My Project

Generated by Doxygen 1.12.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 matrix Namespace Reference

Classes

- class Matrix

  A class for performing operations on 2D matrices such as addition, subtraction, and multiplication.
- class vector

  A class for performing vector operations such as addition, subtraction, scaling, dot product, magnitude, and cosine similarity.

Chapter 5

Class Documentation

5.1 matrix::Matrix Class Reference

A class for performing operations on 2D matrices such as addition, subtraction, and multiplication.

```
#include <vector.h>
```

Static Public Member Functions

- static std::vector< std::vector< int > > add2D (const std::vector< std::vector< int > > &firstMatrix, const std::vector< std::vector< int > > &secondMatrix)

  Adds two 2D matrices element-wise.
- static std::vector< std::vector< int > > substract2D (const std::vector< std::vector< int > > &firstMatrix, const std::vector< std::vector< int > > &secondMatrix)

  Subtracts the second 2D matrix from the first 2D matrix element-wise.
- static std::vector< std::vector< int > > multiply2D (const std::vector< std::vector< int > > &firstMatrix, const std::vector< std::vector< int > > &secondMatrix)

  Multiplies two 2D matrices.

5.1.1 Detailed Description

A class for performing operations on 2D matrices such as addition, subtraction, and multiplication.

Definition at line 138 of file vector.h.

5.1.2 Member Function Documentation

5.1.2.1 add2D()

```
static std::vector< std::vector< int > > matrix::Matrix::add2D (
           const std::vector< std::vector< int > > & firstMatrix,
           const std::vector< std::vector< int > > & secondMatrix)  [inline], [static]
```

Adds two 2D matrices element-wise.

Parameters

firstMatrix	The first matrix.
secondMatrix	The second matrix.

Returns

The resultant matrix after addition.

Exceptions

std::invalid_argument	if the matrices have different dimensions.

Definition at line 147 of file vector.h.

5.1.2.2 multiply2D()

```
static std::vector< std::vector< int > > matrix::Matrix::multiply2D (
             const std::vector< std::vector< int > > & firstMatrix,
             const std::vector< std::vector< int > > & secondMatrix)  [inline], [static]
```

Multiplies two 2D matrices.

Parameters

firstMatrix	The first matrix.
secondMatrix	The second matrix.

Returns

The resultant matrix after multiplication.

Exceptions

std::invalid_argument	if the number of columns in the first matrix is not equal to the number of rows in the second matrix.

Definition at line 205 of file vector.h.

5.1.2.3 substract2D()

```
static std::vector< std::vector< int > > matrix::Matrix::substract2D (
             const std::vector< std::vector< int > > & firstMatrix,
             const std::vector< std::vector< int > > & secondMatrix)  [inline], [static]
```

Subtracts the second 2D matrix from the first 2D matrix element-wise.

Parameters

firstMatrix	The first matrix.
secondMatrix	The second matrix.

Returns

The resultant matrix after subtraction.

Exceptions

std::invalid_argument	if the matrices have different dimensions.

Definition at line 176 of file vector.h.

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

- vector.h

5.2 matrix::vector Class Reference

A class for performing vector operations such as addition, subtraction, scaling, dot product, magnitude, and cosine similarity.

```
#include <vector.h>
```

Static Public Member Functions

- static std::vector< int > add (std::vector< int > firstVector, std::vector< int > secondVector)

  Adds two vectors element-wise.
- static std::vector< int > substract (std::vector< int > firstVector, std::vector< int > secondVector)

  Subtracts the second vector from the first vector element-wise.
- static std::vector< int > scale (std::vector< int > vector, int scale)

  Scales a vector by a given factor.
- static int dotProduct (const std::vector< int > &firstVector, const std::vector< int > &secondVector=std↩ ::vector< int >())

  Computes the dot product of two vectors.
- static double magnitude (const std::vector< int > &firstVector, const std::vector< int > &secondVector=std↩ ::vector< int >())

  Computes the magnitude of the difference between two vectors.
- static double cosineSimilarity (const std::vector< int > &firstVector, const std::vector< int > &secondVector)

  Computes the cosine similarity between two vectors.

5.2.1 Detailed Description

A class for performing vector operations such as addition, subtraction, scaling, dot product, magnitude, and cosine similarity.

Definition at line 15 of file vector.h.

5.2.2 Member Function Documentation

5.2.2.1 add()

```
static std::vector< int > matrix::vector::add (
            std::vector< int > firstVector,
            std::vector< int > secondVector)  [inline], [static]
```

Adds two vectors element-wise.

Parameters

firstVector	The first vector.
secondVector	The second vector.

Returns

The resultant vector after addition.

Definition at line 23 of file vector.h.

5.2.2.2 cosineSimilarity()

```
static double matrix::vector::cosineSimilarity (
            const std::vector< int > & firstVector,
            const std::vector< int > & secondVector)  [inline], [static]
```

Computes the cosine similarity between two vectors.

Parameters

firstVector	The first vector.
secondVector	The second vector.

Returns

The cosine similarity between the two vectors.

Exceptions

std::invalid_argument	if the vectors are of different lengths.

Definition at line 122 of file vector.h.

5.2.2.3 dotProduct()

```
static int matrix::vector::dotProduct (
            const std::vector< int > & firstVector,
            const std::vector< int > & secondVector = std::vector<int>())  [inline], [static]
```

Computes the dot product of two vectors.

Parameters

firstVector	The first vector.
secondVector	The second vector. Defaults to an empty vector (treated as a zero vector).

Returns

The dot product of the two vectors.

Exceptions

std::invalid_argument	if the vectors are of different lengths.

Definition at line 73 of file vector.h.

5.2.2.4 magnitude()

```
static double matrix::vector::magnitude (
             const std::vector< int > & firstVector,
             const std::vector< int > & secondVector = std::vector<int>())  [inline], [static]
```

Computes the magnitude of the difference between two vectors.

Parameters

firstVector	The first vector.
secondVector	The second vector. Defaults to an empty vector (treated as a zero vector).

Returns

The magnitude of the difference between the two vectors.

Exceptions

std::invalid_argument	if the vectors are of different lengths.

Definition at line 99 of file vector.h.

5.2.2.5 scale()

```
static std::vector< int > matrix::vector::scale (
             std::vector< int > vector,
             int scale)  [inline], [static]
```

Scales a vector by a given factor.

Parameters

vector	The vector to scale.
scale	The scaling factor.

Returns

The resultant scaled vector.

Definition at line 59 of file vector.h.

5.2.2.6 substract()

```
static std::vector< int > matrix::vector::substract (
            std::vector< int > firstVector,
            std::vector< int > secondVector)  [inline], [static]
```

Subtracts the second vector from the first vector element-wise.

Parameters

firstVector	The first vector.
secondVector	The second vector.

Returns

The resultant vector after subtraction.

Definition at line 41 of file vector.h.

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

- vector.h

Chapter 6

File Documentation

6.1 vector.cpp File Reference

```
#include "vector.h"
```

Functions

- int main ()

  The main function demonstrating various vector and matrix operations.

6.1.1 Function Documentation

6.1.1.1 main()

```
int main ()
```

The main function demonstrating various vector and matrix operations.

This function performs the following operations:

- Addition and subtraction of two vectors.
- Addition, subtraction, and multiplication of 2D matrices.
- Calculation of the dot product of a vector.
- Calculation of the cosine similarity between two vectors.

Returns

int Returns 0 on successful execution.

Definition at line 44 of file vector.cpp.

6.2 vector.cpp

Go to the documentation of this file.

```
#include "vector.h"

int main() {
    // Vectors for addition and subtraction
    std::vector<int> firstVector = {1, 3, 6, 9};
    std::vector<int> secondVector = {1, 3, -6, 9};

    // Perform vector addition
    std::vector<int> resultant = matrix::vector::add(firstVector, secondVector);

    // Perform vector subtraction
    std::vector<int> resultantt = matrix::vector::substract(firstVector, secondVector);

    // 2D matrices for addition and subtraction
    std::vector<std::vector<int» firstMatrix = {{1, 3, 6, 9},
                                                 {5, 9, 8, 0},
                                                 {1, 9, 7, 5}};
    std::vector<std::vector<int» secondMatrix = {{1, 3, 6, 9},
                                                  {5, 9, 8, 0},
                                                  {1, 9, 7, 5}};

    // Perform 2D matrix addition
    matrix::Matrix::add2D(firstMatrix, secondMatrix);

    // Perform 2D matrix subtraction
    matrix::Matrix::substract2D(firstMatrix, secondMatrix);

    // 2D matrices for multiplication
    std::vector<std::vector<int» firstMatrixForMultiplication = {{1, 3, 6, 9},
                                                                  {5, 9, 8, 0},
                                                                  {1, 9, 7, 5}};
    std::vector<std::vector<int» secondMatrixForMultiplication = {{1, 3,  6, 9},
                                                                   {5, 9,  8, 0},
                                                                   {1, 9,  7, 5},
                                                                   {1, 18, 9, 8}};

    // Perform 2D matrix multiplication
    matrix::Matrix::multiply2D(firstMatrixForMultiplication, secondMatrixForMultiplication);

    // Vector for dot product calculation
    std::vector<int> vec1 = {1, 2, 3};

    // Calculate dot product of a vector with a zero vector (default)
    int result2 = matrix::vector::dotProduct(vec1);

    // Calculate cosine similarity between two vectors
    std::cout « matrix::vector::cosineSimilarity(firstVector, secondVector) * 100 « " %" « std::endl;

    return 0;
}
```

6.3 vector.h File Reference

```
#include <utility>
#include <vector>
#include <iostream>
#include <cmath>
```

Classes

- class matrix::vector

    A class for performing vector operations such as addition, subtraction, scaling, dot product, magnitude, and cosine similarity.

- class matrix::Matrix

    A class for performing operations on 2D matrices such as addition, subtraction, and multiplication.

Namespaces

- namespace matrix

6.4 vector.h

Go to the documentation of this file.

```
#ifndef VECTOR_VECTOR_H
#define VECTOR_VECTOR_H

#include <utility>
#include <vector>
#include <iostream>
#include <cmath>

namespace matrix {

    class vector {
    public:
        static std::vector<int> add(std::vector<int> firstVector, std::vector<int> secondVector) {
            std::vector<int> resultant(firstVector.size());
            for (int i = 0; i < firstVector.size(); i++) {
                resultant[i] = firstVector[i] + secondVector[i];
            }
            for (int i = 0; i < firstVector.size(); i++) {
                std::cout « resultant[i] « " ";
            }
            std::cout « std::endl;
            return resultant;
        }

        static std::vector<int> substract(std::vector<int> firstVector, std::vector<int> secondVector) {
            std::vector<int> resultant(firstVector.size());
            for (int i = 0; i < firstVector.size(); i++) {
                resultant[i] = firstVector[i] - secondVector[i];
            }
            for (int i = 0; i < firstVector.size(); i++) {
                std::cout « resultant[i] « " ";
            }
            std::cout « std::endl;
            return resultant;
        }

        static std::vector<int> scale(std::vector<int> vector, int scale) {
            for (int i = 0; i < vector.size(); i++) {
                vector[i] *= scale;
            }
            return vector;
        }

        static int dotProduct(const std::vector<int> &firstVector, const std::vector<int> &secondVector = std::vector<int>()) {
            std::vector<int> adjustedSecondVector = secondVector;

            if (adjustedSecondVector.empty()) {
                adjustedSecondVector.resize(firstVector.size(), 0);
            }

            if (firstVector.size() != adjustedSecondVector.size()) {
                throw std::invalid_argument("Vectors must be of same length");
            }

            int result = 0;
            for (size_t i = 0; i < firstVector.size(); ++i) {
                result += firstVector[i] * adjustedSecondVector[i];
            }

            return result;
        }

        static double magnitude(const std::vector<int> &firstVector, const std::vector<int> &secondVector = std::vector<int>()) {
            std::vector<int> adjustedSecondVector = secondVector;

            if (adjustedSecondVector.empty()) {
                adjustedSecondVector.resize(firstVector.size(), 0);
            }
            if (firstVector.size() != adjustedSecondVector.size()) {
                throw std::invalid_argument("Vectors must be of same length");
            }
            int result = 0;
            for (int i = 0; i < firstVector.size(); i++) {
                result += (adjustedSecondVector[i] - firstVector[i]) * (adjustedSecondVector[i] - firstVector[i]);
            }
            return sqrt((double) result);
        }

        static double cosineSimilarity(const std::vector<int> &firstVector, const std::vector<int> &secondVector) {
            if (firstVector.size() != secondVector.size()) {
                throw std::invalid_argument("Vectors must be of same length");
            }
            int dotProduct = matrix::vector::dotProduct(firstVector, secondVector);
            double magnitudeOfA = matrix::vector::magnitude(firstVector);
            double magnitudeOfB = matrix::vector::magnitude(secondVector);
            double cosineSimilarity = dotProduct / (magnitudeOfA * magnitudeOfB);
            return cosineSimilarity;
        }
    };

    class Matrix {
    public:
        static std::vector<std::vector<int» add2D(const std::vector<std::vector<int» &firstMatrix, const std::vector<std::vector<int» &secondMatrix) {
            if (firstMatrix.size() != secondMatrix.size() || firstMatrix[0].size() != secondMatrix[0].size()) {
                throw std::invalid_argument("Matrices must have the same dimensions");
            }

            std::vector<std::vector<int» resultant(firstMatrix.size(), std::vector<int>(firstMatrix[0].size()));

            for (size_t i = 0; i < firstMatrix.size(); ++i) {
                for (size_t j = 0; j < firstMatrix[i].size(); ++j) {
                    resultant[i][j] = firstMatrix[i][j] + secondMatrix[i][j];
                }
            }
            for (const auto &row: resultant) {
                for (const int element: row) {
                    std::cout « element « "   ";
                }
                std::cout « std::endl;
            }

            return resultant;
        }

        static std::vector<std::vector<int» substract2D(const std::vector<std::vector<int» &firstMatrix, const std::vector<std::vector<int» &secondMatrix) {
            if (firstMatrix.size() != secondMatrix.size() || firstMatrix[0].size() != secondMatrix[0].size()) {
                throw std::invalid_argument("Matrices must have the same dimensions");
            }

            std::vector<std::vector<int» resultant(firstMatrix.size(), std::vector<int>(firstMatrix[0].size()));

            for (size_t i = 0; i < firstMatrix.size(); i++) {
                for (size_t j = 0; j < firstMatrix[i].size(); j++) {
                    resultant[i][j] = firstMatrix[i][j] - secondMatrix[i][j];
                }
            }
            for (const auto &row: resultant) {
                for (const int element: row) {
                    std::cout « element « "   ";
                }
                std::cout « std::endl;
            }

            return resultant;
        }

        static std::vector<std::vector<int» multiply2D(const std::vector<std::vector<int» &firstMatrix, const std::vector<std::vector<int» &secondMatrix) {
            if (firstMatrix[0].size() != secondMatrix.size()) {
                throw std::invalid_argument("The number of columns in the first matrix must be equal to the number of rows in the second matrix");
            }
            std::vector<std::vector<int» resultant(firstMatrix.size(), std::vector<int>(secondMatrix[0].size()));
            for (size_t i = 0; i < firstMatrix.size(); i++) {
                for (size_t j = 0; j < firstMatrix[i].size(); j++) {
                    resultant[i][j] = 0;
                    for (size_t k = 0; k < secondMatrix.size(); k++) {
                        resultant[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
                    }
                }
            }

            for (const auto &row: resultant) {
                for (const int element: row) {
                    std::cout « element « "   ";
                }
                std::cout « std::endl;
            }

            return resultant;
        }
    };
}

#endif //VECTOR_VECTOR_H
```

Index