# Quadratic Equation Solver

Generated by Doxygen 1.9.1

# Chapter 1

# File Index

## 1.1 File List

Here is a list of all files with brief descriptions:

# Chapter 2

# File Documentation

## 2.1 main.cpp File Reference

```
#include "quadr_solver.h"
```

**Functions**

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

### 2.1.1 Function Documentation

#### 2.1.1.1 main()

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

## 2.2 quadr_io.cpp File Reference

```
#include "quadr_solver.h"
```

## Functions

- void **getnum** (double ∗num)

    *Reads input data. If the data is not a valid number prints "This does not seem to be a valid number" and tries to read again.*
- void **print_roots** (int **roots_amount**, double ∗x1, double ∗x2)

    *Prints the solutions of the equation if they exits. Prints "x can be any real number." if there are infinite solutions and "No real solutions." if there are no real solutions.*
- void **clear_buffer** (void)

    *Clears input buffer using getchar() function.*
- void **ask_for_coeff** (char coeff_name, double ∗coeff_address)

    *Prints a prompt and reads input data using **getnum()** (p. 11) function.*
- void **get_coeffs** (double ∗a, double ∗b, double ∗c)

    *Gets coefficients of the equation.*

### 2.2.1  Function Documentation

#### 2.2.1.1  ask_for_coeff()

```
void ask_for_coeff (
            char coeff_name,
            double * coeff_address )
```

Prints a prompt and reads input data using **getnum()** (p. 11) function.

**Parameters**

| | | |
|---|---|---|
| in | *coeff_name* | Name of the coefficient. |
| in | *coeff_address* | Pointer to the coefficient. |

#### 2.2.1.2  clear_buffer()

```
void clear_buffer (
            void  )
```

Clears input buffer using getchar() function.

#### 2.2.1.3  get_coeffs()

```
void get_coeffs (
            double * a,
            double * b,
            double * c )
```

Gets coefficients of the equation.

**Parameters**

| out | *a* | Pointer to a-coefficient of the equation. |
|-----|-----|-------------------------------------------|
| out | *b* | Pointer to b-coefficient of the equation. |
| out | *c* | Pointer to c-coefficient of the equation. |

### 2.2.1.4 getnum()

```
void getnum (
            double * num )
```

Reads input data. If the data is not a valid number prints "This does not seem to be a valid number" and tries to read again.

**Parameters**

| in | *num* | Pointer to the number. |
|----|-------|------------------------|

### 2.2.1.5 print_roots()

```
void print_roots (
            int roots_amount,
            double * x1,
            double * x2 )
```

Prints the solutions of the equation if they exits. Prints "x can be any real number." if there are infinite solutions and "No real solutions." if there are no real solutions.

**Parameters**

| in | *roots_amount* | Amount of the solutions of the equation. |
|----|----------------|------------------------------------------|
| in | *x1*           | Pointer to x1.                           |
| in | *x2*           | Pointer to x2.                           |

## 2.3 quadr_solver.cpp File Reference

```
#include "quadr_solver.h"
```

## Functions

- int **solve_quadr** (double a, double b, double c, double ∗x1, double ∗x2)

*Solves equation ax$^\wedge$2 + bx + c = 0.*

- int **solve_lin** (double a, double b, double ∗x)

   *Solves equation ax + b = 0.*

- int **are_doubles_equal** (double n1, double n2)

   *Returns 1 if |n1 - n2| <= EPSILON and 0 otherwise.*

- double **max_of_two** (double n1, double n2)

   *Identifies which of two numbers is larger.*

- double **min_of_two** (double n1, double n2)

   *Identifies which of two numbers is smaller.*

### 2.3.1 Function Documentation

#### 2.3.1.1 are_doubles_equal()

```
int are_doubles_equal (
            double n1,
            double n2 )
```

Returns 1 if |n1 - n2| <= EPSILON and 0 otherwise.

**Parameters**

| in | *n1* | First number. |
|----|------|---------------|
| in | *n2* | Second number. |

**Returns**

Returns 1 or 0.

#### 2.3.1.2 max_of_two()

```
double max_of_two (
            double n1,
            double n2 )
```

Identifies which of two numbers is larger.

**Parameters**

| in | *n1* | First number. |
|----|------|---------------|
| in | *n2* | Second number. |

**Returns**

> Returns n1 if n1 $>$ n2 and n2 otherwise.

### 2.3.1.3 min_of_two()

```
double min_of_two (
            double n1,
            double n2 )
```

Identifies which of two numbers is smaller.

**Parameters**

| in | *n1* | First number. |
|----|------|---------------|
| in | *n2* | Second number. |

**Returns**

> Returns n2 if n1 $>$ n2 and n2 otherwise.

### 2.3.1.4 solve_lin()

```
int solve_lin (
            double a,
            double b,
            double * x )
```

Solves equation ax + b = 0.

**Parameters**

| in | *a* | a-coefficient of the equation. |
|-----|-----|-------------------------------|
| in | *b* | b-coefficient of the equation. |
| out | *x* | Pointer to the solution of the equation. |

**Returns**

> Returns the amount of the solutions of the equation and INF_ROOTS if the equation has infinite solutions.

### 2.3.1.5 solve_quadr()

```
int solve_quadr (
            double a,
```

```
        double b,
        double c,
        double * x1,
        double * x2 )
```

Solves equation $ax^2 + bx + c = 0$.

**Parameters**

| in | *a* | a-coefficient of the equation. |
|---|---|---|
| in | *b* | b-coefficient of the equation. |
| in | *c* | c-coefficient of the equation. |
| out | *x1* | Pointer to one of the solutions of the equation. |
| out | *x2* | Pointer to one of the solutions of the equation. |

**Returns**

Returns the amount of the solutions of the equation and INF_ROOTS if the equation has infinite solutions.

## 2.4 quadr_solver.h File Reference

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

### Macros

- #define **MY_ASSERT**(condition) if (!(condition)) printf("Error %s in line %d in function %s\n", #condition, __LINE__, __func__) \

  *If condition is false, prints an error message. Does not abort the program.*

### Enumerations

- enum **roots_amount** { **NO_ROOTS** , **ONE_ROOT** , **TWO_ROOTS** , **INF_ROOTS** = -1 }

  *Possible amounts of solutions of the equation.*

### Functions

- void **getnum** (double ∗num)

  *Reads input data. If the data is not a valid number prints "This does not seem to be a valid number" and tries to read again.*
- void **print_roots** (int **roots_amount**, double ∗x1, double ∗x2)

  *Prints the solutions of the equation if they exits. Prints "x can be any real number." if there are infinite solutions and "No real solutions." if there are no real solutions.*
- int **solve_quadr** (double a, double b, double c, double ∗x1, double ∗x2)

*Solves equation $ax^2 + bx + c = 0$.*

- int **solve_lin** (double a, double b, double ∗x)

  *Solves equation $ax + b = 0$.*

- int **are_doubles_equal** (double n1, double n2)

  *Returns 1 if $|n1 - n2| <=$ EPSILON and 0 otherwise.*

- void **clear_buffer** (void)

  *Clears input buffer using getchar() function.*

- void **ask_for_coeff** (char coeff_name, double ∗coeff_address)

  *Prints a prompt and reads input data using **getnum()** (p. 11) function.*

- void **get_coeffs** (double ∗a, double ∗b, double ∗c)

  *Gets coefficients of the equation.*

- double **max_of_two** (double n1, double n2)

  *Identifies which of two numbers is larger.*

- double **min_of_two** (double n1, double n2)

  *Identifies which of two numbers is smaller.*

- void **test_eq** (double a, double b, double c, int expected_nRoots, double expected_x1, double expected_x2)

  *Tests if **solve_quadr()** (p. 13) works as expected.*

- void **start_tests** (char ∗filename)

  *Launches the tests. By default gets test data from test_eq_data.txt file. File can be changed with a command line argument (see README.md). Prints an error message if the file was not found.*

## Variables

- const double **EPSILON** = 1e-7

  *Determines precision of **are_doubles_equal()** (p. 10).*

### 2.4.1 Macro Definition Documentation

#### 2.4.1.1 MY_ASSERT

```
#define MY_ASSERT(
            condition ) if (!(condition)) printf("Error %s in line %d in function %s\n",
#condition, __LINE__, __func__) \
```

If condition is false, prints an error message. Does *not* abort the program.

**Parameters**

| | | |
|---|---|---|
| in | *condition* | Condition. |

### 2.4.2 Enumeration Type Documentation

**2.4.2.1 roots_amount**

enum **roots_amount**

Possible amounts of solutions of the equation.

**Enumerator**

| NO_ROOTS | |
|---|---|
| ONE_ROOT | |
| TWO_ROOTS | |
| INF_ROOTS | |

## 2.4.3 Function Documentation

**2.4.3.1 are_doubles_equal()**

```
int are_doubles_equal (
            double n1,
            double n2 )
```

Returns 1 if |n1 - n2| <= EPSILON and 0 otherwise.

**Parameters**

| in | *n1* | First number. |
|---|---|---|
| in | *n2* | Second number. |

**Returns**

Returns 1 or 0.

**2.4.3.2 ask_for_coeff()**

```
void ask_for_coeff (
            char coeff_name,
            double * coeff_address )
```

Prints a prompt and reads input data using **getnum()** (p. 11) function.

**Parameters**

| in | *coeff_name* | Name of the coefficient. |
|---|---|---|
| in | *coeff_address* | Pointer to the coefficient. |

### 2.4.3.3 clear_buffer()

```
void clear_buffer (
            void  )
```

Clears input buffer using getchar() function.

### 2.4.3.4 get_coeffs()

```
void get_coeffs (
            double * a,
            double * b,
            double * c )
```

Gets coefficients of the equation.

**Parameters**

| out | *a* | Pointer to a-coefficient of the equation. |
|-----|-----|-------------------------------------------|
| out | *b* | Pointer to b-coefficient of the equation. |
| out | *c* | Pointer to c-coefficient of the equation. |

### 2.4.3.5 getnum()

```
void getnum (
            double * num )
```

Reads input data. If the data is not a valid number prints "This does not seem to be a valid number" and tries to read again.

**Parameters**

| in | *num* | Pointer to the number. |
|----|-------|------------------------|

### 2.4.3.6 max_of_two()

```
double max_of_two (
            double n1,
            double n2 )
```

Identifies which of two numbers is larger.

**Parameters**

| in | *n1* | First number. |
|----|------|---------------|
| in | *n2* | Second number. |

**Returns**

Returns n1 if n1 > n2 and n2 otherwise.

### 2.4.3.7 min_of_two()

```
double min_of_two (
            double n1,
            double n2 )
```

Identifies which of two numbers is smaller.

**Parameters**

| in | *n1* | First number. |
|----|------|---------------|
| in | *n2* | Second number. |

**Returns**

Returns n2 if n1 > n2 and n2 otherwise.

### 2.4.3.8 print_roots()

```
void print_roots (
            int roots_amount,
            double * x1,
            double * x2 )
```

Prints the solutions of the equation if they exits. Prints "x can be any real number." if there are infinite solutions and "No real solutions." if there are no real solutions.

**Parameters**

| in | *roots_amount* | Amount of the solutions of the equation. |
|----|----------------|------------------------------------------|
| in | *x1* | Pointer to x1. |
| in | *x2* | Pointer to x2. |

### 2.4.3.9 solve_lin()

```
int solve_lin (
            double a,
            double b,
            double * x )
```

Solves equation ax + b = 0.

**Parameters**

| in | a | a-coefficient of the equation. |
|---|---|---|
| in | b | b-coefficient of the equation. |
| out | x | Pointer to the solution of the equation. |

**Returns**

Returns the amount of the solutions of the equation and INF_ROOTS if the equation has infinite solutions.

### 2.4.3.10 solve_quadr()

```
int solve_quadr (
            double a,
            double b,
            double c,
            double * x1,
            double * x2 )
```

Solves equation $ax^2 + bx + c = 0$.

**Parameters**

| in | a | a-coefficient of the equation. |
|---|---|---|
| in | b | b-coefficient of the equation. |
| in | c | c-coefficient of the equation. |
| out | x1 | Pointer to one of the solutions of the equation. |
| out | x2 | Pointer to one of the solutions of the equation. |

**Returns**

Returns the amount of the solutions of the equation and INF_ROOTS if the equation has infinite solutions.

### 2.4.3.11 start_tests()

```
void start_tests (
            char * filename )
```

Launches the tests. By default gets test data from test_eq_data.txt file. File can be changed with a command line argument (see README.md). Prints an error message if the file was not found.

**Parameters**

| in | *filename* | Name of a file with test data. |
| --- | --- | --- |

### 2.4.3.12 test_eq()

```
void test_eq (
            double a,
            double b,
            double c,
            int expected_nRoots,
            double expected_x1,
            double expected_x2 )
```

Tests if **solve_quadr()** (p. 13) works as expected.

**Parameters**

| in | *a* | a-coefficient of the equation. |
| --- | --- | --- |
| in | *b* | b-coefficient of the equation. |
| in | *c* | c-coefficient of the equation. |
| in | *expected_nRoots* | Expected amount of the solutions of the equation. |
| in | *expected_x1* | Expected x1. |
| in | *expected_x2* | Expected x2. |

## 2.4.4 Variable Documentation

### 2.4.4.1 EPSILON

```
const double EPSILON = 1e-7
```

Determines precision of **are_doubles_equal()** (p. 10).

# 2.5 quadr_tests.cpp File Reference

```
#include "quadr_solver.h"
```

## Functions

- void **test_eq** (double a, double b, double c, int expected_nRoots, double expected_x1, double expected_x2)

    *Tests if **solve_quadr()** (p. 13) works as expected.*

- void **start_tests** (char ∗filename)

    *Launches the tests. By default gets test data from test_eq_data.txt file. File can be changed with a command line argument (see README.md). Prints an error message if the file was not found.*

## Variables

- const int **TEST_NUMBER_LENGTH** = 20

    *Determines maximal length of a number in test_eq_data.txt file.*

### 2.5.1  Function Documentation

#### 2.5.1.1  start_tests()

```
void start_tests (
            char * filename )
```

Launches the tests. By default gets test data from test_eq_data.txt file. File can be changed with a command line argument (see README.md). Prints an error message if the file was not found.

**Parameters**

| in | *filename* | Name of a file with test data. |
|----|------------|--------------------------------|

#### 2.5.1.2  test_eq()

```
void test_eq (
            double a,
            double b,
            double c,
            int expected_nRoots,
            double expected_x1,
            double expected_x2 )
```

Tests if **solve_quadr()** (p. 13) works as expected.

**Parameters**

| in | *a* | a-coefficient of the equation. |
|----|-----|--------------------------------|
| in | *b* | b-coefficient of the equation. |
| in | *c* | c-coefficient of the equation. |
| in | *expected_nRoots* | Expected amount of the solutions of the equation. |
| in | *expected_x1* | Expected x1. |
| in | *expected_x2* | Expected x2. |

## 2.5.2 Variable Documentation

### 2.5.2.1 TEST_NUMBER_LENGTH

```
const int TEST_NUMBER_LENGTH = 20
```

Determines maximal length of a number in test_eq_data.txt file.

# Index