
4.12.1.1 operation . . . . .	21
4.12.2 Function Documentation . . . . .	21
4.12.2.1 convert() . . . . .	22
4.12.2.2 ReadcmdArgs() . . . . .	22
4.13 Structures.h File Reference . . . . .	22
4.13.1 Enumeration Type Documentation . . . . .	23
4.13.1.1 operation . . . . .	23
4.13.2 Function Documentation . . . . .	23
4.13.2.1 convert() . . . . .	24
4.13.2.2 ReadcmdArgs() . . . . .	24
4.14 Structures.h . . . . .	24
<b>Index</b>	<b>27</b>

# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">cmdArgs</a> . . . . .	5
-----------------------------------	---



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">LoadingMatrix.cpp</a>	7
<a href="#">LoadingMatrix.h</a>	7
<a href="#">main.cpp</a>	11
<a href="#">Manipulators.cpp</a>	11
<a href="#">Manipulators.h</a>	13
<a href="#">Matrix.cpp</a>	14
<a href="#">Matrix.h</a>	17
<a href="#">Sample data .cpp</a>	20
<a href="#">Structures.cpp</a>	20
<a href="#">Structures.h</a>	22





## 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 <cstdlib>
#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

<i>const</i>	std::string& file_name pointer to the file been written into
<i>cconst</i>	int nrows number of rows the matrix been written into file should have
<i>cconst</i>	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

<i>const</i>	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

<i>const</i>	<code>std::string&amp; file_name</code> 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

<i>double</i>	solution double with value to the printed to file
<i>std::string&amp;</i>	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

<i>std::string</i>	file_name string with error message to the printed to file
<i>std::string</i>	&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

<i>const</i>	std::string& file_name pointer to the file been saved to
<i>const</i>	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.](#)

```
1 #ifndef LOADINGMATRIX_H
2 #define LOADINGMATRIX
3
4 #include <vector>
5
10 int main(int argc, char* argv[]);
11
17 std::vector< std::vector<double> > read_matrix(std::string source);
18
23 void debugMatrix(const std::vector<std::vector<double> > vect);
24
30 void save_solution(std::vector<std::vector<double> > solution, std::string& outputFile);
31
37 void save_solution(double solution, std::string& outputFile);
38
44 void save_solution(std::string input, std::string& outputFile);
45
51 void Create_matrix(std::string& file_name, int nrow, int ncol);
52
56 const double random_value();
57
58 //void LOG(std::string message);
59
60 #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**

<i>std::string</i>	&line ampersand to address of line to be evaluated
<i>std::string</i>	&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 delimiters or separators with white space

**Parameters**

<i>std::string&amp;</i>	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**

<i>double</i>	&solution ampersand to address of double been rounded
<i>int</i>	&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 extracts line if the variable in the memory location @hash is encountered

#### Parameters

<i>std::string</i>	&line ampersand to address of line to be evaluated
<i>std::string</i>	&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 delimiters or separators with white space

**Parameters**

<code>std::string&amp;</code>	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**

<code>double</code>	&solution ampersand to address of double been rounded
<code>int</code>	&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.](#)

```
1 #ifndef MANIPULATORS_H
2 #define MANIPULATORS_H
3
9 std::string Replace(std::string& line, char& delim);
10
17 std::string Removehash(std::string& line, std::string& hash);
18
25 double roundDouble(double& solution, int& precise);
26
27 #endif
28
```

**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

<i>const</i>	std::vector<std::vector<int>>A first matrix from file
<i>const</i>	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

<i>const</i>	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**

<code>std::vector&lt;std::vector&lt;double&gt;&gt;</code>	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**

<code>const</code>	<code>std::vector&lt;std::vector&lt;int&gt;&gt;</code> matrix The input matrix
<code>std::string&amp;</code>	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**

<code>const</code>	<code>std::vector&lt;std::vector&lt;int&gt;&gt;</code> A first Vector of matrix to be multiplied
<code>const</code>	<code>std::vector&lt;std::vector&lt;int&gt;&gt;</code> B second vector of matrix to be multiplied
<code>std::string&amp;</code>	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

<i>const</i>	std::vector<std::vector<int>> matrix1 first matrix from file
<i>const</i>	std::vector<std::vector<int>> matrix2 second matrix from file
<i>std::string&amp;</i>	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

<i>const</i>	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

<i>const</i>	std::vector<std::vector<int>>>A first matrix from file
<i>const</i>	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

<i>const</i>	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

<code>std::vector&lt;std::vector&lt;double&gt;&gt;</code>	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

<code>const</code>	<code>std::vector&lt;std::vector&lt;int&gt;&gt;</code> matrix The input matrix
<code>std::string&amp;</code>	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

<code>const</code>	<code>std::vector&lt;std::vector&lt;int&gt;&gt;</code> A first Vector of matrix to be multiplied
<code>const</code>	<code>std::vector&lt;std::vector&lt;int&gt;&gt;</code> B second vector of matrix to be multiplied
<code>std::string&amp;</code>	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**

<i>const</i>	std::vector<std::vector<int>>> matrix1 first matrix from file
<i>const</i>	std::vector<std::vector<int>>> matrix2 second matrix from file
<i>std::string&amp;</i>	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**

<i>const</i>	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.](#)

```
1 #ifndef MATRIX_H
2 #define MATRIX_H
3
4 #include <vector>
5
10 double Determinant(std::vector<std::vector<double>> A);
11
17 std::vector<std::vector<double>> Add( std::vector<std::vector<double>> A, std::vector<std::vector<double>>
    B);
18
24 std::vector<std::vector<double>> Subtract( std::vector<std::vector<double>> A,
    std::vector<std::vector<double>> B);
25
31 std::vector<std::vector<double>> Multiply(std::vector<std::vector<double>> A,
    std::vector<std::vector<double>> B);
32
38 std::vector<std::vector<double>> Transpose( std::vector<std::vector<double>> A);
39
45 std::vector<std::vector<double>> Cofactor(std::vector<std::vector<double>>A);
46
52 std::vector<std::vector<double>> Inverse(std::vector<std::vector<double>> A);
53
54 #endif
55
```

**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

<code>std,</code>	string &Operation memory address holding string of the operation to be performed
<code>:</code>	

##### 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

<code>int</code>	arg memory address holding string of the operation to be performed
<code>char*</code>	argv[] pointer to an array of strings holding values inputted from the console switches @cmdArgs& <a href="#">cmdArgs</a> 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

<code>std,</code>	string &Operation memory address holding string of the operation to be performed
<code>:</code>	

##### 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

<code>int</code>	arg memory address holding string of the operation to be performed
<code>char*</code>	argv[] pointer to an array of strings holding values inputted from the console switches @cmdArgs& <a href="#">cmdArgs</a> 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.](#)

```
1 #ifndef STRUCTURES
2 #define STRUCTURES
3
4 #include <vector>
5
6 enum class operation { add = 0, subtract, multiply, determinant, inverse, cofactor, transpose,
7     not_supported };
8
9
10
11
12
13 struct cmdArgs
14 {
15     std::string inputAFile;
16     std::string inputBFile;
17     std::string outputFile;
18     std::string OperationType;
```

```
19 };
20
25 operation convert(std::string& Operation);
26
33 cmdArgs ReadcmdArgs(int argc, char* argv[], cmdArgs& cmdArgs);
34
35 #endif
```



# Index

- Add
  - Matrix.cpp, [15](#)
  - Matrix.h, [18](#)
- add
  - Structures.cpp, [21](#)
  - Structures.h, [23](#)
- cmdArgs, [5](#)
  - inputAFile, [5](#)
  - inputBFile, [5](#)
  - OperationType, [5](#)
  - outputFile, [6](#)
- Cofactor
  - Matrix.cpp, [15](#)
  - Matrix.h, [18](#)
- cofactor
  - Structures.cpp, [21](#)
  - Structures.h, [23](#)
- convert
  - Structures.cpp, [21](#)
  - Structures.h, [23](#)
- Create\_matrix
  - LoadingMatrix.h, [8](#)
- debugMatrix
  - LoadingMatrix.h, [8](#)
- Determinant
  - Matrix.cpp, [16](#)
  - Matrix.h, [18](#)
- determinant
  - Structures.cpp, [21](#)
  - Structures.h, [23](#)
- inputAFile
  - cmdArgs, [5](#)
- inputBFile
  - cmdArgs, [5](#)
- Inverse
  - Matrix.cpp, [16](#)
  - Matrix.h, [19](#)
- inverse
  - Structures.cpp, [21](#)
  - Structures.h, [23](#)
- LOADINGMATRIX
  - LoadingMatrix.h, [8](#)
- LoadingMatrix.cpp, [7](#)
- LoadingMatrix.h, [7](#)
  - Create\_matrix, [8](#)
  - debugMatrix, [8](#)
  - LOADINGMATRIX, [8](#)
  - main, [8](#)
  - random\_value, [9](#)
  - read\_matrix, [9](#)
  - save\_solution, [9](#), [10](#)
- main
  - LoadingMatrix.h, [8](#)
  - main.cpp, [11](#)
- main.cpp, [11](#)
  - main, [11](#)
- Manipulators.cpp, [11](#)
  - Removehash, [11](#)
  - Replace, [12](#)
  - roundDouble, [12](#)
- Manipulators.h, [13](#)
  - Removehash, [13](#)
  - Replace, [13](#)
  - roundDouble, [14](#)
- Matrix.cpp, [14](#)
  - Add, [15](#)
  - Cofactor, [15](#)
  - Determinant, [16](#)
  - Inverse, [16](#)
  - Multiply, [16](#)
  - Subtract, [17](#)
  - Transpose, [17](#)
- Matrix.h, [17](#)
  - Add, [18](#)
  - Cofactor, [18](#)
  - Determinant, [18](#)
  - Inverse, [19](#)
  - Multiply, [19](#)
  - Subtract, [19](#)
  - Transpose, [20](#)
- Multiply
  - Matrix.cpp, [16](#)
  - Matrix.h, [19](#)
- multiply
  - Structures.cpp, [21](#)
  - Structures.h, [23](#)
- not\_supported
  - Structures.cpp, [21](#)
  - Structures.h, [23](#)
- operation
  - Structures.cpp, [21](#)
  - Structures.h, [23](#)
- OperationType

- cmdArgs, 5
- outputFile
  - cmdArgs, 6
- random\_value
  - LoadingMatrix.h, 9
- read\_matrix
  - LoadingMatrix.h, 9
- ReadcmdArgs
  - Structures.cpp, 22
  - Structures.h, 24
- Removehash
  - Manipulators.cpp, 11
  - Manipulators.h, 13
- Replace
  - Manipulators.cpp, 12
  - Manipulators.h, 13
- roundDouble
  - Manipulators.cpp, 12
  - Manipulators.h, 14
- Sample data .cpp, 20
- save\_solution
  - LoadingMatrix.h, 9, 10
- Structures.cpp, 20
  - add, 21
  - cofactor, 21
  - convert, 21
  - determinant, 21
  - inverse, 21
  - multiply, 21
  - not\_supported, 21
  - operation, 21
  - ReadcmdArgs, 22
  - subtract, 21
  - transpose, 21
- Structures.h, 22
  - add, 23
  - cofactor, 23
  - convert, 23
  - determinant, 23
  - inverse, 23
  - multiply, 23
  - not\_supported, 23
  - operation, 23
  - ReadcmdArgs, 24
  - subtract, 23
  - transpose, 23
- Subtract
  - Matrix.cpp, 17
  - Matrix.h, 19
- subtract
  - Structures.cpp, 21
  - Structures.h, 23
- Transpose
  - Matrix.cpp, 17
  - Matrix.h, 20
- transpose
  - Structures.cpp, 21
  - Structures.h, 23