Project 1.1

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 Matrix Class Reference

`#include <Matrix.hpp>`

Public Member Functions

- Matrix ()
- Matrix (const std::vector< int > &A, unsigned int n)
- Matrix (const std::vector< int > &A, unsigned int m, unsigned int n)
- int get (unsigned int i) const
- int get (unsigned int i, unsigned int j) const
- bool set (unsigned int i, int ai)
- bool set (unsigned int i, unsigned int j, int aij)
- unsigned int size (unsigned int dim) const
- bool equal (const Matrix &rhs) const
- const Matrix add (const Matrix &rhs) const
- const Matrix sub (const Matrix &rhs) const
- const Matrix mult (const Matrix &rhs) const
- const Matrix mult (int c) const
- const Matrix pow (unsigned int n) const
- const Matrix trans (const Matrix &rhs) const
- void output (std::ostream &out) const

2.1.1 Detailed Description

This is a basic C++ class to represent two-dimensional matrices. It's not meant to be difficult but as a refresher on classes.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Matrix() [1/3]

```
Matrix::Matrix ( )
```

Default constructor. It should create a 2-by-2 matrix will all elements set to zero.

2.1.2.2 Matrix() [2/3]

```
Matrix::Matrix (
            const std::vector< int > & A,
            unsigned int n )
```

Parameterized constructor. Use the parameters to set the matrix element; if parameters are inconsistent then create a 0-by-0 matrix.

Parameters

A	- values for matrix elements, specified column-wise.
n	- number of columns for the new matrix.

2.1.2.3 Matrix() [3/3]

```
Matrix::Matrix (
            const std::vector< int > & A,
            unsigned int m,
            unsigned int n )
```

Another parameterized constructor. Use the parameters to set the matrix element; if parameters are inconsistent then create a 0-by-0 matrix.

Parameters

A	- values for matrix elements, specified column-wise.
m	- number of rows for the new matrix.
n	- number of columns for the new matrix.

2.1.3 Member Function Documentation

2.1.3.1 add()

```
const Matrix Matrix::add (
            const Matrix & rhs ) const
```

Creates and returns a new Matrix object representing the matrix addition of two Matrix objects.

Returns

a new Matrix object that contains the appropriate summed elements, a 0-by-0 matrix if matrices can't be added.

Parameters

rhs	- the Matrix object to add to this object.

2.1.3.2 equal()

```
bool Matrix::equal (
            const Matrix & rhs ) const
```

Returns true if the elements for this object and rhs are the same, false otherwise.

Parameters

rhs	- the Matrix object to compare to this object.

Returns

true if elements in both objects are the same, false otherwise.

2.1.3.3 get() [1/2]

```
int Matrix::get (
            unsigned int i ) const
```

Returns the element at specified linear index.

Parameters

i	- column-wise (linear) index of object.

Returns

element at specified linear index or smallest possible value for int if index is invalid.

2.1.3.4 get() [2/2]

```
int Matrix::get (
            unsigned int i,
            unsigned int j ) const
```

Returns the element at specified row, column index.

Parameters

i	- row index of object.
j	- column index of object.

Returns

element at specified row, column index or smallest possible value for int if index is invalid.

2.1.3.5 mult() [1/2]

```
const Matrix Matrix::mult (
            const Matrix & rhs ) const
```

Creates and returns a new Matrix object that is the multiplication of this and the given Matrix object.

Returns

a new Matrix object that contains the multiplication of this and the given Matrix object, a 0-by-0 matrix if matrices can't be multiplied.

Parameters

rhs	- the Matrix object to multiply with this object.

2.1.3.6 mult() [2/2]

```
const Matrix Matrix::mult (
            int c ) const
```

Creates and returns a new Matrix object that is the multiplication of this and the given scalar.

Returns

a new Matrix object that contains the multiplication of this and the given scalar.

Parameters

rhs	- the scalar value to multiply with this object.

2.1.3.7 output()

```
void Matrix::output (
            std::ostream & out ) const
```

Outputs this Matrix object on the given ostream (for debugging).

Parameters

out	- the ostream object to use to output.

2.1.3.8 pow()

```
const Matrix Matrix::pow (
            unsigned int n ) const
```

Creates and returns a new Matrix object that is the power of this.

Returns

a new Matrix object that raises this and to the given power.

Parameters

n	- the power to which this object should be raised.

2.1.3.9 set() [1/2]

```
bool Matrix::set (
            unsigned int i,
            int ai )
```

Sets the element at specified linear index i to given value; if index is invalid matrix should not be modified.

Parameters

i	- column-wise (linear) index of object to set.
ai	- value for element at index i

Returns

true if set is successful, false otherwise.

2.1.3.10 set() [2/2]

```
bool Matrix::set (
            unsigned int i,
            unsigned int j,
            int aij )
```

Sets the element at specified row, column index to given value; if either index is invalid matrix should not be modified.

Parameters

i	- row index of object to set.
j	- column index of object to set.
aij	- value for element at index i, j

Returns

true if set is successful, false otherwise.

2.1.3.11 size()

```
unsigned int Matrix::size (
            unsigned int dim ) const
```

Returns the size of the matrix along a given dimension (i.e., numbers of elements in a row or column)

Parameters

dim	- 1 for row, 2 for column

Returns

the number of elements according to dimension specified, if dimension is not valid return 0

2.1.3.12 sub()

```
const Matrix Matrix::sub (
            const Matrix & rhs ) const
```

Creates and returns a new Matrix object representing the matrix subtraction of two Matrix objects.

Returns

a new Matrix object that contains the appropriate difference elements, a 0-by-0 matrix if matrices can't be subtracted.

Parameters

rhs	- the Matrix object to subtract from this object.

2.1.3.13 trans()

```
const Matrix Matrix::trans (
            const Matrix & rhs ) const
```

Creates and returns a new Matrix object that is the transpose of this.

Returns

a new Matrix object that is the transpose of this object.

The documentation for this class was generated from the following file:

- Matrix.hpp