MathProject

1.2

Generated by Doxygen 1.8.17

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 include/SqEq.hh File Reference

Macros

- #define SQEQ_ERROR 1

    return this from main on error
- #define MAX_ROOT_NUM 2

    maximal munber of roots
- #define COEFF_NUM 3

    number of coefficients
- #define EPS 1e-12

    epsilon

Enumerations

- enum ResType {
RT_ERROR, RT_INV_COEFF_ERROR, RT_NULLPTR_ERROR, RT_VALID,
RT_NO_ROOTS, RT_ONE_ROOT, RT_TWO_ROOTS, RT_INF_ROOTS }

    enum with result types of solve functions

Functions

- bool read_coeffs (const char ∗prompt, double coeffs[COEFF_NUM])

    Reading 3 coefficients from stdin.
- void print_res (int res_type, const double results[MAX_ROOT_NUM])

    Print squaree equations roots.
- int solve_sqeq (const double coeffs[COEFF_NUM], double results[MAX_ROOT_NUM])

    Solving square equation.
- int solve_linear (double b, double c, double ∗x_ptr)

    Solving linear equation.
- int is_equal (double n1, double n2)

    Compares two double numbers.
- int ret_code (int res_type)

    Generate program return code.
- bool unit_testing ()

    Testing SqEq functions.

2.1.1 Macro Definition Documentation

2.1.1.1 SQEQ_ERROR

```
#define SQEQ_ERROR 1
```

return this from main on error

Definition at line 8 of file SqEq.hh.

2.1.1.2 MAX_ROOT_NUM

```
#define MAX_ROOT_NUM 2
```

maximal munber of roots

Definition at line 13 of file SqEq.hh.

2.1.1.3 COEFF_NUM

```
#define COEFF_NUM 3
```

number of coefficients

Definition at line 18 of file SqEq.hh.

2.1.1.4 EPS

```
#define EPS 1e-12
```

epsilon

Definition at line 23 of file SqEq.hh.

2.1.2 Enumeration Type Documentation

2.1.2.1 ResType

```
enum ResType
```

enum with result types of solve functions

Enumerator

RT_ERROR	
RT_INV_COEFF_ERROR	
RT_NULLPTR_ERROR	
RT_VALID	
RT_NO_ROOTS	
RT_ONE_ROOT	
RT_TWO_ROOTS	
RT_INF_ROOTS	

Definition at line 28 of file SqEq.hh.

2.1.3 Function Documentation

2.1.3.1 read_coeffs()

```
bool read_coeffs (
            const char * prompt,
            double coeffs[COEFF_NUM] )
```

Reading 3 coefficients from stdin.

Parameters

in	*prompt*	message to user
out	*coeffs*	array for coeffs, length must be >= 3

Returns

true if all OK

false on error

Definition at line 13 of file SqEq.cc.

References COEFF_NUM.

2.1.3.2 print_res()

```
void print_res (
            int res_type,
            const double results[MAX_ROOT_NUM] )
```

Print squaree equations roots.

Parameters

in	*res_type*	defines error/number of roots
in	*results*	roots

Definition at line 92 of file SqEq.cc.

References RT_ERROR, RT_INF_ROOTS, RT_INV_COEFF_ERROR, RT_NO_ROOTS, RT_NULLPTR_ERROR, RT_ONE_ROOT, and RT_TWO_ROOTS.

2.1.3.3 solve_sqeq()

```
int solve_sqeq (
            const double coeffs[COEFF_NUM],
            double results[MAX_ROOT_NUM] )
```

Solving square equation.

Parameters

in	*coeffs*	array with coefficients, length must must be >= 3
out	*results*	array with roots, length must must be >= 2

Returns

result type

Definition at line 20 of file SqEq.cc.

References is_equal(), RT_ERROR, RT_INV_COEFF_ERROR, RT_NO_ROOTS, RT_NULLPTR_ERROR, RT_ONE_ROOT, RT_TWO_ROOTS, and solve_linear().

2.1.3.4 solve_linear()

```
int solve_linear (
            double b,
            double c,
            double * x_ptr )
```

Solving linear equation.

Parameters

in	*b*	coeff on x
in	*c*	free member
out	*x_ptr*	result pointer

Returns

resutl type

Definition at line 71 of file SqEq.cc.

References is_equal(), RT_INF_ROOTS, RT_NO_ROOTS, and RT_ONE_ROOT.

Referenced by solve_sqeq().

2.1.3.5 is_equal()

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

Compares two double numbers.

Parameters

in	*n1*	1st num
in	*n2*	2nd num

Returns

int

Definition at line 87 of file SqEq.cc.

References EPS.

Referenced by solve_linear(), and solve_sqeq().

2.1.3.6 ret_code()

```
int ret_code (
            int res_type )
```

Generate program return code.

Parameters

in	*res_type*	value returned by solve function

Returns

int

Definition at line 124 of file SqEq.cc.

References RT_VALID, and SQEQ_ERROR.

2.1.3.7 unit_testing()

```
bool unit_testing ( )
```

Testing SqEq functions.

Returns

true if all tests passed

false if tests not passed

2.2 SqEq.hh

```
/**
 * @brief return this from main on error
 */

#ifndef SQEQ
#define SQEQ

#define SQEQ_ERROR 1

/**
 * @brief maximal munber of roots
 */
#define MAX_ROOT_NUM 2

/**
 * @brief number of coefficients
 */
#define COEFF_NUM 3

/**
 * @brief epsilon
 */
#define EPS 1e-12

/**
 * @brief enum with result types of solve functions
 */
enum ResType
{
  RT_ERROR, // Errors from here
  RT_INV_COEFF_ERROR,
  RT_NULLPTR_ERROR,

  RT_VALID, // Valid codes from here
  RT_NO_ROOTS,
  RT_ONE_ROOT,
  RT_TWO_ROOTS,
  RT_INF_ROOTS
};

/**
 * @brief Reading 3 coefficients from stdin
 *
 * @param[in] prompt message to user
 * @param[out] coeffs array for coeffs, length must be >= 3
 * @return true if all OK
 * @return false on error
 */
```

```
bool read_coeffs(const char *prompt, double coeffs[COEFF_NUM]);

/**
 * @brief Print squaree equations roots
 *
 * @param[in] res_type defines error/number of roots
 * @param[in] results roots
 */
void print_res(int res_type, const double results[MAX_ROOT_NUM]);

/**
 * @brief Solving square equation
 *
 * @param[in] coeffs array with coefficients, length must must be >= 3
 * @param[out] results array with roots, length must must be >= 2
 * @return result type
 */
int solve_sqeq(const double coeffs[COEFF_NUM], double results[MAX_ROOT_NUM]);

/**
 * @brief Solving linear equation
 *
 * @param[in] b coeff on x
 * @param[in] c free member
 * @param[out] x_ptr result pointer
 * @return resutl type
 */
int solve_linear(double b, double c, double *x_ptr);

/**
 * @brief Compares two double numbers
 *
 * @param[in] n1 1st num
 * @param[in] n2 2nd num
 * @return int
 */
int is_equal(double n1, double n2);

/**
 * @brief Generate program return code
 *
 * @param[in] res_type value returned by solve function
 * @return int
 */
int ret_code(int res_type);

/**
 * @brief Testing SqEq functions
 *
 * @return true if all tests passed
 * @return false if tests not passed
 */
bool unit_testing();


#endif // SQEQ
```

2.3 lib/SqEq.cc File Reference

File with most important SqEq functions.

```
#include <assert.h>
#include <math.h>
#include <stdio.h>
#include "SqEq.hh"
```

Functions

- bool read_coeffs (const char *prompt, double coeffs[COEFF_NUM])

  Reading 3 coefficients from stdin.
- int solve_sqeq (const double coeffs[COEFF_NUM], double results[MAX_ROOT_NUM])

Solving square equation.
- int solve_linear (double b, double c, double ∗x_ptr)

  Solving linear equation.
- int is_equal (double n1, double n2)

  Compares two double numbers.
- void print_res (int res_type, const double results[MAX_ROOT_NUM])

  Print squaree equations roots.
- int ret_code (int res_type)

  Generate program return code.

2.3.1 Detailed Description

File with most important SqEq functions.

Author

Tako

Definition in file SqEq.cc.

2.3.2 Function Documentation

2.3.2.1 read_coeffs()

```
bool read_coeffs (
            const char * prompt,
            double coeffs[COEFF_NUM] )
```

Reading 3 coefficients from stdin.

Parameters

in	*prompt*	message to user
out	*coeffs*	array for coeffs, length must be >= 3

Returns

true if all OK

false on error

Definition at line 13 of file SqEq.cc.

References COEFF_NUM.

2.3.2.2 solve_sqeq()

```
int solve_sqeq (
            const double coeffs[COEFF_NUM],
            double results[MAX_ROOT_NUM] )
```

Solving square equation.

Parameters

in	*coeffs*	array with coefficients, length must must be >= 3
out	*results*	array with roots, length must must be >= 2

Returns

result type

Definition at line 20 of file SqEq.cc.

References is_equal(), RT_ERROR, RT_INV_COEFF_ERROR, RT_NO_ROOTS, RT_NULLPTR_ERROR, RT_ONE_ROOT, RT_TWO_ROOTS, and solve_linear().

2.3.2.3 solve_linear()

```
int solve_linear (
            double b,
            double c,
            double * x_ptr )
```

Solving linear equation.

Parameters

in	*b*	coeff on x
in	*c*	free member
out	*x_ptr*	result pointer

Returns

resutl type

Definition at line 71 of file SqEq.cc.

References is_equal(), RT_INF_ROOTS, RT_NO_ROOTS, and RT_ONE_ROOT.

Referenced by solve_sqeq().

2.3.2.4 is_equal()

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

Compares two double numbers.

Parameters

in	*n1*	1st num
in	*n2*	2nd num

Returns

int

Definition at line 87 of file SqEq.cc.

References EPS.

Referenced by solve_linear(), and solve_sqeq().

2.3.2.5 print_res()

```
void print_res (
            int res_type,
            const double results[MAX_ROOT_NUM] )
```

Print squaree equations roots.

Parameters

in	*res_type*	defines error/number of roots
in	*results*	roots

Definition at line 92 of file SqEq.cc.

References RT_ERROR, RT_INF_ROOTS, RT_INV_COEFF_ERROR, RT_NO_ROOTS, RT_NULLPTR_ERROR, RT_ONE_ROOT, and RT_TWO_ROOTS.

2.3.2.6 ret_code()

```
int ret_code (
            int res_type )
```

Generate program return code.

Parameters

in	*res_type*	value returned by solve function

Returns

int

Definition at line 124 of file SqEq.cc.

References RT_VALID, and SQEQ_ERROR.

2.4 SqEq.cc

```
/**
 * @file SqEq.cc
 * @author Tako
 * @brief File with most important SqEq functions
 */

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

#include "SqEq.hh"

bool read_coeffs(const char *prompt, double coeffs[COEFF_NUM])
{
  puts(prompt);
  int scanned = scanf("%lf %lf %lf", coeffs, coeffs + 1, coeffs + 2);
  return COEFF_NUM == scanned;
}

int solve_sqeq(const double coeffs[COEFF_NUM], double results[MAX_ROOT_NUM])
{
  if ((nullptr == coeffs) || (nullptr == results))
    return RT_NULLPTR_ERROR;

  double a = coeffs[0], b = coeffs[1], c = coeffs[2];
  if (!(isfinite(a) && isfinite(b) && isfinite(c)))
    return RT_INV_COEFF_ERROR;

  if (is_equal(a, 0))
    return (solve_linear(b, c, results));
  else if (is_equal(c, 0))
  {
    if (is_equal(b, 0))
    {
      results[0] = 0;
      results[1] = NAN;
      return RT_ONE_ROOT;
    }
    else
    {
      results[0] = 0;
      return (solve_linear(a, b, results + 1) + 1);
    }
  }

  else
  {
    double d = b * b - 4 * a * c;

    if (is_equal(d, 0))
    {
      results[0] = -b / (2 * a);
      results[1] = NAN;
      return RT_ONE_ROOT;
    }
    else if (d < 0)
    {
      return RT_NO_ROOTS;
    }
    else if (d > 0)
    {
      double sqrt_d = sqrt(d);
      results[0] = (-b + sqrt_d) / (2 * a);
      results[1] = (-b - sqrt_d) / (2 * a);
      return RT_TWO_ROOTS;
    }
  }
  return RT_ERROR;
}

int solve_linear(double b, double c, double *x_ptr)
{
  if (is_equal(b, 0))
    return is_equal(c, 0) ? RT_INF_ROOTS : RT_NO_ROOTS;
  else if (is_equal(c, 0))
  {
    *x_ptr = 0;
    return RT_ONE_ROOT;
  }
  else
  {
    *x_ptr = -c / b;
    return RT_ONE_ROOT;
  }
}

int is_equal(double n1, double n2)
{
  return (fabs(n1 - n2) < EPS);
}

void print_res(int res_type, const double results[MAX_ROOT_NUM])
{
  printf("Answer:\n");
  switch (res_type)
  {
  case RT_NO_ROOTS:
    printf("No roots\n");
    break;
  case RT_ONE_ROOT:
    printf("x = %lg\n", results[0]);
    break;
  case RT_TWO_ROOTS:
    printf("x1 = %lg\nx2 = %lg\n", results[0], results[1]);
    break;
  case RT_INF_ROOTS:
    printf("Infinite number of roots\n");
    break;
  case RT_INV_COEFF_ERROR:
    printf("INVALID COEFFICIENT VALUE\n");
    break;
  case RT_NULLPTR_ERROR:
    printf("COEFF OR ROOT ARRAY IS NULLPTR\n");
    break;
  case RT_ERROR:
    printf("ERROR\n");
    break;
  default:
    printf("UNKNOWN ERROR\n");
    break;
  }
}

int ret_code(int res_type)
{
  return (res_type >= RT_VALID) ? 0 : SQEQ_ERROR;
}
```

Index