Quadratic

Generated by Doxygen 1.9.3

Chapter 1

quadratic

First week assignment from the system programming course by Huawei and MIPT.

Command line program that solves a quadratic equation with given coefficients.

Chapter 2

Class Index

2.1 Class List

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

testCase
Encapsulates the set of values needed for testing . . . . **??**

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/quadratic.h . . . . ??
include/test.h . . . . ??
src/main.cpp . . . . ??
src/quadratic.cpp . . . . ??
src/test.cpp . . . . ??

Chapter 4

Class Documentation

4.1 testCase Struct Reference

Encapsulates the set of values needed for testing.

```
#include <test.h>
```

Public Attributes

- size_t id
- double a
- double b
- double c
- size_t nRoots
- double x1
- double x2

4.1.1 Detailed Description

Encapsulates the set of values needed for testing.

4.1.2 Member Data Documentation

4.1.2.1 a

```
double testCase::a
```

a-coefficient

4.1.2.2 b

```
double testCase::b
```

b-coefficient

4.1.2.3 c

```
double testCase::c
```

c-coefficient

4.1.2.4 id

```
size_t testCase::id
```

Test id

4.1.2.5 nRoots

```
size_t testCase::nRoots
```

Expected number of roots

4.1.2.6 x1

```
double testCase::x1
```

Expected first root

4.1.2.7 x2

```
double testCase::x2
```

Expected second root

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

- include/test.h

Chapter 5

File Documentation

5.1 include/quadratic.h File Reference

Enumerations

- enum NRoots { ZERO , ONE , TWO , INF_ROOTS }

Functions

- NRoots solveQuadratic (double a, double b, double c, double ∗x1, double ∗x2)
- double solveLinear (double a, double b)
- bool isEqualDouble (double lhs, double rhs)
- int getCoefsFromInput (double ∗a, double ∗b, double ∗c)
- void printResult (NRoots nRoots, double x1, double x2)
- void printEquation (double a, double b, double c)

Variables

- const double EPSILON = 1e-2

  Determines the precision of comparison in isEqualDouble function.

5.1.1 Enumeration Type Documentation

5.1.1.1 NRoots

```
enum NRoots
```

Enum type with different possible return values of the solveQuadratic function

Enumerator

ZERO	
ONE	
TWO	
INF_ROOTS	

5.1.2 Function Documentation

5.1.2.1 getCoefsFromInput()

```
int getCoefsFromInput (
            double * a,
            double * b,
            double * c )
```

Gets equation coefficients from the standard input

Parameters

out	*a*	Pointer to a-coefficient
out	*b*	Pointer to b-coefficient
out	*c*	Pointer to c-coefficient

Returns

1 if operation was successful, 0 otherwise

5.1.2.2 isEqualDouble()

```
bool isEqualDouble (
            double lhs,
            double rhs )
```

Determines if two double precision floats are equal

Parameters

in	*lhs*	First number
in	*rhs*	Second number

Returns

true if numbers are equal, false otherwise

Note

Comparison is performed with the accuracy of EPSILON

5.1.2.3 printEquation()

```
void printEquation (
             double a,
             double b,
             double c )
```

Prints quadratic equation ax^2 + bx + c = 0 with given coefficients

Parameters

in	*a*	a-coefficient
in	*b*	b-coefficient
in	*c*	c-coefficient

5.1.2.4 printResult()

```
void printResult (
             NRoots nRoots,
             double x1,
             double x2 )
```

Prints program result to the standard output

Parameters

in	*nRoots*	Number of roots in the solution
in	*x1*	First root
in	*x2*	Second root

5.1.2.5 solveLinear()

```
double solveLinear (
             double a,
             double b )
```

Solves linear equation ax + b = 0

Parameters

in	*a*	a-coefficient
in	*b*	b-coefficient

Returns

Value of x

5.1.2.6 solveQuadratic()

```
NRoots solveQuadratic (
            double a,
            double b,
            double c,
            double * x1,
            double * x2 )
```

Solves the quadratic equation ax^2 + bx + c = 0

Parameters

in	*a*	a-coefficient
in	*b*	b-coefficient
in	*c*	c-coefficient
out	*x1*	Pointer to the 1st root
out	*x2*	Pointer to the 2nd root

Returns

Number of roots

Note

In the case of infinite roots returns INF_ROOTS

5.1.3 Variable Documentation

5.1.3.1 EPSILON

```
const double EPSILON = 1e-2
```

Determines the precision of comparison in isEqualDouble function.

5.2 quadratic.h

Go to the documentation of this file.

```
#ifndef QUADRATIC_H
#define QUADRATIC_H

const double EPSILON = 1e-2;

enum NRoots {
    ZERO,
    ONE,
    TWO,
    INF_ROOTS
};

//------------------------------------------------------------
//------------------------------------------------------------
NRoots solveQuadratic(double a, double b, double c, double *x1, double *x2);

//------------------------------------------------------------
//------------------------------------------------------------
double solveLinear(double a, double b);

//------------------------------------------------------------
//------------------------------------------------------------
bool isEqualDouble(double lhs, double rhs);

//------------------------------------------------------------
//------------------------------------------------------------
int getCoefsFromInput(double *a, double *b, double *c);

//------------------------------------------------------------
//------------------------------------------------------------
void printResult(NRoots nRoots, double x1, double x2);

//------------------------------------------------------------
//------------------------------------------------------------
void printEquation(double a, double b, double c);

#endif
```

5.3 include/test.h File Reference

```
#include <math.h>
```

Classes

- struct testCase

  Encapsulates the set of values needed for testing.

Typedefs

- typedef struct testCase tCase

  Encapsulates the set of values needed for testing.

Functions

- int checkTestCase (tCase test)
- void runTests (const char *path)

5.3.1 Typedef Documentation

5.3.1.1 tCase

```
typedef struct testCase tCase
```

Encapsulates the set of values needed for testing.

5.3.2 Function Documentation

5.3.2.1 checkTestCase()

```
int checkTestCase (
            tCase test )
```

Checks result of solveQuadratic function against test values

Parameters

in	*test*	Struct, encapsulating test info, equation coefficients and expected values

Returns

1 if test is successful, 0 otherwise

5.3.2.2 runTests()

```
void runTests (
            const char * path )
```

Parses test cases from a file and runs them with checkTestCase

Parameters

in	*path*	Path to file, containing test cases

5.4 test.h

Go to the documentation of this file.

```
#include <math.h>

typedef struct testCase {
    size_t id;
    double a;
    double b;
    double c;
    size_t nRoots;
     double x1;
     double x2;
} tCase;

//------------------------------------------------------------
//------------------------------------------------------------
int checkTestCase(tCase test);

//------------------------------------------------------------
//------------------------------------------------------------
void runTests(const char* path);
```

5.5 README.md File Reference

5.6 src/main.cpp File Reference

```
#include <stdio.h>
#include <math.h>
#include "../include/quadratic.h"
#include "../include/test.h"
```

Macros

- #define NDEBUG_MODE

Functions

- int main ()

5.6.1 Macro Definition Documentation

5.6.1.1 NDEBUG_MODE

```
#define NDEBUG_MODE
```

5.6.2 Function Documentation

5.6.2.1 main()

```
int main ( )
```

5.7 src/quadratic.cpp File Reference

```
#include <math.h>
#include <assert.h>
#include <stdio.h>
```

Functions

- int getCoefsFromInput (double ∗a, double ∗b, double ∗c)
- void printResult (NRoots nRoots, double x1, double x2)
- void printEquation (double a, double b, double c)
- bool isEqualDouble (double lhs, double rhs)
- NRoots solveQuadratic (double a, double b, double c, double ∗x1, double ∗x2)
- double solveLinear (double a, double b)

5.7.1 Function Documentation

5.7.1.1 getCoefsFromInput()

```
int getCoefsFromInput (
            double * a,
            double * b,
            double * c )
```

Gets equation coefficients from the standard input

Parameters

out	*a*	Pointer to a-coefficient
out	*b*	Pointer to b-coefficient
out	*c*	Pointer to c-coefficient

Returns

1 if operation was successful, 0 otherwise

5.7.1.2 isEqualDouble()

```
bool isEqualDouble (
             double lhs,
             double rhs )
```

Determines if two double precision floats are equal

Parameters

in	*lhs*	First number
in	*rhs*	Second number

Returns

true if numbers are equal, false otherwise

Note

Comparison is performed with the accuracy of EPSILON

5.7.1.3 printEquation()

```
void printEquation (
             double a,
             double b,
             double c )
```

Prints quadratic equation ax^2 + bx + c = 0 with given coefficients

Parameters

in	*a*	a-coefficient
in	*b*	b-coefficient
in	*c*	c-coefficient

5.7.1.4 printResult()

```
void printResult (
             NRoots nRoots,
             double x1,
             double x2 )
```

Prints program result to the standard output

Parameters

in	*nRoots*	Number of roots in the solution
in	*x1*	First root
in	*x2*	Second root

5.7.1.5 solveLinear()

```
double solveLinear (
            double a,
            double b )
```

Solves linear equation ax + b = 0

Parameters

in	*a*	a-coefficient
in	*b*	b-coefficient

Returns

Value of x

5.7.1.6 solveQuadratic()

```
NRoots solveQuadratic (
            double a,
            double b,
            double c,
            double * x1,
            double * x2 )
```

Solves the quadratic equation ax^2 + bx + c = 0

Parameters

in	*a*	a-coefficient
in	*b*	b-coefficient
in	*c*	c-coefficient
out	*x1*	Pointer to the 1st root
out	*x2*	Pointer to the 2nd root

Returns

Number of roots

Note

In the case of infinite roots returns INF_ROOTS

5.8 src/test.cpp File Reference

```
#include <math.h>
#include <stdio.h>
#include "../include/quadratic.h"
#include "../include/test.h"
```

Functions

- int checkTestCase (tCase test)
- void runTests (const char *path)

5.8.1 Function Documentation

5.8.1.1 checkTestCase()

```
int checkTestCase (
            tCase test )
```

Checks result of solveQuadratic function against test values

Parameters

in	*test*	Struct, encapsulating test info, equation coefficients and expected values

Returns

1 if test is successful, 0 otherwise

5.8.1.2 runTests()

```
void runTests (
            const char * path )
```

Parses test cases from a file and runs them with checkTestCase

Parameters

`in`	*path*	Path to file, containing test cases