Complex Number

0.1.2

Generated by Doxygen 1.7.5

Tue May 8 2012 11:24:55

Contents

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 complex Namespace Reference

Complex namespace.

4.1.1 Detailed Description

Complex namespace. This is the complex namespace. It contains all classes and functions for dealing with complex numbers.

4.2 math Namespace Reference

Math namespace.

4.2.1 Detailed Description

Math namespace. This is the math namespace. It contains all classes and functions pertaining to mathematics.

Chapter 5

Class Documentation

5.1 ComplexNumber Class Reference

A class template which supports complex numbers.

`#include <include/ComplexNumber.h>`

Public Member Functions

- ComplexNumber ()

  Default constructor for ComplexNumber.
- ComplexNumber (const T &real, const T &imaginary)

  Overloaded constructor for ComplexNumber.
- ComplexNumber (const ComplexNumber< T > &z)

  Copy constructor for ComplexNumber.
- virtual ~ComplexNumber ()

  Default destructor for ComplexNumber.
- ComplexNumber< T > operator+ () const

  Unary plus operator.
- ComplexNumber< T > operator- () const

  Unary minus operator.
- template<class U >
  operator ComplexNumber< U > () const

  Type conversion operator.
- template<class U >
  ComplexNumber< T > & operator= (const ComplexNumber< U > &z)

  Assingns a ComplexNumber to this ComplexNumber.

- template<class U >
  ComplexNumber< T > & operator= (const U &real)

  Assingns a real number to this ComplexNumber.
- template<class U >
  ComplexNumber< T > & operator+= (const ComplexNumber< U > &z)

  Adds and assigns a ComplexNumber to this.
- template<class U >
  ComplexNumber< T > & operator+= (const U &real)

  Adds and assigns a real value to this.
- template<class U >
  ComplexNumber< T > & operator-= (const ComplexNumber< U > &z)

  Subtracts and assigns a ComplexNumber to this.
- template<class U >
  ComplexNumber< T > & operator-= (const U &real)

  Subtracts and assigns a real value to this.
- template<class U >
  ComplexNumber< T > & operator*= (const ComplexNumber< U > &z)

  Multiplies and assigns a ComplexNumber to this.
- template<class U >
  ComplexNumber< T > & operator*= (const U &real)

  Multiplies and assigns a real value to this.
- template<class U >
  ComplexNumber< T > & operator/= (const ComplexNumber< U > &z)

  Divides and assigns a ComplexNumber to this.
- template<class U >
  ComplexNumber< T > & operator/= (const U &real)

  Divides and assigns a real value to this.
- T getReal () const

  Get the real part.
- T getImaginary () const

  Get the imaginary part.
- void setReal (const T &real)

  Set the real part.
- void setImaginary (const T &imaginary)

  Set the imaginary part.
- template<>
  const ComplexNumber< double > i (0, 1)

Static Public Member Functions

- static ComplexNumber< T > polar (const T &r, const T &theta)

  Construct a ComplexNumber from the given polar coordinates.

Static Public Attributes

- static const ComplexNumber < double > i

  A public constant representing the value i.

Private Attributes

- T real

  The value of the real part of the complex number.
- T imaginary

  The value of the non-real part of the complex number.

Friends

- template<class U >
  std::ostream & operator<< (std::ostream &out, const ComplexNumber< U > &z)

  Inserts a ComplexNumber into the ostream.
- template<class U >
  std::istream & operator>> (std::istream &in, ComplexNumber< U > &z)

  Extracts a ComplexNumber from the istream.

Related Functions

(Note that these are not member functions.)

- template<class T >
  T abs (const ComplexNumber< T > &z)

  Absolute value of a ComplexNumber.
- template<class T >
  T abs2 (const ComplexNumber< T > &z)

  Square absolute value of a ComplexNumber.
- template<class T >
  T arg (const ComplexNumber< T > &z)

  Argument of a ComplexNumber.
- template<class T >
  T logabs (const ComplexNumber< T > &z)

  Natural logarithm absolute value of a ComplexNumber.
- template<class T >
  ComplexNumber< T > conjugate (const ComplexNumber< T > &z)

Complex conjugate of a ComplexNumber.

- template<class T >
  ComplexNumber< T > exp (const ComplexNumber< T > &z)

  Exponential function of a ComplexNumber.
- template<class T >
  ComplexNumber< T > inverse (const ComplexNumber< T > &z)

  Inverse of a ComplexNumber.
- template<class T >
  ComplexNumber< T > log (const ComplexNumber< T > &z)

  Natural logarithm of a ComplexNumber.
- template<class T >
  ComplexNumber< T > log (const T &x)

  Natural logarithm of a real value.
- template<class T >
  ComplexNumber< T > log10 (const ComplexNumber< T > &z)

  Logarithm base 10 of a ComplexNumber.
- template<class T >
  ComplexNumber< T > log10 (const T &x)

  Logarithm base 10 of a real value.
- template<class T >
  ComplexNumber< T > logb (const ComplexNumber< T > &z, const ComplexNumber< T > &b)

  Logarithm base b of a ComplexNumber.
- template<class T >
  ComplexNumber< T > logb (const ComplexNumber< T > &z, const T &b)

  Logarithm base b of a ComplexNumber.
- template<class T >
  ComplexNumber< T > logb (const T &x, const ComplexNumber< T > &b)

  Logarithm base b of a real value.
- template<class T >
  ComplexNumber< T > logb (const T &x, const T &b)

  Logarithm base b of a real value.
- template<class T >
  ComplexNumber< T > pow (const ComplexNumber< T > &a, const ComplexNumber< T > &b)

  Exponentiation of ComplexNumbers.
- template<class T >
  ComplexNumber< T > pow (const ComplexNumber< T > &a, const T &b)

  Exponentiation of a ComplexNumber and real value.
- template<class T >
  ComplexNumber< T > pow (const T &a, const ComplexNumber< T > &b)

  Exponentiation of a real value and a ComplexNumber.

- template<class T >
  ComplexNumber< T > sqrt (const ComplexNumber< T > &z)

  Square root of a ComplexNumber.
- template<class T >
  ComplexNumber< T > sqrt (const T &x)

  Square root of a real value.
- template<class T >
  ComplexNumber< T > cbrt (const ComplexNumber< T > &z)

  Cube root of a ComplexNumber.
- template<class T >
  ComplexNumber< T > nthRoot (const ComplexNumber< T > &z, const int n)

  nth root of a ComplexNumber
- template<class T >
  std::vector< ComplexNumber< T > > nthRoots (const ComplexNumber< T > &z, const int n)

  nth roots of a ComplexNumber
- template<class T >
  void nthRoots (const ComplexNumber< T > &z, std::vector< ComplexNumber< T > > roots, const int n)

  nth roots of a ComplexNumber
- template<class T >
  ComplexNumber< T > sin (const ComplexNumber< T > &z)

  Sine of a ComplexNumber.
- template<class T >
  ComplexNumber< T > cos (const ComplexNumber< T > &z)

  Cosine of a ComplexNumber.
- template<class T >
  ComplexNumber< T > tan (const ComplexNumber< T > &z)

  Tangent of a ComplexNumber.
- template<class T >
  ComplexNumber< T > csc (const ComplexNumber< T > &z)

  Cosecant of a ComplexNumber.
- template<class T >
  ComplexNumber< T > sec (const ComplexNumber< T > &z)

  Secant of a ComplexNumber.
- template<class T >
  ComplexNumber< T > cot (const ComplexNumber< T > &z)

  Cotangent of a ComplexNumber.
- template<class T >
  ComplexNumber< T > asin (const ComplexNumber< T > &z)

  Arcsine of a ComplexNumber.

- template<class T >
  ComplexNumber< T > asin (const T &x)

  Arcsine of a real value.
- template<class T >
  ComplexNumber< T > acos (const ComplexNumber< T > &z)

  Arccosine of a ComplexNumber.
- template<class T >
  ComplexNumber< T > acos (const T &x)

  Arccosine of a real value.
- template<class T >
  ComplexNumber< T > atan (const ComplexNumber< T > &z)

  Arctangent of a ComplexNumber.
- template<class T >
  ComplexNumber< T > atan (const T &x)

  Arctangent of a real value.
- template<class T >
  ComplexNumber< T > acsc (const ComplexNumber< T > &z)

  Arccosecant of a ComplexNumber.
- template<class T >
  ComplexNumber< T > acsc (const T &x)

  Arccosecant of a real value.
- template<class T >
  ComplexNumber< T > asec (const ComplexNumber< T > &z)

  Arcsecant of a ComplexNumber.
- template<class T >
  ComplexNumber< T > asec (const T &x)

  Arcsecant of a real value.
- template<class T >
  ComplexNumber< T > acot (const ComplexNumber< T > &z)

  Arccotangent of a ComplexNumber.
- template<class T >
  ComplexNumber< T > acot (const T &x)

  Arccotangent of a real value.
- template<class T >
  ComplexNumber< T > sinh (const ComplexNumber< T > &z)

  Hyperbolic sine of a ComplexNumber.
- template<class T >
  ComplexNumber< T > cosh (const ComplexNumber< T > &z)

  Hyperbolic cosine of a ComplexNumber.
- template<class T >
  ComplexNumber< T > tanh (const ComplexNumber< T > &z)

Hyperbolic tangent of a ComplexNumber.

- template<class T >
  ComplexNumber< T > csch (const ComplexNumber< T > &z)

  Hyperbolic cosecant of a ComplexNumber.
- template<class T >
  ComplexNumber< T > sech (const ComplexNumber< T > &z)

  Hyperbolic secant of a ComplexNumber.
- template<class T >
  ComplexNumber< T > coth (const ComplexNumber< T > &z)

  Hyperbolic cotangent of a ComplexNumber.
- template<class T >
  ComplexNumber< T > asinh (const ComplexNumber< T > &z)

  Hyperbolic arsine of a ComplexNumber.
- template<class T >
  ComplexNumber< T > asinh (const T &x)

  Hyperbolic arsine of a real value.
- template<class T >
  ComplexNumber< T > acosh (const ComplexNumber< T > &z)

  Hyperbolic arcosine of a ComplexNumber.
- template<class T >
  ComplexNumber< T > acosh (const T &x)

  Hyperbolic arcosine of a real value.
- template<class T >
  ComplexNumber< T > atanh (const ComplexNumber< T > &z)

  Hyperbolic artangent of a ComplexNumber.
- template<class T >
  ComplexNumber< T > atanh (const T &x)

  Hyperbolic artangent of a real value.
- template<class T >
  ComplexNumber< T > acsch (const ComplexNumber< T > &z)

  Hyperbolic arcosecant of a ComplexNumber.
- template<class T >
  ComplexNumber< T > acsch (const T &x)

  Hyperbolic arcosecant of a real value.
- template<class T >
  ComplexNumber< T > asech (const ComplexNumber< T > &z)

  Hyperbolic arsecant of a ComplexNumber.
- template<class T >
  ComplexNumber< T > asech (const T &x)

  Hyperbolic arsecant of a real value.
- template<class T >
  ComplexNumber< T > acoth (const ComplexNumber< T > &z)

Hyperbolic arcotangent of a ComplexNumber.

- template<class T >
  ComplexNumber< T > acoth (const T &x)

  Hyperbolic arcotangent of a real value.
- template<class T >
  ComplexNumber< T > operator+ (const ComplexNumber< T > &lhs, const ComplexNumber< T > &rhs)

  Adds a ComplexNumber to another ComplexNumber.
- template<class T >
  ComplexNumber< T > operator+ (const ComplexNumber< T > &lhs, const T &rhs)

  Adds a real value to a ComplexNumber.
- template<class T >
  ComplexNumber< T > operator+ (const T &lhs, const ComplexNumber< T > &rhs)

  Adds a ComplexNumber to a real value.
- template<class T >
  ComplexNumber< T > operator- (const ComplexNumber< T > &lhs, const ComplexNumber< T > &rhs)

  Subtracts a ComplexNumber from another ComplexNumber.
- template<class T >
  ComplexNumber< T > operator- (const ComplexNumber< T > &lhs, const T &rhs)

  Subtracts a real value from a ComplexNumber.
- template<class T >
  ComplexNumber< T > operator- (const T &lhs, const ComplexNumber< T > &rhs)

  Subtracts a ComplexNumber from a real value.
- template<class T >
  ComplexNumber< T > operator* (const ComplexNumber< T > &lhs, const ComplexNumber< T > &rhs)

  Multiplies a ComplexNumber by another ComplexNumber.
- template<class T >
  ComplexNumber< T > operator* (const ComplexNumber< T > &lhs, const T &rhs)

  Multiplies a ComplexNumber by a real value.
- template<class T >
  ComplexNumber< T > operator* (const T &lhs, const ComplexNumber< T > &rhs)

  Multiplies a real value by a ComplexNumber.
- template<class T >
  ComplexNumber< T > operator/ (const ComplexNumber< T > &lhs, const ComplexNumber< T > &rhs)

Divides a ComplexNumber by another ComplexNumber.

- template<class T >
  ComplexNumber< T > operator/ (const ComplexNumber< T > &lhs, const T &rhs)

  Divides a ComplexNumber by a real value.
- template<class T >
  ComplexNumber< T > operator/ (const T &lhs, const ComplexNumber< T > &rhs)

  Divides a real value by a ComplexNumber.
- template<class T , class U >
  bool operator== (const ComplexNumber< T > &lhs, const ComplexNumber< U > &rhs)

  Compares a ComplexNumber to another ComplexNumber.
- template<class T , class U >
  bool operator== (const ComplexNumber< T > &lhs, const U &rhs)

  Compares a ComplexNumber to a real value.
- template<class T , class U >
  bool operator== (const T &lhs, const ComplexNumber< U > &rhs)

  Compares a real value to a ComplexNumber.
- template<class T , class U >
  bool operator!= (const ComplexNumber< T > &lhs, const ComplexNumber< U > &rhs)

  Compares a ComplexNumber to another ComplexNumber.
- template<class T , class U >
  bool operator!= (const ComplexNumber< T > &lhs, const U &rhs)

  Compares a ComplexNumber to a real value.
- template<class T , class U >
  bool operator!= (const T &lhs, const ComplexNumber< U > &rhs)

  Compares a real value to a ComplexNumber.

5.1.1 Detailed Description

A class template which supports complex numbers.

This class template manages numbers on the complex plane, providing a set of operators and corresponding template functions

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ComplexNumber::ComplexNumber ( )

Default constructor for ComplexNumber.

This constructor creates and initializes an empty ComplexNumber, with default values of zero for the real and imaginay parts

Definition at line 74 of file ComplexNumber.cpp.

5.1.2.2 ComplexNumber::ComplexNumber (const T & *real*, const T & *imaginary*)

Overloaded constructor for ComplexNumber.

This constructor creates and initializes a ComplexNumber with the specified real and imaginary values

Parameters

in	*real*	The value of the real component of this ComplexNumber
in	*imaginary*	The value of the non-real component of this ComplexNumber

Definition at line 80 of file ComplexNumber.cpp.

5.1.2.3 ComplexNumber::ComplexNumber (const ComplexNumber< T > & *z*)

Copy constructor for ComplexNumber.

This constructor creates and initializes a copy of a ComplexNumber

Parameters

in	*z*	The ComplexNumber to be copied

Definition at line 86 of file ComplexNumber.cpp.

5.1.2.4 ComplexNumber::~ComplexNumber ( ) [virtual]

Default destructor for ComplexNumber.

The destructor cleans up memory allocated by ComplexNumber. ComplexNumber does not allocate any memory, and therefore, the destructor does nothing.

Definition at line 92 of file ComplexNumber.cpp.

5.1.3 Member Function Documentation

5.1.3.1 T ComplexNumber::getImaginary ( ) const

Get the imaginary part.

Gets the non-real part of this ComplexNumber

Returns

The value of the non-real component of this ComplexNumber

Definition at line 229 of file ComplexNumber.cpp.

5.1.3.2 T ComplexNumber::getReal ( ) const

Get the real part.

Gets the real part of this ComplexNumber

Returns

The value of the real component of this ComplexNumber

Definition at line 223 of file ComplexNumber.cpp.

5.1.3.3 template<> const ComplexNumber< double > ComplexNumber< double >::i (0 , 1)

5.1.3.4 template<class U > ComplexNumber::operator ComplexNumber< U > ( ) const

Type conversion operator.

Returns this ComplexNumber type-casted as a different ComplexNumber

Returns

This ComplexNumber type-casted as a different ComplexNumber

Definition at line 116 of file ComplexNumber.cpp.

5.1.3.5 template<class U > ComplexNumber< T > & ComplexNumber::operator*= (const ComplexNumber< U > & *z*)

Multiplies and assigns a ComplexNumber to this.

Multiplies this ComplexNumber by another ComplexNumber and assigns the product to this ComplexNumber

$$z = z \times z_1$$

Parameters

in	*z*	The ComplexNumber by which this ComplexNumber will be multiplied

Returns

A reference to this ComplexNumber consisting of the product of this ComplexNumber and another ComplexNumber

Definition at line 178 of file ComplexNumber.cpp.

5.1.3.6 template<class U > ComplexNumber< T > & ComplexNumber::operator*= (const U & *real*)

Multiplies and assigns a real value to this.

Multiplies this ComplexNumber by a real value and assigns the product to this ComplexNumber

$$z = z \times x$$

Parameters

in	*real*	The real value by which this ComplexNumber will be multiplied

Returns

A reference to this ComplexNumber consisting of the product of this ComplexNumber and a real value

Definition at line 187 of file ComplexNumber.cpp.

5.1.3.7 ComplexNumber< T > ComplexNumber::operator+ ( ) const

Unary plus operator.

Returns the positive of this ComplexNumber

$$z_2 = +z_1 = a + bi$$

Returns

A ComplexNumber that is the positive of this ComplexNumber

Definition at line 104 of file ComplexNumber.cpp.

5.1.3.8 template<class U > ComplexNumber< T > & ComplexNumber::operator+= (const ComplexNumber< U > & *z*)

Adds and assigns a ComplexNumber to this.

Adds a ComplexNumber to this ComplexNumber and assigns the sum to this ComplexNumber

$$z = z + z_1$$

Parameters

in	*z*	The ComplexNumber which will be added to this ComplexNumber

Returns

A reference to this ComplexNumber consisting of the sum of this ComplexNumber and another ComplexNumber

Definition at line 144 of file ComplexNumber.cpp.

5.1.3.9 template<class U > ComplexNumber< T > & ComplexNumber::operator+= (const U & *real*)

Adds and assigns a real value to this.

Adds a real value to this ComplexNumber and assigns the sum to this ComplexNumber

$$z = z + x$$

Parameters

in	*real*	The real value which will be added to this ComplexNumber

Returns

A reference to this ComplexNumber consisting of the sum of this ComplexNumber and a real value

Definition at line 153 of file ComplexNumber.cpp.

5.1.3.10 ComplexNumber< T > ComplexNumber::operator- ( ) const

Unary minus operator.

Finds the negative of this ComplexNumber and returns it

$$z_2 = -z_1 = -a_1 - b_1 i$$

Returns

A ComplexNumber that is the negative of this ComplexNumber

Definition at line 110 of file ComplexNumber.cpp.

5.1.3.11 template<class U > ComplexNumber< T > & ComplexNumber::operator-= (const ComplexNumber< U > & *z*)

Subtracts and assigns a ComplexNumber to this.

Subtracts a ComplexNumber from this ComplexNumber and assigns the difference to this ComplexNumber

$$z = z - z_1$$

Parameters

in	*z*	The ComplexNumber which will be subtracted from this - ComplexNumber

Returns

A reference to this ComplexNumber consisting of the difference of this Complex-Number and another ComplexNumber

Definition at line 161 of file ComplexNumber.cpp.

5.1.3.12 template<class U > ComplexNumber< T > & ComplexNumber::operator-= (const U & *real*)

Subtracts and assigns a real value to this.

Subtracts a real value from this ComplexNumber and assigns the difference to this - ComplexNumber

$$z = z - x$$

Parameters

in	*real*	The real value which will be subtracted from this ComplexNumber

Returns

A reference to this ComplexNumber consisting of the difference of this ComplexNumber and a real value

Definition at line 170 of file ComplexNumber.cpp.

5.1.3.13 template<class U > ComplexNumber< T > & ComplexNumber::operator/= (const ComplexNumber< U > & *z*)

Divides and assigns a ComplexNumber to this.

Divides this ComplexNumber by another ComplexNumber and assigns the quotient to this ComplexNumber

$$z = \frac{z}{z_1}$$

Parameters

in	*z*	The ComplexNumber by which this ComplexNumber will be divided

Exceptions

DivisionByZeroException	if attempting to divide this ComplexNumber by a ComplexNumber z that is equal to zero, a DivisionByZeroException is thrown

Returns

A reference to this ComplexNumber consisting of the quotient of this ComplexNumber and another ComplexNumber

Definition at line 196 of file ComplexNumber.cpp.

5.1.3.14 template<class U > ComplexNumber< T > & ComplexNumber::operator/= (const U & *real*)

Divides and assigns a real value to this.

Divides this ComplexNumber by a real value and assigns the quotient to this ComplexNumber

$$z = \frac{z}{x}$$

Parameters

in	*real*	The real value by which this ComplexNumber will be divided

Exceptions

DivisionByZeroException	if attempting to divide this ComplexNumber by a value real that is equal to zero, a DivisionByZeroException is thrown

Returns

A reference to this ComplexNumber consisting of the quotient of this ComplexNumber and another ComplexNumber

Definition at line 211 of file ComplexNumber.cpp.

5.1.3.15 template<class U > ComplexNumber< T > & ComplexNumber::operator= (const ComplexNumber< U > & *z*)

Assingns a ComplexNumber to this ComplexNumber.

Assigns a ComplexNumber to this ComplexNumber, copying its private data

$$z = z_1$$

Parameters

in	*z*	The ComplexNumber which will be copied and assigned to this ComplexNumber

Returns

A reference to this ComplexNumber consisting of the copied contents of the -ComplexNumber

Definition at line 123 of file ComplexNumber.cpp.

5.1.3.16 template<class U > ComplexNumber< T > & ComplexNumber::operator= (const U & *real*)

Assingns a real number to this ComplexNumber.

Assigns a real number to this ComplexNumber

$$z = x$$

Parameters

in	*real*	The real value which will be copied and to this ComplexNumber

Returns

A reference to this ComplexNumber consisting of the copied contents of the real value

Definition at line 135 of file ComplexNumber.cpp.

5.1.3.17 ComplexNumber< T > ComplexNumber::polar (const T & *r*, const T & *theta*) [static]

Construct a ComplexNumber from the given polar coordinates.

Converts the polar form of a complex number to the rectangular form used in this class

$$z = re^{i\varphi} = r(\cos\varphi + i\sin\varphi)$$

Parameters

in	*r*	The magnitude of the complex number
in	*theta*	The angle of the complex number

Returns

A ComplexNumber

Definition at line 98 of file ComplexNumber.cpp.

5.1.3.18 void ComplexNumber::setImaginary (const T & *imaginary*)

Set the imaginary part.

Sets the non-real part of this ComplexNumber

Parameters

in	*imaginary*	The value of the non-real component of this ComplexNumber

Definition at line 241 of file ComplexNumber.cpp.

5.1.3.19 void ComplexNumber::setReal (const T & *real*)

Set the real part.

Sets the real part of this ComplexNumber

Parameters

in	*real*	The value of the real component of this ComplexNumber

Definition at line 235 of file ComplexNumber.cpp.

5.1.4 Friends And Related Function Documentation

5.1.4.1 template<class T> T abs (const ComplexNumber< T > & *z*) [related]

Absolute value of a ComplexNumber.

Computes the absolute value, magnitude, or modulus, of a ComplexNumber

$$r = |z| = \sqrt{a^2 + b^2}$$

Parameters

in	*z*	The ComplexNumber of which to compute its absolute value

Returns

The absolute value of a ComplexNumber

Definition at line 73 of file ComplexMath.cpp.

5.1.4.2 template<class T > T abs2 (const ComplexNumber< T > & z) [related]

Square absolute value of a ComplexNumber.

Computes the square of the absolute value, magnitude or modulus, of a ComplexNumber.

$$r^2 = |z|^2 = a^2 + b^2$$

Parameters

in	*z*	The ComplexNumber of which to compute its squared absolute value

Returns

The squared absolute value of a ComplexNumber

Definition at line 80 of file ComplexMath.cpp.

5.1.4.3 template<class T > ComplexNumber< T > acos (const ComplexNumber< T > & z) [related]

Arccosine of a ComplexNumber.

Computes the complex arccosine, or inverse cosine, of a ComplexNumber. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\arccos z = -i \ln\left(z + i\sqrt{1 - z^2}\right)$$

This is based on the exponential definition of cosine

$$\theta = \arccos z \Rightarrow z = \cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2}$$

So, to solve for θ, we form a quadratic equation

$$\begin{aligned} z &= \frac{e^{i\theta} + e^{-i\theta}}{2} \\ 2z &= e^{i\theta} + e^{-i\theta} \\ 2ze^{i\theta} &= e^{2i\theta} + 1 \\ 0 &= e^{2i\theta} - 2ze^{i\theta} + 1 \end{aligned}$$

Applying the quadratic formula,

$$e^{i\theta} = \frac{2z \pm \sqrt{(-2z)^2 - 4}}{2} = \frac{-2z \pm 2\sqrt{z^2 - 1}}{2} = z \pm \sqrt{z^2 - 1}$$

Choosing the positive branch,

$$\theta = \frac{\ln\left(z + \sqrt{z^2 - 1}\right)}{i} = -i\ln\left(z + \sqrt{z^2 - 1}\right)$$

Parameters

in	*z*	The ComplexNumber of which to compute the arccosine

Returns

An angle that is the arccosine of a ComplexNumber

Definition at line 361 of file ComplexMath.cpp.

5.1.4.4 template<class T > ComplexNumber< T > acos (const T & *x*) [related]

Arccosine of a real value.

Computes the complex arccosine, or inverse cosine, of a real value. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\arccos x = -i\ln\left(x + i\sqrt{1 - x^2}\right)$$

This is based on the exponential definition of cosine

$$\theta = \arccos x \Rightarrow x = \cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2}$$

So, to solve for θ, we form a quadratic equation

$$\begin{aligned} x &= \frac{e^{i\theta} + e^{-i\theta}}{2} \\ 2x &= e^{i\theta} + e^{-i\theta} \\ 2xe^{i\theta} &= e^{2i\theta} + 1 \\ 0 &= e^{2i\theta} - 2xe^{i\theta} + 1 \end{aligned}$$

Applying the quadratic formula,

$$e^{i\theta} = \frac{2x \pm \sqrt{(-2x)^2 - 4}}{2} = \frac{-2x \pm 2\sqrt{x^2-1}}{2} = x \pm \sqrt{x^2-1}$$

Choosing the positive branch,

$$\theta = \frac{\ln\left(x + \sqrt{x^2-1}\right)}{i} = -i\ln\left(x + \sqrt{x^2-1}\right)$$

This can be further simplified to deal with the specific values of x that return complex values. Namely, where $\{x \in \mathbb{R} : |x| > 1\}$.

$$\begin{aligned} \arccos x &= -i\ln\left(x + i\sqrt{1-x^2}\right) \\ &= -i\ln\left(x - \sqrt{x^2-1}\right) \\ &= -i\left(i\varphi + \ln\left|x - \sqrt{x^2-1}\right|\right) \\ &= \varphi - i\ln\left|x - \sqrt{x^2-1}\right| \\ &= \begin{cases} \pi - i\ln\left(-x + \sqrt{x^2-1}\right) & \text{if } x < -1 \\ 0 + i\ln\left(\frac{1}{x - \sqrt{x^2-1}}\right) & \text{if } x > 1 \end{cases} \\ &= \begin{cases} \pi - i\operatorname{arcosh}(-x) & \text{if } x < -1 \\ i\ln\left(\frac{1}{x - \sqrt{x^2-1}}\left(\frac{\sqrt{x^2-1}+x}{\sqrt{x^2-1}+x}\right)\right) & \text{if } x > 1 \end{cases} \\ &= \begin{cases} \pi - i\operatorname{arcosh}(-x) & \text{if } x < -1 \\ i\ln\left(x + \sqrt{x^2-1}\right) & \text{if } x > 1 \end{cases} \\ &= \begin{cases} \pi - i\operatorname{arcosh}(-x) & \text{if } x < -1 \\ i\operatorname{arcosh} x & \text{if } x > 1 \end{cases} \end{aligned}$$

Parameters

in	*x*	The real value of which to compute the arccosine

Returns

An angle that is the arccosine of a real value

Definition at line 371 of file ComplexMath.cpp.

5.1.4.5 template<class T > ComplexNumber< T > acosh (const ComplexNumber< T > & *z*) [related]

Hyperbolic arcosine of a ComplexNumber.

Computes the complex hyperbolic arcosine, or inverse hyperbolic cosine, of a ComplexNumber. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arcosh} z = \ln\left(z + \sqrt{z-1}\sqrt{z+1}\right)$$

This is based on the exponential definition of hyperbolic cosine

$$\alpha = \operatorname{arcosh} z \Rightarrow z = \cosh\alpha = \frac{e^{\alpha} + e^{-\alpha}}{2}$$

So, to solve for α, we form a quadratic equation

$$\begin{aligned} z &= \frac{e^{\alpha} + e^{-\alpha}}{2} \\ 2z &= e^{\alpha} + e^{-\alpha} \\ 2ze^{\alpha} &= e^{2\alpha} + 1 \\ 0 &= e^{2\alpha} - 2ze^{\alpha} + 1 \end{aligned}$$

Applying the quadratic formula,

$$e^{\alpha} = \frac{2z \pm \sqrt{(-2z)^2 - 4}}{2} = \frac{2z \pm 2\sqrt{z^2-1}}{2} = z \pm \sqrt{z^2-1} = z \pm \sqrt{z-1}\sqrt{z+1}$$

Choosing the positive branch,

$$\alpha = \ln\left(z + \sqrt{z-1}\sqrt{z+1}\right)$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic arcosine

Returns

An area that is the hyperbolic arcosine of a ComplexNumber

Definition at line 497 of file ComplexMath.cpp.

5.1.4.6 template<class T > ComplexNumber< T > acosh (const T & *x*) [related]

Hyperbolic arcosine of a real value.

Computes the complex hyperbolic arcosine, or inverse hyperbolic cosine, of a real value. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arcosh} x = \ln\left(x + \sqrt{x^2 - 1}\right)$$

This is based on the exponential definition of hyperbolic cosine

$$\alpha = \operatorname{arcosh} x \Rightarrow x = \cosh\alpha = \frac{e^{\alpha} + e^{-\alpha}}{2}$$

So, to solve for α, we form a quadratic equation

$$\begin{aligned} x &= \frac{e^{\alpha} + e^{-\alpha}}{2} \\ 2x &= e^{\alpha} + e^{-\alpha} \\ 2xe^{\alpha} &= e^{2\alpha} + 1 \\ 0 &= e^{2\alpha} - 2xe^{\alpha} + 1 \end{aligned}$$

Applying the quadratic formula,

$$e^{\alpha} = \frac{2x \pm \sqrt{(-2x)^2 - 4}}{2} = \frac{2x \pm 2\sqrt{x^2 - 1}}{2} = x \pm \sqrt{x^2 - 1}$$

Choosing the positive branch,

$$\alpha = \ln\left(x + \sqrt{x^2 - 1}\right)$$

Parameters

in	*x*	The real value of which to compute the hyperbolic arcosine

Returns

An area that is the hyperbolic arcosine of a real value

Definition at line 507 of file ComplexMath.cpp.

5.1.4.7 template<class T > ComplexNumber< T > acot (const ComplexNumber< T > & *z*) [related]

Arccotangent of a ComplexNumber.

Computes the complex arccotangent, or inverse cotangent, of a ComplexNumber. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\operatorname{arccot} z = \arctan \frac{1}{z} = \frac{i}{2} \ln \left(\frac{z-i}{z+i} \right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned} \theta = \operatorname{arccot} z \Rightarrow \cot \theta &= z \\ \tan \theta &= \frac{1}{z} \\ \theta &= \arctan \frac{1}{z} \\ \operatorname{arccot} z &= \arctan \frac{1}{z} \end{aligned}$$

Parameters

in	*z*	The ComplexNumber of which to compute the arccotangent

Returns

An angle that is the arccotangent of a ComplexNumber

Definition at line 426 of file ComplexMath.cpp.

5.1.4.8 template<class T > ComplexNumber< T > acot (const T & *x*) [related]

Arccotangent of a real value.

Computes the complex arccotangent, or inverse cotangent, of a real value. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

While the domain of the arccotangent function is $\mathbb{R}$, this function is included for the sake of completeness.

Parameters

in	*x*	The real value of which to compute the arccotangent

Returns

An angle that is the arccotangent of a real value

Definition at line 432 of file ComplexMath.cpp.

5.1.4.9 template<class T > ComplexNumber< T > acoth (const ComplexNumber< T > & *z*) [related]

Hyperbolic arcotangent of a ComplexNumber.

Computes the complex hyperbolic arcotangent, or inverse hyperbolic cotangent, of a ComplexNumber. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arcoth} z = \operatorname{artanh} \frac{1}{z} = \frac{1}{2} \ln \left(\frac{z+1}{z-1} \right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned} \alpha = \operatorname{arcoth} z \Rightarrow \coth \alpha &= z \\ \tanh \alpha &= \frac{1}{z} \\ \alpha &= \operatorname{artanh} \frac{1}{z} \\ \operatorname{arcoth} z &= \operatorname{artanh} \frac{1}{z} \end{aligned}$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic arcotangent

Returns

An area that is the hyperbolic arcotangent of a ComplexNumber

Definition at line 569 of file ComplexMath.cpp.

5.1.4.10 template<class T > ComplexNumber< T > acoth (const T & *x*) [related]

Hyperbolic arcotangent of a real value.

Computes the complex hyperbolic arcotangent, or inverse hyperbolic cotangent, of a real value. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arcoth} x = \operatorname{artanh} \frac{1}{x} = \frac{1}{2} \ln \left(\frac{x+1}{x-1} \right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned} \alpha = \operatorname{arcoth} x \Rightarrow \coth \alpha &= x \\ \tanh \alpha &= \frac{1}{x} \\ \alpha &= \operatorname{artanh} \frac{1}{x} \\ \operatorname{arcoth} x &= \operatorname{artanh} \frac{1}{x} \end{aligned}$$

Parameters

in	*x*	The real value of which to compute the hyperbolic arcotangent

Returns

An area that is the hyperbolic arcotangent of a real value

Definition at line 575 of file ComplexMath.cpp.

5.1.4.11 template<class T > ComplexNumber< T > acsc (const ComplexNumber< T > & *z*) [related]

Arccosecant of a ComplexNumber.

Computes the complex arccosecant, or inverse cosecant, of a ComplexNumber. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\operatorname{arccsc} z = \arcsin \frac{1}{z} = -i \ln \left(\frac{i}{z} + \sqrt{1 - \frac{1}{z^2}} \right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned} \theta = \operatorname{arccsc} z \Rightarrow \csc \theta &= z \\ \sin \theta &= \frac{1}{z} \\ \theta &= \arcsin \frac{1}{z} \\ \operatorname{arccsc} z &= \arcsin \frac{1}{z} \end{aligned}$$

Parameters

in	*z*	The ComplexNumber of which to compute the arccosecant

Returns

An angle that is the arccosecant of a ComplexNumber

Definition at line 402 of file ComplexMath.cpp.

5.1.4.12 template<class T > ComplexNumber< T > acsc (const T & *x*) [related]

Arccosecant of a real value.

Computes the complex arccosecant, or inverse cosecant, of a real value. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\operatorname{arccsc} x = \arcsin \frac{1}{x} = -i \ln \left(\frac{i}{x} + \sqrt{1 - \frac{1}{x^2}} \right)$$

This can be proven using the reciprocal identities as follows,

$$\theta = \operatorname{arccsc} x \Rightarrow \csc\theta = x$$
$$\sin\theta = \frac{1}{x}$$
$$\theta = \arcsin\frac{1}{x}$$
$$\operatorname{arccsc} x = \arcsin\frac{1}{x}$$

Parameters

in	*x*	The real value of which to compute the arccosecant

Returns

An angle that is the arccosecant of a real value

Definition at line 408 of file ComplexMath.cpp.

5.1.4.13 template<class T> ComplexNumber< T > acsch (const ComplexNumber< T > & *z*) [related]

Hyperbolic arcosecant of a ComplexNumber.

Computes the complex hyperbolic arcosecant, or inverse hyperbolic cosecant, of a ComplexNumber. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arcsch} z = \operatorname{arsinh}\frac{1}{z} = \ln\left(\sqrt{1+\frac{1}{z^2}} + \frac{1}{z}\right)$$

This can be proven using the reciprocal identities as follows,

$$\alpha = \operatorname{arcsch} z \Rightarrow \operatorname{csch}\alpha = z$$
$$\sinh\alpha = \frac{1}{z}$$
$$\alpha = \operatorname{arsinh}\frac{1}{z}$$
$$\operatorname{arcsch} z = \operatorname{arsinh}\frac{1}{z}$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic arcosecant

Returns

An area that is the hyperbolic arcosecant of a ComplexNumber

Definition at line 541 of file ComplexMath.cpp.

5.1.4.14 template<class T > ComplexNumber< T > acsch (const T & *x*) [related]

Hyperbolic arcosecant of a real value.

Computes the complex hyperbolic arcosecant, or inverse hyperbolic cosecant, of a real value. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arcsch} x = \operatorname{arsinh} \frac{1}{x} = \ln\left(\sqrt{1 + \frac{1}{x^2}} + \frac{1}{x}\right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned} \alpha = \operatorname{arcsch} x &\Rightarrow \operatorname{csch} \alpha = x \\ \sinh \alpha &= \frac{1}{x} \\ \alpha &= \operatorname{arsinh} \frac{1}{x} \\ \operatorname{arcsch} x &= \operatorname{arsinh} \frac{1}{x} \end{aligned}$$

Parameters

in	*x*	The real value of which to compute the hyperbolic arcosecant

Returns

An area that is the hyperbolic arcosecant of a real value

Definition at line 547 of file ComplexMath.cpp.

5.1.4.15 template<class T > T arg (const ComplexNumber< T > & z) [related]

Argument of a ComplexNumber.

Computes the argument, the angle between the positive real axis and the vector representing z.

$$\varphi = \arg z = \operatorname{atan2}(b,a)$$

Where

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \pi + \arctan(\frac{y}{x}) & \text{if } y \geq 0, x < 0 \\ -\pi + \arctan(\frac{y}{x}) & \text{if } y < 0, x < 0 \\ \frac{\pi}{2} & \text{if } y > 0, x = 0 \\ -\frac{\pi}{2} & \text{if } y < 0, x = 0 \\ \text{undefined} & \text{if } y = 0, x = 0 \end{cases}$$

Parameters

in	z	The ComplexNumber of which to compute its argument

Returns

The argument of a ComplexNumber

Definition at line 87 of file ComplexMath.cpp.

5.1.4.16 template<class T > ComplexNumber< T > asec (const ComplexNumber< T > & z) [related]

Arcsecant of a ComplexNumber.

Computes the complex arcsecant, or inverse secant, of a ComplexNumber. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\operatorname{arcsec} z = \arccos \frac{1}{z} = -i \ln \left(\frac{1}{z} + \sqrt{1 - \frac{i}{z^2}} \right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned}\theta = \operatorname{arcsec} z &\Rightarrow \sec\theta = z \\ \cos\theta &= \frac{1}{z} \\ \theta &= \arccos\frac{1}{z} \\ \operatorname{arcsec} z &= \arccos\frac{1}{z}\end{aligned}$$

Parameters

in	*z*	The ComplexNumber of which to compute the arcsecant

Returns

An angle that is the arcsecant of a ComplexNumber

Definition at line 414 of file ComplexMath.cpp.

5.1.4.17 template<class T > ComplexNumber< T > asec (const T & *x*) [related]

Arcsecant of a real value.

Computes the complex arcsecant, or inverse secant, of a real value. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\operatorname{arcsec} x = \arccos\frac{1}{x} = -i\ln\left(\frac{1}{x} + \sqrt{1 - \frac{i}{x^2}}\right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned}\theta = \operatorname{arcsec} x &\Rightarrow \sec\theta = x \\ \cos\theta &= \frac{1}{x} \\ \theta &= \arccos\frac{1}{x} \\ \operatorname{arcsec} x &= \arccos\frac{1}{x}\end{aligned}$$

Parameters

in	*x*	The real value of which to compute the arcsecant

Returns

An angle that is the arcsecant of a real value

Definition at line 420 of file ComplexMath.cpp.

5.1.4.18 template<class T > ComplexNumber< T > asech (const ComplexNumber< T > & *z*) [related]

Hyperbolic arsecant of a ComplexNumber.

Computes the complex hyperbolic arsecant, or inverse hyperbolic secant, of a ComplexNumber. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\alpha = \operatorname{arsech} z = \operatorname{arcosh} \frac{1}{z} = \ln\left(\sqrt{\frac{1}{z} - 1}\sqrt{\frac{1}{z} + 1} + \frac{1}{z}\right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned} \alpha = \operatorname{arsech} z &\Rightarrow \operatorname{sech} \alpha = z \\ \cosh \alpha &= \frac{1}{z} \\ \alpha &= \operatorname{arcosh} \frac{1}{z} \\ \operatorname{arsech} z &= \operatorname{arcosh} \frac{1}{z} \end{aligned}$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic arsecant

Returns

An area that is the hyperbolic arsecant of a ComplexNumber

Definition at line 553 of file ComplexMath.cpp.

5.1.4.19 template<class T > ComplexNumber< T > asech (const T & *x*) [related]

Hyperbolic arsecant of a real value.

Computes the complex hyperbolic arsecant, or inverse hyperbolic secant, of a real value. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arsech} x = \operatorname{arcosh} \frac{1}{x} = \ln\left(\sqrt{\frac{1}{x} - 1}\sqrt{\frac{1}{x} + 1} + \frac{1}{x}\right)$$

This can be proven using the reciprocal identities as follows,

$$\begin{aligned} \alpha = \operatorname{arsech} x \Rightarrow \operatorname{sech} \alpha &= x \\ \cosh \alpha &= \frac{1}{x} \\ \alpha &= \operatorname{arcosh} \frac{1}{x} \\ \operatorname{arsech} x &= \operatorname{arcosh} \frac{1}{x} \end{aligned}$$

Parameters

in	*x*	The real value of which to compute the hyperbolic arsecant

Returns

An area that is the hyperbolic arsecant of a real value

Definition at line 559 of file ComplexMath.cpp.

5.1.4.20 template<class T > ComplexNumber< T > asin (const ComplexNumber< T > & *z*) [related]

Arcsine of a ComplexNumber.

Computes the complex arcsine, or inverse sine, of a ComplexNumber. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\arcsin z = -i \ln\left(iz + \sqrt{1 - z^2}\right)$$

This is based on the exponential definition of sine

$$\theta = \arcsin z \Rightarrow z = \sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}$$

So, to solve for θ, we form a quadratic equation

$$\begin{aligned} z &= \frac{e^{i\theta} - e^{-i\theta}}{2i} \\ 2iz &= e^{i\theta} - e^{-i\theta} \\ 2ize^{i\theta} &= e^{2i\theta} - 1 \\ 0 &= e^{2i\theta} - 2ize^{i\theta} - 1 \end{aligned}$$

Applying the quadratic formula,

$$e^{i\theta} = \frac{2iz \pm \sqrt{(2iz)^2 + 4}}{2} = \frac{2iz \pm 2\sqrt{1 - z^2}}{2} = iz \pm \sqrt{1 - z^2}$$

Choosing the positive branch,

$$\theta = \frac{\ln\left(iz + \sqrt{1 - z^2}\right)}{i} = -i\ln\left(iz + \sqrt{1 - z^2}\right)$$

Parameters

in	*z*	The ComplexNumber of which to compute the arcsine

Returns

An angle that is the arcsine of a ComplexNumber

Definition at line 336 of file ComplexMath.cpp.

5.1.4.21 template<class T > ComplexNumber< T > asin (const T & *x*) [related]

Arcsine of a real value.

Computes the complex arcsine, or inverse sine, of a real value. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\arcsin x = -i\ln\left(ix + \sqrt{1 - x^2}\right)$$

This is based on the exponential definition of sine

$$\theta = \arcsin x \Rightarrow x = \sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}$$

So, to solve for θ, we form a quadratic equation

$$\begin{aligned} x &= \frac{e^{i\theta} - e^{-i\theta}}{2i} \\ 2ix &= e^{i\theta} - e^{-i\theta} \\ 2ixe^{i\theta} &= e^{2i\theta} - 1 \\ 0 &= e^{2i\theta} - 2ixe^{i\theta} - 1 \end{aligned}$$

Applying the quadratic formula,

$$e^{i\theta} = \frac{2ix \pm \sqrt{(2ix)^2 + 4}}{2} = \frac{2ix \pm 2\sqrt{1-x^2}}{2} = ix \pm \sqrt{1-x^2}$$

Choosing the positive branch,

$$\theta = \frac{\ln\left(ix + \sqrt{1-x^2}\right)}{i} = -i\ln\left(ix + \sqrt{1-x^2}\right)$$

This can be further simplified to deal with the specific values of x that return complex values. Namely, where $\{x \in \mathbb{R} : |x| > 1\}$.

$$
\begin{aligned}
\arcsin x &= -i\ln\left(ix+\sqrt{1-x^2}\right)\\
&= -i\ln\left(i\left(x+\sqrt{x^2-1}\right)\right)\\
&= -i\left(i\varphi+\ln\left|i\left(x+\sqrt{x^2-1}\right)\right|\right)\\
&= \varphi - i\ln\left|x+\sqrt{x^2-1}\right|\\
&= \begin{cases} -\frac{\pi}{2}+i\ln\left(\frac{1}{-x-\sqrt{x^2-1}}\right) & \text{if } x<-1\\ \frac{\pi}{2}-i\ln\left(x+\sqrt{x^2-1}\right) & \text{if } x>1 \end{cases}\\
&= \begin{cases} -\frac{\pi}{2}+i\ln\left(\frac{1}{-x-\sqrt{x^2-1}}\left(\frac{\sqrt{x^2-1}-x}{\sqrt{x^2-1}-x}\right)\right) & \text{if } x<-1\\ \frac{\pi}{2}-i\operatorname{arcosh} x & \text{if } x>1 \end{cases}\\
&= \begin{cases} -\frac{\pi}{2}+i\ln\left(-x+\sqrt{x^2-1}\right) & \text{if } x<-1\\ \frac{\pi}{2}-i\operatorname{arcosh} x & \text{if } x>1 \end{cases}\\
&= \begin{cases} -\frac{\pi}{2}+i\operatorname{arcosh}(-x) & \text{if } x<-1\\ \frac{\pi}{2}-i\operatorname{arcosh} x & \text{if } x>1 \end{cases}
\end{aligned}
$$

Parameters

in	*x*	The real value of which to compute the arcsine

Returns

An angle that is the arcsine of a real value

Definition at line 347 of file ComplexMath.cpp.

5.1.4.22 template<class T > ComplexNumber< T > asinh (const ComplexNumber< T > & *z*) [related]

Hyperbolic arsine of a ComplexNumber.

Computes the complex hyperbolic arsine, or inverse hyperbolic sine, of a ComplexNumber. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arsinh} z = \ln\left(z+\sqrt{z^2+1}\right)$$

This is based on the definition of hyperbolic sine

$$\alpha = \operatorname{arsinh} z \Rightarrow z = \sinh \alpha = \frac{e^{\alpha} - e^{-\alpha}}{2}$$

So, to solve for α, we form a quadratic equation

$$\begin{aligned} z &= \frac{e^{\alpha} - e^{-\alpha}}{2} \\ 2z &= e^{\alpha} - e^{-\alpha} \\ 2ze^{\alpha} &= e^{2\alpha} - 1 \\ 0 &= e^{2\alpha} - 2ze^{\alpha} - 1 \end{aligned}$$

Applying the quadratic formula,

$$e^{\alpha} = \frac{2z \pm \sqrt{(-2z)^2 + 4}}{2} = \frac{2z \pm 2\sqrt{z^2+1}}{2} = z \pm \sqrt{z^2+1}$$

Choosing the positive branch,

$$\alpha = \ln\left(z + \sqrt{z^2+1}\right)$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic arsine

Returns

An area that is the hyperbolic arsine of a ComplexNumber

Definition at line 481 of file ComplexMath.cpp.

5.1.4.23 template<class T > ComplexNumber< T > asinh (const T & *x*) [related]

Hyperbolic arsine of a real value.

Computes the complex hyperbolic arsine, or inverse hyperbolic sine, of a real value. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{arsinh} x = \ln\left(x + \sqrt{x^2+1}\right)$$

This is based on the definition of hyperbolic sine

$$\alpha = \operatorname{arsinh} x \Rightarrow x = \sinh \alpha = \frac{e^{\alpha} - e^{-\alpha}}{2}$$

So, to solve for α, we form a quadratic equation

$$\begin{aligned} x &= \frac{e^{\alpha} - e^{-\alpha}}{2} \\ 2x &= e^{\alpha} - e^{-\alpha} \\ 2xe^{\alpha} &= e^{2\alpha} - 1 \\ 0 &= e^{2\alpha} - 2xe^{\alpha} - 1 \end{aligned}$$

Applying the quadratic formula,

$$e^{\alpha} = \frac{2x \pm \sqrt{(-2x)^2 + 4}}{2} = \frac{2x \pm 2\sqrt{x^2+1}}{2} = x \pm \sqrt{x^2+1}$$

Choosing the positive branch,

$$\alpha = \ln\left(x + \sqrt{x^2+1}\right)$$

Parameters

in	*x*	The real value of which to compute the hyperbolic arsine

Returns

An area that is the hyperbolic arsine of a real value

Definition at line 491 of file ComplexMath.cpp.

5.1.4.24 template<class T > ComplexNumber< T > atan (const ComplexNumber< T > & *z*) [related]

Arctangent of a ComplexNumber.

Computes the complex arctangent, or inverse tangent, of a ComplexNumber. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

$$\arctan z = \frac{i}{2} \ln\left(\frac{i+z}{i-z}\right)$$

This is based on the exponential definition of tangent

$$\theta = \arctan z \Rightarrow z = \tan\theta = \frac{\sin\theta}{\cos\theta} = \frac{\frac{1}{2i}e^{i\theta} - e^{-i\theta}}{\frac{1}{2}e^{i\theta} + e^{-i\theta}} = \frac{e^{i\theta} - e^{-i\theta}}{i(e^{i\theta} + e^{-i\theta})}$$

So, to solve for θ, we form a quadratic equation

$$\begin{aligned} z &= \frac{e^{i\theta} - e^{-i\theta}}{i(e^{i\theta} + e^{-i\theta})} \\ iz(e^{i\theta} + e^{-i\theta}) &= e^{i\theta} - e^{-i\theta} \\ ize^{i\theta} + ize^{-i\theta} &= e^{i\theta} - e^{-i\theta} \\ ize^{2i\theta} + iz &= e^{2i\theta} - 1 \\ (iz-1)e^{2i\theta} &= -(iz+1) \\ e^{2i\theta} &= -\frac{iz+1}{iz-1} = \frac{i-z}{i+z} \end{aligned}$$

Choosing the positive branch,

$$\theta = \frac{\ln\left(\sqrt{\frac{i-z}{i+z}}\right)}{i} = -i\ln\left(\left(\frac{i+z}{i-z}\right)^{-\frac{1}{2}}\right) = \frac{i}{2}\ln\left(\frac{i+z}{i-z}\right)$$

Parameters

in	*z*	The ComplexNumber of which to compute the arctangent

Returns

An angle that is the arctangent of a ComplexNumber

Definition at line 385 of file ComplexMath.cpp.

5.1.4.25 template<class T > ComplexNumber< T > atan (const T & *x*) [related]

Arctangent of a real value.

Computes the complex arctangent, or inverse tangent, of a real value. This value represents an arclength of a sector of the unit circle $x^2 + y^2 = 1$.

While the domain of the arctangent function is $\mathbb{R}$, this function is included for the sake of completeness.

Parameters

in	*x*	The real value of which to compute the arctangent

Returns

An angle that is the arctangent of a real value

Definition at line 396 of file ComplexMath.cpp.

5.1.4.26 template<class T > ComplexNumber< T > atanh (const ComplexNumber< T > & *z*) [related]

Hyperbolic artangent of a ComplexNumber.

Computes the complex hyperbolic artangent, or inverse hyperbolic tangent, of a ComplexNumber. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{artanh} z = \frac{1}{2} \ln\left(\frac{1+z}{1-z}\right)$$

This is based on the exponential definition of hyperbolic tangent

$$\alpha = \operatorname{artanh} z \Rightarrow z = \tanh\alpha = \frac{\sinh\alpha}{\cosh\alpha} = \frac{\frac{1}{2}e^{\alpha} - e^{-\alpha}}{\frac{1}{2}e^{\alpha} + e^{-\alpha}} = \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}}$$

So, to solve for α, we form a quadratic equation

$$\begin{aligned} z &= \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}} \\ z(e^{\alpha} + e^{-\alpha}) &= e^{\alpha} - e^{-\alpha} \\ ze^{\alpha} + ze^{-\alpha} &= e^{\alpha} - e^{-\alpha} \\ ze^{2\alpha} + z &= e^{2\alpha} - 1 \\ (z-1)e^{2\alpha} &= -(z+1) \\ e^{2\alpha} &= -\frac{z+1}{z-1} = \frac{1+z}{1-z} \end{aligned}$$

Choosing the positive branch,

$$\alpha = \ln\left(\sqrt{\frac{1+z}{1-z}}\right) = \frac{1}{2}\ln\left(\frac{1+z}{1-z}\right)$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic artangent

Returns

An area that is the hyperbolic artangent of a ComplexNumber

Definition at line 521 of file ComplexMath.cpp.

5.1.4.27 template<class T > ComplexNumber< T > atanh (const T & *x*) [related]

Hyperbolic artangent of a real value.

Computes the complex hyperbolic artangent, or inverse hyperbolic tangent, of a real value. This value represents an area of a sector of the unit hyperbola $x^2 - y^2 = 1$.

$$\operatorname{artanh} x = \frac{1}{2} \ln\left(\frac{1+x}{1-x}\right)$$

This is based on the exponential definition of hyperbolic tangent

$$\alpha = \operatorname{artanh} x \Rightarrow x = \tanh\alpha = \frac{\sinh\alpha}{\cosh\alpha} = \frac{\frac{1}{2}e^{\alpha} - e^{-\alpha}}{\frac{1}{2}e^{\alpha} + e^{-\alpha}} = \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}}$$

So, to solve for α, we form a quadratic equation

$$\begin{aligned} x &= \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}} \\ x(e^{\alpha} + e^{-\alpha}) &= e^{\alpha} - e^{-\alpha} \\ xe^{\alpha} + xe^{-\alpha} &= e^{\alpha} - e^{-\alpha} \\ xe^{2\alpha} + x &= e^{2\alpha} - 1 \\ (x-1)e^{2\alpha} &= -(x+1) \\ e^{2\alpha} &= -\frac{x+1}{x-1} = \frac{1+x}{1-x} \end{aligned}$$

Choosing the positive branch,

$$\alpha = \ln\left(\sqrt{\frac{1+x}{1-x}}\right) = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$$

Parameters

in	*x*	The real value of which to compute the hyperbolic artangent

Returns

An area that is the hyperbolic artangent of a real value

Definition at line 531 of file ComplexMath.cpp.

5.1.4.28 template<class T > ComplexNumber< T > cbrt (const ComplexNumber< T > & *z*) [related]

Cube root of a ComplexNumber.

Computes the principle cube root of a ComplexNumber.

If $z = re^{i\varphi}$, where $r = |z|$ and $\varphi = \arg z$, then

$$\sqrt[3]{z} = \sqrt[3]{r}e^{i\frac{\varphi}{3}} = \left(a^2+b^2\right)^{\frac{1}{6}}\left(\cos\left(\frac{\varphi}{3}\right)+i\sin\left(\frac{\varphi}{3}\right)\right)$$

This is a result of de Moivre's formula,

$$(\cos x + i\sin x)^n = \cos nx + i\sin nx$$

which can be proven by Euler's formula,

$$e^{ix} = \cos x + i\sin x$$

and the rules of exponentiation,

$$(e^{ix})^n = e^{inx}$$

So,

$$e^{i(nx)} = (\cos x + i\sin x)^n = \cos nx + i\sin nx$$

Parameters

in	*z*	The ComplexNumber of which to compute the cube root

Returns

A ComplexNumber that is the cube root of a ComplexNumber

Definition at line 242 of file ComplexMath.cpp.

5.1.4.29 template<class T > ComplexNumber< T > conjugate (const ComplexNumber< T > & z) [related]

Complex conjugate of a ComplexNumber.

Finds the complex conjugate of a ComplexNumber

$$\bar{z} = a - bi$$

Parameters

in	z	The ComplexNumber of which to find the complex conjugate.

Returns

A ComplexNumber that is the complex conjugate of another ComplexNumber

Definition at line 105 of file ComplexMath.cpp.

5.1.4.30 template<class T > ComplexNumber< T > cos (const ComplexNumber< T > & z) [related]

Cosine of a ComplexNumber.

Computes the complex cosine of a ComplexNumber.

$$\cos z = \cos(a+bi) = \cos a \cos(bi) - \sin a \sin(bi) = \cos a \cosh b - i \sin a \sinh b$$

This is a result of the addition or subtraction theorems or formulae, which can be proven by Euler's formula, $e^{ix} = \cos x + i \sin x$, and by noting the symmetry of the sine and cosine functions as follows,

$$\begin{aligned}
\cos(\alpha \pm \beta) + i \sin(\alpha \pm \beta) &= e^{i(\alpha \pm \beta)} \\
&= e^{i\alpha} e^{\pm i\beta} \\
&= (\cos\alpha + i \sin\alpha)(\cos\beta \pm i \sin\beta) \\
&= (\cos\alpha \cos\beta \mp \sin\alpha \sin\beta) + i(\sin\alpha \cos\beta \pm \cos\alpha \sin\beta)
\end{aligned}$$

Thus,

$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$$
$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$$

Since, by the exponential definitions of the trigonometric functions derived from Euler's formula and the definitions of the hyperbolic functions,

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i}$$
$$\cos x = \frac{e^{ix} + e^{-ix}}{2}$$
$$\sinh x = \frac{e^{x} - e^{-x}}{2}$$
$$\cosh x = \frac{e^{x} + e^{-x}}{2}$$

Then,

$$\sin(xi) = \frac{e^{i^2 x} - e^{-i^2 x}}{2i} = \frac{e^{-x} - e^{x}}{2i} = \frac{e^{x} - e^{-x}}{2} i = i\sinh x$$
$$\cos(xi) = \frac{e^{i^2 x} + e^{-i^2 x}}{2} = \frac{e^{-x} + e^{x}}{2} = \cosh x$$

Parameters

in	*z*	The ComplexNumber of which to compute the cosine

Returns

A ComplexNumber that is the cosine of a ComplexNumber

Definition at line 300 of file ComplexMath.cpp.

5.1.4.31 template<class T > ComplexNumber< T > cosh (const ComplexNumber< T > & *z*) [related]

Hyperbolic cosine of a ComplexNumber.

Computes the complex hyperbolic cosine of a ComplexNumber.

$$\cosh z = \cosh(a+bi) = \cosh a \cos(bi) + \sinh a \sin(bi) = \cosh a \cos b + i \sinh a \sin b$$

This is a result of the addition or subtraction theorems or formulae,

$$\begin{aligned}\sinh(x \pm y) &= \sinh x \cosh y \pm \cosh x \sinh y \\ \cosh(x \pm y) &= \cosh x \cosh y \pm \sinh x \sinh y\end{aligned}$$

which can be proven as follows, also noting the symmetry of the hyperbolic sine and hyperbolic cosine functions,

$$\begin{aligned}\sinh(x+y) =& \frac{e^{x+y} - e^{-x-y}}{2} \\ =& \frac{e^x e^y - e^x e^{-y} + e^x e^{-y} - e^{-x} e^{-y}}{2} \\ =& e^x \left(\frac{e^y - e^{-y}}{2} \right) + e^{-y} \left(\frac{e^x - e^{-x}}{2} \right) \\ =& (\cosh x + \sinh x) \sinh y + (\cosh y - \sinh y) \sinh x \\ =& \cosh x \sinh y + \sinh x \sinh y + \sinh x \cosh y - \sinh x \sinh y \\ =& \sinh x \cosh y + \cosh x \sinh y\end{aligned}$$

$$\begin{aligned}\cosh(x+y) =& \frac{e^{x+y} + e^{-x-y}}{2} \\ =& \frac{e^x e^y - e^x e^{-y} + e^x e^{-y} + e^{-x} e^{-y}}{2} \\ =& e^x \left(\frac{e^y - e^{-y}}{2} \right) + e^{-y} \left(\frac{e^x + e^{-x}}{2} \right) \\ =& (\cosh x + \sinh x) \sinh y + (\cosh y - \sinh y) \cosh x \\ =& \cosh x \sinh y + \sinh x \sinh y + \cosh x \cosh y - \cosh x \sinh y \\ =& \cosh x \cosh y + \sinh x \sinh y\end{aligned}$$

Euler's formula,

$$e^{ix} = \cos x + i \sin x$$

Since, by the exponential definitions of the trigonometric functions derived from Euler's formula and the definitions of the hyperbolic functions,

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i}$$
$$\cos x = \frac{e^{ix} + e^{-ix}}{2}$$
$$\sinh x = \frac{e^{x} - e^{-x}}{2}$$
$$\cosh x = \frac{e^{x} + e^{-x}}{2}$$

Then,

$$\sinh(xi) = \frac{e^{xi} - e^{-xi}}{2} = i\sin x$$
$$\cosh(xi) = \frac{e^{xi} + e^{-xi}}{2} = \cos x$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic cosine

Returns

A ComplexNumber that is the hyperbolic cosine of a ComplexNumber

Definition at line 445 of file ComplexMath.cpp.

5.1.4.32 template<class T > ComplexNumber< T > cot (const ComplexNumber< T > & *z*) [related]

Cotangent of a ComplexNumber.

Computes the complex cotangent of a ComplexNumber.

$$\cot z = \frac{1}{\tan z}$$

Parameters

in	*z*	The ComplexNumber of which to compute the cotangent

Returns

A ComplexNumber that is the cotangent of a ComplexNumber

Definition at line 330 of file ComplexMath.cpp.

5.1.4.33 template<class T > ComplexNumber< T > coth (const ComplexNumber< T > & z) [related]

Hyperbolic cotangent of a ComplexNumber.

Computes the complex hyperbolic cotangent of a ComplexNumber.

$$\coth z = \frac{1}{\tanh z}$$

Parameters

in	z	The ComplexNumber of which to compute the hyperbolic cotangent

Returns

A ComplexNumber that is the hyperbolic cotangent of a ComplexNumber

Definition at line 475 of file ComplexMath.cpp.

5.1.4.34 template<class T > ComplexNumber< T > csc (const ComplexNumber< T > & z) [related]

Cosecant of a ComplexNumber.

Computes the complex cosecant of a ComplexNumber.

$$\csc z = \frac{1}{\sin z}$$

Parameters

in	z	The ComplexNumber of which to compute the cosecant

Returns

A ComplexNumber that is the cosecant of a ComplexNumber

Definition at line 318 of file ComplexMath.cpp.

5.1.4.35 template<class T > ComplexNumber< T > csch (const ComplexNumber< T > & z) [related]

Hyperbolic cosecant of a ComplexNumber.

Computes the complex hyperbolic cosecant of a ComplexNumber.

$$\operatorname{csch} z = \frac{1}{\sinh z}$$

Parameters

in	z	The ComplexNumber of which to compute the hyperbolic cosecant

Returns

A ComplexNumber that is the hyperbolic cosecant of a ComplexNumber

Definition at line 463 of file ComplexMath.cpp.

5.1.4.36 template<class T > ComplexNumber< T > exp (const ComplexNumber< T > & z) [related]

Exponential function of a ComplexNumber.

Computes the e raised to a ComplexNumber.

$$\exp(z) = e^z = e^{a+bi} = e^a e^{bi} = e^a(\cos b + i \sin b) = e^a \cos b + ie^a \sin b$$

This comes from Euler's formula,

$$e^{ix} = \cos x + i \sin x$$

Parameters

in	z	The ComplexNumber to which e is raised

Returns

A ComplexNumber that is the exponent of a ComplexNumber

Definition at line 111 of file ComplexMath.cpp.

5.1.4.37 template<class T > ComplexNumber< T > inverse (const ComplexNumber< T > & z) [related]

Inverse of a ComplexNumber.

Finds the inverse, or reciprocal, of a ComplexNumber.

$$z^{-1} = \frac{1}{z} = \frac{1}{a+bi} = \frac{1}{a+bi}\frac{a-bi}{a-bi} = \frac{a-bi}{a^2+b^2} = \frac{\bar{z}}{|z|^2}$$

Parameters

in	z	The ComplexNumber of which the inverse is to be found

Returns

A ComplexNumber that is the inverse of a ComplexNumber

Definition at line 118 of file ComplexMath.cpp.

5.1.4.38 template<class T > ComplexNumber< T > log (const ComplexNumber< T > & z) [related]

Natural logarithm of a ComplexNumber.

Computes the natural logarithm of a ComplexNumber.

If $z = re^{i\varphi}$, where $r = |z|$ and $\varphi = \arg z$, then

$$\ln z = \ln\left(re^{i\varphi}\right) = \ln r + i\varphi$$

Parameters

in	z	The ComplexNumber of which the natural logarithm is to be found

Returns

A ComplexNumber that is the natural logarithm of a ComplexNumber

Definition at line 128 of file ComplexMath.cpp.

5.1.4.39 template<class T > ComplexNumber< T > log (const T & *x*) [related]

Natural logarithm of a real value.

Computes the natural logarithm of a real value. This is useful for when $x < 0$.

If $x = re^{i\varphi}$, where $r = |x|$ and $\varphi = \arg x$, then

$$\ln x = \ln\left(re^{i\varphi}\right) = \ln r + i\varphi$$

For values $x > 0, \arg x = 0$, and for values $x < 0, \arg x = -\pi$.

Parameters

in	*x*	The real value of which the natural logarithm is to be found

Returns

A ComplexNumber that is the natural logarithm of a real value

Definition at line 137 of file ComplexMath.cpp.

5.1.4.40 template<class T > ComplexNumber< T > log10 (const ComplexNumber< T > & *z*) [related]

Logarithm base 10 of a ComplexNumber.

Computes the base 10 logarithm of a ComplexNumber.

If $z = re^{i\varphi}$, where $r = |z|$ and $\varphi = \arg z$, then

$$\log_{10} z = \log_{10}(re^{i\varphi}) = \frac{\ln r + i\varphi}{\ln 10} = \frac{\ln z}{\ln 10}$$

This uses the change of base formula, which says

$$\log_b x = \frac{\log_k x}{\log_k b}$$

By the definition of a logarithm,

$$x = b^a \iff \log_b x = a$$

and using the base k logarithm,

$$\log_k x = \log_k b^a = a \log_k b \iff \frac{\log_k x}{\log_k b} = a$$

Since the natural logarithm is defined by the standard C library, and the desired base is 10, we use $k = e$ and $b = 10$.

Parameters

in	z	The ComplexNumber of which the natural logarithm is to be found

Returns

A ComplexNumber that is the base 10 logarithm of a ComplexNumber

Definition at line 146 of file ComplexMath.cpp.

5.1.4.41 template<class T > ComplexNumber< T > log10 (const T & x) [related]

Logarithm base 10 of a real value.

Computes the base 10 logarithm of a real value. This is useful for when $x < 0$.

If $x = re^{i\varphi}$, where $r = |x|$ and $\varphi = \arg x$, then

$$\log_{10} z = \log_{10}(re^{i\varphi}) = \frac{\ln r + i\varphi}{\ln 10} = \frac{\ln z}{\ln 10}$$

This uses the change of base formula, which says

$$\log_b x = \frac{\log_k x}{\log_k b}$$

By the definition of a logarithm,

$$x = b^a \iff \log_b x = a$$

and using the base k logarithm,

$$\log_k x = \log_k b^a = a \log_k b \iff \frac{\log_k x}{\log_k b} = a$$

Since the natural logarithm is defined by the standard C library, and the desired base is 10, we use $k = e$ and $b = 10$.

Parameters

in	*x*	The real value of which the natural logarithm is to be found

Returns

A ComplexNumber that is the natural logarithm of a real value

Definition at line 152 of file ComplexMath.cpp.

5.1.4.42 template<class T > T logabs (const ComplexNumber< T > & *z*) [related]

Natural logarithm absolute value of a ComplexNumber.

Computes the natural logarithm of the absolute value, magnitude, or modulus of a - ComplexNumber.

$$x = \ln|z| = \ln\left(\sqrt{a^2+b^2}\right) = \frac{1}{2}\ln\left(a^2+b^2\right) = \frac{1}{2}\ln|z|^2$$

Parameters

in	*z*	The ComplexNumber of which to compute the natural logarithm of its absolute value

Returns

The argument of a ComplexNumber

Definition at line 96 of file ComplexMath.cpp.

5.1.4.43 template<class T > ComplexNumber< T > logb (const ComplexNumber< T > & *z*, const ComplexNumber< T > & *b*) [related]

Logarithm base b of a ComplexNumber.

Computes the complex base b logarithm of a ComplexNumber.

If $z = r_a e^{i\varphi_a}$ and $b = r_b e^{i\varphi_b}$, where $r_a = |z|$ and $\varphi_a = \arg z$ and $r_b = |b|$ and $\varphi_b = \arg b$, then

$$\log_b z = \log_b(r_a e^{i\varphi_a}) = \frac{\ln r_a + i\varphi_a}{\ln b} = \frac{\ln r_a + i\varphi_a}{\ln r_b + i\varphi_b} = \frac{\ln z}{\ln b}$$

This uses the change of base formula, which says

$$\log_b x = \frac{\log_k x}{\log_k b}$$

By the definition of a logarithm,

$$x = b^a \iff \log_b x = a$$

and using the base k logarithm,

$$\log_k x = \log_k b^a = a \log_k b \iff \frac{\log_k x}{\log_k b} = a$$

Since the natural logarithm is defined by the standard C library, and the desired base is b, we use $k = e$ and $b = b$.

Parameters

in	*z*	The ComplexNumber of which the logarithm base b is to be found
in	*b*	The ComplexNumber that is the base of the logarithm

Returns

A ComplexNumber that is the base b logarithm of a ComplexNumber

Definition at line 162 of file ComplexMath.cpp.

5.1.4.44 template<class T > ComplexNumber< T > logb (const ComplexNumber< T > & *z*, const T & *b*) [related]

Logarithm base b of a ComplexNumber.

Computes the real base b logarithm of a ComplexNumber. This is usefule for when $b < 0$.

If $z = r_a e^{i\varphi_a}$ and $b = r_b e^{i\varphi_b}$, where $r_a = |z|$ and $\varphi_a = \arg z$ and $r_b = |b|$ and $\varphi_b = \arg b$, then

$$\log_b z = \log_b(r_a e^{i\varphi_a}) = \frac{\ln r_a + i\varphi_a}{\ln b} = \frac{\ln r_a + i\varphi_a}{\ln r_b + i\varphi_b} = \frac{\ln z}{\ln b}$$

This uses the change of base formula, which says

$$\log_b x = \frac{\log_k x}{\log_k b}$$

By the definition of a logarithm,

$$x = b^a \iff \log_b x = a$$

and using the base k logarithm,

$$\log_k x = \log_k b^a = a \log_k b \iff \frac{\log_k x}{\log_k b} = a$$

Since the natural logarithm is defined by the standard C library, and the desired base is b, we use $k = e$ and $b = b$.

Parameters

in	*z*	The ComplexNumber of which the logarithm base b is to be found
in	*b*	The real value that is the base of the logarithm

Returns

A ComplexNumber that is the base b logarithm of a ComplexNumber

Definition at line 168 of file ComplexMath.cpp.

5.1.4.45 template<class T > ComplexNumber< T > logb (const T & *x*, const ComplexNumber< T > & *b*) [related]

Logarithm base b of a real value.

Computes the complex base b logarithm of a real value. This is usefule for when $x < 0$.

If $z = r_a e^{i\varphi_a}$ and $b = r_b e^{i\varphi_b}$, where $r_a = |z|$ and $\varphi_a = \arg z$ and $r_b = |b|$ and $\varphi_b = \arg b$, then

$$\log_b z = \log_b (r_a e^{i\varphi_a}) = \frac{\ln r_a + i\varphi_a}{\ln b} = \frac{\ln r_a + i\varphi_a}{\ln r_b + i\varphi_b} = \frac{\ln z}{\ln b}$$

This uses the change of base formula, which says

$$\log_b x = \frac{\log_k x}{\log_k b}$$

By the definition of a logarithm,

$$x = b^a \iff \log_b x = a$$

and using the base k logarithm,

$$\log_k x = \log_k b^a = a \log_k b \iff \frac{\log_k x}{\log_k b} = a$$

Since the natural logarithm is defined by the standard C library, and the desired base is b, we use $k = e$ and $b = b$.

Parameters

in	*x*	The real value of which the logarithm base b is to be found
in	*b*	The ComplexNumber that is the base of the logarithm

Returns

A ComplexNumber that is the base b logarithm of a real value

Definition at line 174 of file ComplexMath.cpp.

5.1.4.46 template<class T > ComplexNumber< T > logb (const T & *x*, const T & *b*) [related]

Logarithm base b of a real value.

Computes the real base b logarithm of a real value. This is useful for when $x < 0$ and/or $b < 0$.

If $x = r_a e^{i\varphi_a}$ and $b = r_b e^{i\varphi_b}$, where $r_a = |x|$ and $\varphi_a = \arg x$ and $r_b = |b|$ and $\varphi_b = \arg b$, then

$$\log_b z = \log_b (r_a e^{i\varphi_a}) = \frac{\ln r_a + i\varphi_a}{\ln b} = \frac{\ln r_a + i\varphi_a}{\ln r_b + i\varphi_b} = \frac{\ln z}{\ln b}$$

This uses the change of base formula, which says

$$\log_b x = \frac{\log_k x}{\log_k b}$$

By the definition of a logarithm,

$$x = b^a \iff \log_b x = a$$

and using the base k logarithm,

$$\log_k x = \log_k b^a = a \log_k b \iff \frac{\log_k x}{\log_k b} = a$$

Since the natural logarithm is defined by the standard C library, and the desired base is b, we use $k = e$ and $b = b$.

Parameters

in	*x*	The real value of which the logarithm base b is to be found
in	*b*	The real value that is the base of the logarithm

Returns

A ComplexNumber that is the base b logarithm of a real value

Definition at line 180 of file ComplexMath.cpp.

5.1.4.47 template<class T> ComplexNumber<T> nthRoot (const ComplexNumber<T> & *z*, const int *n*) [related]

nth root of a ComplexNumber

Computes the principle nth root of a ComplexNumber.

If $z = re^{i\varphi}$, where $r = |z|$ and $\varphi = \arg z$, then

$$z^{\frac{1}{n}} = r^{\frac{1}{n}} e^{i\frac{\varphi}{n}} = \left(a^2 + b^2\right)^{\frac{1}{2n}} \left(\cos\left(\frac{\varphi}{n}\right) + i \sin\left(\frac{\varphi}{n}\right)\right)$$

This is a result of de Moivre's formula,

$$(\cos x + i \sin x)^n = \cos nx + i \sin nx$$

which can be proven by Euler's formula,

$$e^{ix} = \cos x + i \sin x$$

and the rules of exponentiation,

$$(e^{ix})^n = e^{inx}$$

So,

$$e^{i(nx)} = (\cos x + i \sin x)^n = \cos nx + i \sin nx$$

Parameters

in	*z*	The ComplexNumber of which to compute the nth root
in	*n*	The root to which the ComplexNumber is evaluated

Returns

A ComplexNumber that is the nth root of a ComplexNumber

Definition at line 253 of file ComplexMath.cpp.

5.1.4.48 template<class T > std::vector< ComplexNumber< T > > nthRoots (const ComplexNumber< T > & *z*, const int *n*) [related]

nth roots of a ComplexNumber

Computes the nth roots of a ComplexNumber.

If $z = re^{i\varphi}$, where $r = |z|$ and $\varphi = \arg z$, then

$$z^{\frac{1}{n}} = r^{\frac{1}{n}} e^{i\frac{\varphi}{n}} = \left(a^2+b^2\right)^{\frac{1}{2n}} \left(\cos\left(\frac{\varphi+2k\pi}{n}\right) + i\sin\left(\frac{\varphi+2k\pi}{n}\right)\right)$$

Where $\{k \in \mathbb{Z} | 0 \leq k < n\}$

This is a result of de Moivre's formula,

$$(\cos x + i\sin x)^n = \cos nx + i\sin nx$$

which can be proven by Euler's formula,

$$e^{ix} = \cos x + i\sin x$$

and the rules of exponentiation,

$$(e^{ix})^n = e^{inx}$$

So,

$$e^{i(nx)} = (\cos x + i\sin x)^n = \cos nx + i\sin nx$$

Parameters

in	*z*	The ComplexNumber of which to compute the nth roots
in	*n*	The root to which the ComplexNumber is evaluated

Returns

A vector containing the ComplexNumbers that are the nth roots of a ComplexNumber

Definition at line 267 of file ComplexMath.cpp.

5.1.4.49 template<class T > void nthRoots (const ComplexNumber< T > & *z*, std::vector< ComplexNumber< T > > *roots*, const int *n*) [related]

nth roots of a ComplexNumber

Computes the nth roots of a ComplexNumber.

If $z = re^{i\varphi}$, where $r = |z|$ and $\varphi = \arg z$, then

$$z^{\frac{1}{n}} = r^{\frac{1}{n}} e^{i\frac{\varphi}{n}} = \left(a^2+b^2\right)^{\frac{1}{2n}} \left(\cos\left(\frac{\varphi+2k\pi}{n}\right) + i\sin\left(\frac{\varphi+2k\pi}{n}\right)\right)$$

Where $\{k \in \mathbb{Z} | 0 \leq k < n\}$

This is a result of de Moivre's formula,

$$(\cos x + i\sin x)^n = \cos nx + i\sin nx$$

which can be proven by Euler's formula,

$$e^{ix} = \cos x + i\sin x$$

and the rules of exponentiation,

$$(e^{ix})^n = e^{inx}$$

So,

$$e^{i(nx)} = (\cos x + i\sin x)^n = \cos nx + i\sin nx$$

Parameters

in	*z*	The ComplexNumber of which to compute the nth roots
in	*roots*	The vector containing the ComplexNumbers that are the nth roots of a ComplexNumber
in	*n*	The root to which the ComplexNumber is evaluated

5.1.4.50 template<class T , class U > bool operator!= (const ComplexNumber< T > & *lhs*, const ComplexNumber< U > & *rhs*) [related]

Compares a ComplexNumber to another ComplexNumber.

Determines if the ComplexNumber on the left side of the operand is not equal to the ComplexNumber on the right side of the operand

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand to which the ComplexNumber on the right will be compared
in	*rhs*	The ComplexNumber on the right side of the operand, which will be compared to the the ComplexNumber on the left side of the operand

Returns

A boolean indicating whether the ComplexNumber on the left side of the operand is not equal to the ComplexNumber on the right side of the operand

Definition at line 355 of file ComplexNumber.cpp.

5.1.4.51 template<class T , class U > bool operator!= (const ComplexNumber< T > & *lhs*, const U & *rhs*) [related]

Compares a ComplexNumber to a real value.

Determines if the ComplexNumber on the left side of the operand is not equal to the real value on the right side of the operand

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand to which the real value on the right will be compared
in	*rhs*	The real value on the right side of the operand, which will be compared to the the ComplexNumber on the left side of the operand

Returns

A boolean indicating whether the ComplexNumber on the left side of the operand is not equal to the real value on the right side of the operand

Definition at line 361 of file ComplexNumber.cpp.

5.1.4.52 template<class T , class U > bool operator!= (const T & *lhs*, const ComplexNumber< U > & *rhs*) [related]

Compares a real value to a ComplexNumber.

Determines if the real value on the left side of the operand is not equal to the ComplexNumber on the right side of the operand

Parameters

in	*lhs*	The real value on the left side of the operand to which the ComplexNumber on the right will be compared
in	*rhs*	The ComplexNumber on the right side of the operand, which will be compared to the the real value on the left side of the operand

Returns

A boolean indicating whether the real value on the left side of the operand is not equal to the ComplexNumber on the right side of the operand

Definition at line 366 of file ComplexNumber.cpp.

5.1.4.53 template<class T > ComplexNumber< T > operator* (const ComplexNumber< T > & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Multiplies a ComplexNumber by another ComplexNumber.

Multiplies a ComplexNumber on the left side of the operand by a ComplexNumber on the left side of the operand, returning the resultant product as a third ComplexNumber.

$$z_3 = z_1 z_2 = (a_1 + b_1 i)(a_2 + b_2 i) = a_1 a_2 + (b_1 a_2)i + (a_1 b_2)i + (b_1 b_2)i^2 = (a_1 a_2 - b_1 b_2) + (b_1 a_2 + a_1 b_2)i$$

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand, by which the ComplexNumber on the right side of the operand will be multiplied
in	*rhs*	The ComplexNumber on the right side of the operand, by which the ComplexNumber on the left side of the operand will be multiplied

Returns

A ComplexNumber consisting of the product of the two ComplexNumbers

Definition at line 280 of file ComplexNumber.cpp.

5.1.4.54 template<class T > ComplexNumber< T > operator* (const ComplexNumber< T > & *lhs*, const T & *rhs*) [related]

Multiplies a ComplexNumber by a real value.

Multiplies the real and imaginary values of the ComplexNumber on the left side of the operand by a real number on the right side of the operand, returning the resultant product as a third ComplexNumber.

$$z_2 = z_1 x = (a_1 + b_1 i)x = (a_1 x) + (b_1 x)i$$

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand, by which the real value on the right side of the operand will be multiplied
in	*rhs*	The real value on the right side of the operand, by which the ComplexNumber on the left side of the operand will be multiplied

Returns

A ComplexNumber consisting of the product of the ComplexNumber and the real value

Definition at line 289 of file ComplexNumber.cpp.

5.1.4.55 template<class T > ComplexNumber< T > operator* (const T & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Multiplies a real value by a ComplexNumber.

Multiplies a real value on the left side of the operand by the real and imaginary values of a ComplexNumber on the right side of the operand, returning the resultant product as a third ComplexNumber.

$$z_3 = x z_1 = x(a_1 + b_1 i) = (x a_1) + (x b_1)i$$

Parameters

in	*lhs*	The real value on the left side of the operand, by which the ComplexNumber on the right side of the operand will be multiplied
in	*rhs*	The ComplexNumber on the right side of the operand, by which the ComplexNumber on the left side of the operand will be multiplied

Returns

A ComplexNumber consisting of the product of the ComplexNumber and the real value

Definition at line 294 of file ComplexNumber.cpp.

5.1.4.56 template<class T > ComplexNumber< T > operator+ (const ComplexNumber< T > & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Adds a ComplexNumber to another ComplexNumber.

Adds the real and imaginary values of a ComplexNumber on the right and left side of the operand, returning the resultant sum as a third ComplexNumber.

$$z_3 = z_1 + z_2 = (a_1 + b_1 i) + (a_2 + b_2 i) = (a_1 + a_2) + (b_1 + b_2)i$$

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand, to which the ComplexNumber on the right side of the operand will be added
in	*rhs*	The ComplexNumber on the right side of the operand, which will be added to the ComplexNumber on the left side of the operand

Returns

A ComplexNumber consisting of the sum of the two ComplexNumbers

Definition at line 246 of file ComplexNumber.cpp.

5.1.4.57 template<class T > ComplexNumber< T > operator+ (const ComplexNumber< T > & *lhs*, const T & *rhs*) [related]

Adds a real value to a ComplexNumber.

Adds the real value of a ComplexNumber on the left side of the operand and a real number on the right side of the operand, returning the resultant sum as a third ComplexNumber.

$$z_2 = z_1 + x = (a_1 + b_1 i) + x = (a_1 + x) + b_1 i$$

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand, to which the real value on the right side of the operand will be added
in	*rhs*	The real value on the right side of the operand, which will be added to the ComplexNumber on the left side of the operand

Returns

A ComplexNumber consisting of the sum of the ComplexNumber and the real value

Definition at line 253 of file ComplexNumber.cpp.

5.1.4.58 template<class T > ComplexNumber< T > operator+ (const T & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Adds a ComplexNumber to a real value.

Adds a real value on the left side of the operand the real value of a ComplexNumber on the right side of the operand, returning the resultant sum as a third ComplexNumber.

$$z_2 = x + z_1 = x + (a_1 + b_1 i) = (x + a_1) + b_1 i$$

Parameters

in	*lhs*	The real value on the right side of the operand, to which the ComplexNumber on the right side of the operand will be added
in	*rhs*	The ComplexNumber on the right side of the operand, which will be added to the real value on the left side of the operand

Returns

A ComplexNumber consisting of the sum of the ComplexNumber and the real value

Definition at line 258 of file ComplexNumber.cpp.

5.1.4.59 template<class T > ComplexNumber< T > operator- (const ComplexNumber< T > & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Subtracts a ComplexNumber from another ComplexNumber.

Subtracts the real and imaginary values of a ComplexNumber on the right side of the operand from those of the ComplexNumber on the left side of the operand, returning the resultant difference as a third ComplexNumber.

$$z_3 = z_1 - z_2 = (a_1 + b_1 i) - (a_2 + b_2 i) = (a_1 - a_2) + (b_1 - b_2)i$$

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand, from which the ComplexNumber on the right side of the operand will be subtracted
in	*rhs*	The ComplexNumber on the right side of the operand, which will be subtracted from the ComplexNumber on the left side of the operand

Returns

A ComplexNumber consisting of the difference of the two ComplexNumbers

Definition at line 263 of file ComplexNumber.cpp.

5.1.4.60 template<class T > ComplexNumber< T > operator- (const ComplexNumber< T > & *lhs*, const T & *rhs*) [related]

Subtracts a real value from a ComplexNumber.

Subtracts a real value on the right side of the operand from the real value of the -ComplexNumber on the left side of the operand, returning the resultant difference as a third ComplexNumber.

$$z_2 = z_1 - x = (a_1 + b_1 i) - x = (a_1 - x) + b_1 i$$

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand, from which the real value on the right side of the operand will be subtracted
in	*rhs*	The real value on the right side of the operand, which will be subtracted from the ComplexNumber on the left side of the operand

Returns

A ComplexNumber consisting of the difference of the ComplexNumber and the real value

Definition at line 270 of file ComplexNumber.cpp.

5.1.4.61 template<class T > ComplexNumber< T > operator- (const T & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Subtracts a ComplexNumber from a real value.

Subtracts the real value of the ComplexNumber on the right side of the operand from a real value on the left side of the operand, returning the resultant difference as a third ComplexNumber.

$$z_2 = x - z_1 = x - (a_1 + b_1 i) = (x - a_1) + b_1 i$$

Parameters

in	*lhs*	The real value on the left side of the operand, from which the ComplexNumber on the right side of the operand will be subtracted
in	*rhs*	The ComplexNumber on the right side of the operand, which will be subtracted from the real value on the left side of the operand

Returns

A ComplexNumber consisting of the difference of the ComplexNumber and the real value

Definition at line 275 of file ComplexNumber.cpp.

5.1.4.62 template<class T > ComplexNumber< T > operator/ (const ComplexNumber< T > & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Divides a ComplexNumber by another ComplexNumber.

Divides the ComplexNumber on the left side of the operand by the ComplexNumber on the right side of the operand, using the complex conjugate of the denominator, returning the resultant quotient as a third ComplexNumber.

$$z_3 = \frac{z_1}{z_2} = \frac{a_1 + b_1 i}{a_2 + b_2 i} = \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} + \frac{b_1 a_2 - a_1 b_2}{a_2^2 + b_2^2} i$$

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand, by which the ComplexNumber on the right side of the operand will be divided
in	*rhs*	The ComplexNumber on the right side of the operand, which will divide the ComplexNumber on the left side of the operand

Exceptions

DivisionByZero-Exception	if attempting to divide a ComplexNumber lhs by a ComplexNumber rhs that is equal to zero, a DivisionByZeroException is thrown

Returns

A ComplexNumber consisting of the quotient of the two ComplexNumbers

Definition at line 299 of file ComplexNumber.cpp.

5.1.4.63 template<class T > ComplexNumber< T > operator/ (const ComplexNumber< T > & *lhs*, const T & *rhs*) [related]

Divides a ComplexNumber by a real value.

Divides the real and imaginary values of a ComplexNumber on the left side of the operand by a real value on the right side of the operand, returning the resultant quotient as a third ComplexNumber.

$$z_2 = \frac{z_1}{x} = \frac{a_1 + b_1 i}{x} = \frac{a_1}{x} + \frac{b_1}{x} i$$

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand, by which the real value on the right side of the operand will be divided
in	*rhs*	The real value on the right side of the operand, which will divide the ComplexNumber on the left side of the operand

Exceptions

DivisionByZero-Exception	if attempting to divide a ComplexNumber lhs by a value rhs that is equal to zero, a DivisionByZeroException is thrown

Returns

A ComplexNumber consisting of the quotient of the ComplexNumber and the real value

Definition at line 316 of file ComplexNumber.cpp.

5.1.4.64 template<class T > ComplexNumber< T > operator/ (const T & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Divides a real value by a ComplexNumber.

Divides a real value on the left side of the operand by a ComplexNumber on the right side of the operand, returning the resultant quotient as a third ComplexNumber.

$$z_2 = \frac{x}{z_1} = \frac{x}{a_1 + b_1 i} = \frac{a_1 x}{a_1^2 + b_1^2} - \frac{b_1 x}{a_1^2 + b_1^2} i$$

Parameters

in	*lhs*	The real value on the left side of the operand, by which the ComplexNumber on the right side of the operand will be divided
in	*rhs*	The ComplexNumber on the right side of the operand, which will divide the real value on the left side of the operand

Exceptions

DivisionByZero-Exception	if attempting to divide a value lhs by a ComplexNumber rhs that is equal to zero, a DivisionByZeroException is thrown

Returns

A ComplexNumber consisting of the quotient of the real value and the ComplexNumber

Definition at line 325 of file ComplexNumber.cpp.

5.1.4.65 template<class U > std::ostream& operator<< (std::ostream & *out,* const ComplexNumber< U > & *z*) [friend]

Inserts a ComplexNumber into the ostream.

Sends a representation of this ComplexNumber into the ostream

Parameters

in	*out*	The ostream reference into which the ComplexNumber will be inserted
in	*z*	The ComplexNumber from which the data will be obtained

Returns

A reference to the modified ostream

Definition at line 371 of file ComplexNumber.cpp.

5.1.4.66 template<class T , class U > bool operator== (const ComplexNumber< T > & *lhs,* const ComplexNumber< U > & *rhs*) [related]

Compares a ComplexNumber to another ComplexNumber.

Determines if the ComplexNumber on the left side of the operand is equal to the ComplexNumber on the right side of the operand

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand to which the ComplexNumber on the right will be compared
in	*rhs*	The ComplexNumber on the right side of the operand, which will be compared to the the ComplexNumber on the left side of the operand

Returns

A boolean indicating whether the ComplexNumber on the left side of the operand is equal to the ComplexNumber on the right side of the operand

Definition at line 339 of file ComplexNumber.cpp.

5.1.4.67 template<class T , class U > bool operator== (const ComplexNumber< T > & *lhs,* const U & *rhs*) [related]

Compares a ComplexNumber to a real value.

Determines if the ComplexNumber on the left side of the operand is equal to the real value on the right side of the operand

Parameters

in	*lhs*	The ComplexNumber on the left side of the operand to which the real value on the right will be compared
in	*rhs*	The real value on the right side of the operand, which will be compared to the the ComplexNumber on the left side of the operand

Returns

A boolean indicating whether the ComplexNumber on the left side of the operand is equal to the real value on the right side of the operand

Definition at line 345 of file ComplexNumber.cpp.

5.1.4.68 template<class T , class U > bool operator== (const T & *lhs,* const ComplexNumber< U > & *rhs*) [related]

Compares a real value to a ComplexNumber.

Determines if the real value on the left side of the operand is equal to the ComplexNumber on the right side of the operand

Parameters

in	*lhs*	The real value on the left side of the operand to which the ComplexNumber on the right will be compared
in	*rhs*	The ComplexNumber on the right side of the operand, which will be compared to the the real value on the left side of the operand

Returns

A boolean indicating whether the real value on the left side of the operand is equal to the ComplexNumber on the right side of the operand

Definition at line 350 of file ComplexNumber.cpp.

5.1.4.69 template<class U > std::istream& operator>> (std::istream & *in*, ComplexNumber< U > & *z*) [friend]

Extracts a ComplexNumber from the istream.

Initializes a ComplexNumber from values extracted from the istream

Parameters

in	*in*	The istream reference from which the ComplexNumber will be extracted
in	*z*	The ComplexNumber into which the data will be inserted

Returns

A reference to the modified istream

Definition at line 383 of file ComplexNumber.cpp.

5.1.4.70 template<class T > ComplexNumber< T > pow (const ComplexNumber< T > & *a*, const ComplexNumber< T > & *b*) [related]

Exponentiation of ComplexNumbers.

Computes the exponentiation of a ComplexNumber and another ComplexNumber.

$$a^b = e^{\ln a^b} = e^{b \ln a} = e^{b \ln|a| + i \arg a}$$

Parameters

in	*a*	The ComplexNumber that is the base of the exponentiation
in	*b*	The ComplexNumber that is the exponent of the exponentiation

Returns

A ComplexNumber that is the exponentiation of two ComplexNumbers

Definition at line 186 of file ComplexMath.cpp.

5.1.4.71 template<class T > ComplexNumber< T > pow (const ComplexNumber< T > & *a*, const T & *b*) [related]

Exponentiation of a ComplexNumber and real value.

Computes the exponentiation of a ComplexNumber and a real value.

$$a^b = e^{\ln a^b} = e^{b \ln a} = e^{b \ln|a| + i \arg a}$$

Parameters

in	*a*	The ComplexNumber that is the base of the exponentiation
in	*b*	The real value that is the exponent of the exponentiation

Returns

A ComplexNumber that is the exponentiation of a ComplexNumber and a real value

Definition at line 199 of file ComplexMath.cpp.

5.1.4.72 template<class T > ComplexNumber< T > pow (const T & *a*, const ComplexNumber< T > & *b*) [related]

Exponentiation of a real value and a ComplexNumber.

Computes the exponentiation of a real value and a ComplexNumber.

$$a^b = e^{\ln a^b} = e^{b \ln a} = e^{b \ln|a| + i \arg a}$$

Parameters

in	*a*	The real value that is the base of the exponentiation
in	*b*	The ComplexNumber that is the exponent of the exponentiation

Returns

A ComplexNumber that is the exponentiation of a real value and a ComplexNumber

Definition at line 208 of file ComplexMath.cpp.

5.1.4.73 template<class T > ComplexNumber< T > sec (const ComplexNumber< T > & z) [related]

Secant of a ComplexNumber.

Computes the complex secant of a ComplexNumber.

$$\sec z = \frac{1}{\cos z}$$

Parameters

in	z	The ComplexNumber of which to compute the secant

Returns

A ComplexNumber that is the secant of a ComplexNumber

Definition at line 324 of file ComplexMath.cpp.

5.1.4.74 template<class T > ComplexNumber< T > sech (const ComplexNumber< T > & z) [related]

Hyperbolic secant of a ComplexNumber.

Computes the complex hyperbolic secant of a ComplexNumber.

$$\operatorname{sech} z = \frac{1}{\cosh z}$$

Parameters

in	z	The ComplexNumber of which to compute the hyperbolic secant

Returns

A ComplexNumber that is the hyperbolic secant of a ComplexNumber

Definition at line 469 of file ComplexMath.cpp.

5.1.4.75 template<class T > ComplexNumber< T > sin (const ComplexNumber< T > & z) [related]

Sine of a ComplexNumber.

Computes the complex sine of a ComplexNumber.

$$\sin z = \sin(a+bi) = \sin a \cos(bi) + \cos a \sin(bi) = \sin a \cosh b + i \cos a \sinh b$$

This is a result of the addition or subtraction theorems or formulae, which can be proven by Euler's formula, $e^{ix} = \cos x + i \sin x$, and by noting the symmetry of the sine and cosine functions as follows,

$$\begin{aligned}\cos(\alpha \pm \beta) + i \sin(\alpha \pm \beta) &= e^{i(\alpha \pm \beta)} \\ &= e^{i\alpha} e^{\pm i\beta} \\ &= (\cos\alpha + i \sin\alpha)(\cos\beta \pm i \sin\beta) \\ &= (\cos\alpha\cos\beta \mp \sin\alpha\sin\beta) + i(\sin\alpha\cos\beta \pm \cos\alpha\sin\beta)\end{aligned}$$

Thus,

$$\begin{aligned}\sin(\alpha \pm \beta) &= \sin\alpha\cos\beta \pm \cos\alpha\sin\beta \\ \cos(\alpha \pm \beta) &= \cos\alpha\cos\beta \mp \sin\alpha\sin\beta\end{aligned}$$

Since, by the exponential definitions of the trigonometric functions derived from Euler's formula and the definitions of the hyperbolic functions,

$$\begin{aligned}\sin x &= \frac{e^{ix} - e^{-ix}}{2i} \\ \cos x &= \frac{e^{ix} + e^{-ix}}{2} \\ \sinh x &= \frac{e^{x} - e^{-x}}{2} \\ \cosh x &= \frac{e^{x} + e^{-x}}{2}\end{aligned}$$

Then,

$$\sin(xi) = \frac{e^{i^2 x} - e^{-i^2 x}}{2i} = \frac{e^{-x} - e^{x}}{2i} = \frac{e^{x} - e^{-x}}{2} i = i \sinh x$$
$$\cos(xi) = \frac{e^{i^2 x} + e^{-i^2 x}}{2} = \frac{e^{-x} + e^{x}}{2} = \cosh x$$

Parameters

in	*z*	The ComplexNumber of which to compute the sine

Returns

A ComplexNumber that is the sine of a ComplexNumber

Definition at line 293 of file ComplexMath.cpp.

5.1.4.76 template<class T > ComplexNumber< T > sinh (const ComplexNumber< T > & *z*) [related]

Hyperbolic sine of a ComplexNumber.

Computes the complex hyperbolic sine of a ComplexNumber.

$$\sinh z = \sinh(a + bi) = \sinh a \cosh(bi) + \cosh a \sinh(bi) = \sinh a \cos b + i \cosh a \sin b$$

This is a result of the addition or subtraction theorems or formulae,

$$\sinh(x \pm y) = \sinh x \cosh y \pm \cosh x \sinh y$$
$$\cosh(x \pm y) = \cosh x \cosh y \pm \sinh x \sinh y$$

which can be proven as follows, also noting the symmetry of the hyperbolic sine and hyperbolic cosine functions,

$$\begin{aligned}\sinh(x+y) =& \frac{e^{x+y}-e^{-x-y}}{2}\\ =& \frac{e^x e^y - e^x e^{-y} + e^x e^{-y} - e^{-x}e^{-y}}{2}\\ =& e^x\left(\frac{e^y-e^{-y}}{2}\right) + e^{-y}\left(\frac{e^x-e^{-x}}{2}\right)\\ =& (\cosh x+\sinh x)\sinh y + (\cosh y - \sinh y)\sinh x\\ =& \cosh x\sinh y + \sinh x\sinh y + \sinh x\cosh y - \sinh x\sinh y\\ =& \sinh x\cosh y + \cosh x\sinh y\end{aligned}$$

$$\begin{aligned}\cosh(x+y) =& \frac{e^{x+y}+e^{-x-y}}{2}\\ =& \frac{e^x e^y - e^x e^{-y} + e^x e^{-y} + e^{-x}e^{-y}}{2}\\ =& e^x\left(\frac{e^y-e^{-y}}{2}\right) + e^{-y}\left(\frac{e^x+e^{-x}}{2}\right)\\ =& (\cosh x+\sinh x)\sinh y + (\cosh y - \sinh y)\cosh x\\ =& \cosh x\sinh y + \sinh x\sinh y + \cosh x\cosh y - \cosh x\sinh y\\ =& \cosh x\cosh y + \sinh x\sinh y\end{aligned}$$

Euler's formula,

$$e^{ix} = \cos x + i\sin x$$

Since, by the exponential definitions of the trigonometric functions derived from Euler's formula and the definitions of the hyperbolic functions,

$$\begin{aligned}\sin x &= \frac{e^{ix}-e^{-ix}}{2i}\\ \cos x &= \frac{e^{ix}+e^{-ix}}{2}\\ \sinh x &= \frac{e^{x}-e^{-x}}{2}\\ \cosh x &= \frac{e^{x}+e^{-x}}{2}\end{aligned}$$

Then,

$$\sinh(xi) = \frac{e^{xi} - e^{-xi}}{2} = i\sin x$$
$$\cosh(xi) = \frac{e^{xi} + e^{-xi}}{2} = \cos x$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic sine

Returns

A ComplexNumber that is the hyperbolic sine of a ComplexNumber

Definition at line 438 of file ComplexMath.cpp.

5.1.4.77 template<class T > ComplexNumber< T > sqrt (const ComplexNumber< T > & *z*) [related]

Square root of a ComplexNumber.

Computes the principle square root of a ComplexNumber.

If $z = re^{i\varphi}$, where $r = |z|$ and $\varphi = \arg z$, then

$$\sqrt{z} = \sqrt{r}e^{i\frac{\varphi}{2}} = \left(a^2 + b^2\right)^{\frac{1}{4}} \left(\cos\left(\frac{\varphi}{2}\right) + i\sin\left(\frac{\varphi}{2}\right)\right)$$

This is a result of de Moivre's formula,

$$(\cos x + i\sin x)^n = \cos nx + i\sin nx$$

which can be proven by Euler's formula,

$$e^{ix} = \cos x + i\sin x$$

and the rules of exponentiation,

$$(e^{ix})^n = e^{inx}$$

So,

$$e^{i(nx)} = (\cos x + i\sin x)^n = \cos nx + i\sin nx$$

Parameters

in	*z*	The ComplexNumber of which to compute the square roots

Returns

A ComplexNumber that is the square root of a ComplexNumber

Definition at line 221 of file ComplexMath.cpp.

5.1.4.78 template<class T > ComplexNumber< T > sqrt (const T & *x*) [related]

Square root of a real value.

Computes the principle square root of a real value.

If $x < 0$, the solutions are both real numbers and can be computed as usual.

If $x < 0$, since $i^2 = -1$, we can factor i out of the radical and compute the square root of x as usual, resulting in imaginary solutions.

Parameters

in	*x*	The real value of which to compute the square root

Returns

A ComplexNumber that is the square root of a real value

Definition at line 232 of file ComplexMath.cpp.

5.1.4.79 template<class T > ComplexNumber< T > tan (const ComplexNumber< T > & *z*) [related]

Tangent of a ComplexNumber.

Computes the complex tangent of a ComplexNumber.

$$\tan z = \tan(a+bi) = \frac{\sin 2a}{\cos 2a + \cosh 2b} + \frac{\sinh 2b}{\cos 2a + \cosh 2a} i$$

This is a result of the exponential definitions for the trigonometric functions derived from Euler's formula and the definitions of the hyperbolic functions,

$$\sin x = \frac{e^{ix} - e^{-ix}}{2i}$$
$$\cos x = \frac{e^{ix} + e^{-ix}}{2}$$
$$\tan x = \frac{\sin x}{\cos x} = -i\frac{e^{ix} - e^{-ix}}{e^{ix} + e^{-ix}}$$
$$\sinh x = \frac{e^{x} - e^{-x}}{2}$$
$$\cosh x = \frac{e^{x} + e^{-x}}{2}$$

Euler's formula,

$$e^{ix} = \cos x + i\sin x$$

and the double angle formulae for sine and cosine, which can be proven by either the multiple-angle formulae or the angle sum and difference identities,

$$\sin 2\theta = 2\sin\theta\cos\theta$$
$$\cos 2\theta = \cos^2\theta - \sin^2\theta$$

So,

$$\begin{aligned}
\tan z &= \tan(a+bi) \\
&= -i\frac{e^{i(a+bi)} - e^{-i(a+bi)}}{e^{i(a+bi)} + e^{-i(a+bi)}} \\
&= -i\frac{e^{ai-b} - e^{b-ai}}{e^{ai-b} + e^{b-ai}} \\
&= -i\frac{e^{ai}e^{-b} - e^{b}e^{-ai}}{e^{ai}e^{-b} + e^{b}e^{-ai}} \\
&= -i\frac{e^{-b}(\cos a + i\sin a) - e^{b}(\cos a - i\sin a)}{e^{-b}(\cos a + i\sin a) + e^{b}(\cos a - i\sin a)} \\
&= \frac{(e^{-b}\sin a + e^{b}\sin a) + (e^{b}\cos a + e^{-b}\cos a)i}{(e^{-b}\cos a + e^{b}\cos a) + (e^{-b}\sin a - e^{b}\sin a)i}
\end{aligned}$$

Multiplying the numerator and denominator by the complex conjugate of the denominator,

$$\frac{(e^{-b}\sin a+e^{b}\sin a)(e^{-b}\cos a+e^{b}\cos a)+(e^{b}\cos a-e^{-b}\cos a)(e^{-b}\sin a-e^{b}\sin a)}{(e^{-b}\cos a+e^{b}\cos a)^2+(e^{-b}\sin a-e^{b}\sin a)^2}+$$
$$\frac{(e^{b}\cos a-e^{-b}\cos a)(e^{-b}\cos a+e^{b}\cos a)-(e^{-b}\sin a+e^{b}\sin a)(e^{-b}\sin a-e^{b}\sin a)}{(e^{-b}\cos a+e^{b}\cos a)^2+(e^{-b}\sin a-e^{b}\sin a)^2}i=$$
$$\frac{(4\sin a\cos a)+(e^{2b}\cos^2 a-e^{-2b}\cos^2 a+e^{2b}\sin^2 a-e^{-2b}\sin^2 a)i}{e^{2b}\cos^2 a+e^{-2b}\cos^2 a+2\cos^2 a+e^{2b}\sin^2 a+e^{-2b}\sin^2 a-2\sin^2 a}=$$
$$\frac{(4\sin a\cos a)+(e^{2b}-e^{-2b})(\cos^2 a+\sin^2 a)i}{(e^{2b}+e^{-2b})(\cos^2 a+\sin^2 a)+2(\cos^2 a-\sin^2 a)}=$$
$$\frac{2\left(2\sin a\cos a+\frac{e^{2b}-e^{-2b}}{2}i\right)}{2\left(\frac{e^{2b}+e^{-2b}}{2}+\cos^2 a-\sin^2 a\right)}=$$
$$\frac{\sin 2a}{\cos 2a+\cosh 2b}+\frac{\sinh 2b}{\cos 2a+\cosh 2b}i$$

Parameters

in	*z*	The ComplexNumber of which to compute the tangent

Returns

A ComplexNumber that is the tangent of a ComplexNumber

Definition at line 307 of file ComplexMath.cpp.

5.1.4.80 template<class T > ComplexNumber< T > tanh (const ComplexNumber< T > & *z*) [related]

Hyperbolic tangent of a ComplexNumber.

Computes the complex hyperbolic tangent of a ComplexNumber.

$$\tanh z=\tanh(a+bi)=\frac{\sinh 2a}{\cosh 2a+\cos 2b}+\frac{\sin 2b}{\cosh 2a+\cos 2b}i$$

This is a result of the definitions for the hyperbolic functions,

$$\sinh x = \frac{e^x - e^{-x}}{2}$$
$$\cosh x = \frac{e^x + e^{-x}}{2}$$
$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Euler's formula,

$$e^{ix} = \cos x + i \sin x$$

and the double angle formulae for sine and cosine, which can be proven by either the multiple-angle formulae or the angle sum and difference identities,

$$\sin 2\theta = 2\sin\theta\cos\theta$$
$$\cos 2\theta = \cos^2\theta - \sin^2\theta$$

So,

$$\begin{aligned}
\tanh z =& \tanh(a+bi) \\
=& \frac{e^{a+bi} - e^{-a-bi}}{e^{a+bi} + e^{-a-bi}} \\
=& \frac{e^a e^{bi} - e^{-a} e^{-bi}}{e^a e^{bi} + e^{-a} e^{-bi}} \\
=& \frac{e^a(\cos b + i\sin b) - e^{-a}(\cos b - i\sin b)}{e^a(\cos b + i\sin b) + e^{-a}(\cos b - i\sin b)} \\
=& \frac{(e^a\cos b - e^{-a}\cos b) + (e^a\sin b + e^{-a}\sin b)i}{(e^a\cos b + e^{-a}\cos b) + (e^a\sin b - e^{-a}\sin b)i}
\end{aligned}$$

Multiplying the numerator and denominator by the complex conjugate of the denominator,

$$\frac{(e^a\cos b - e^{-a}\cos b)(e^a\cos b + e^{-a}\cos b) + (e^a\sin b + e^{-a}\sin b)(e^a\sin b - e^{-a}\sin b)}{(e^a\cos b + e^{-a}\cos b)^2 + (e^a\sin b - e^{-a}\sin b)^2} +$$
$$\frac{(e^a\sin b + e^{-a}\sin b)(e^a\cos b + e^{-a}\cos b) - (e^a\cos b - e^{-a}\cos b)(e^a\sin b - e^{-a}\sin b)}{(e^a\cos b + e^{-a}\cos b)^2 + (e^a\sin b - e^{-a}\sin b)^2} i =$$
$$\frac{(e^{2a}\cos^2 b - e^{-2a}\cos^2 b + e^{2a}\sin^2 b - e^{-2a}\sin^2 b) + (4\sin b\cos b)i}{e^{2a}\cos^2 b + e^{-2a}\cos^2 b + 2\cos^2 b + e^{2a}\sin^2 b + e^{-2a}\sin^2 b - 2\sin^2 b} =$$
$$\frac{(e^{2a} - e^{-2a})(\cos^2 b + \sin^2 b) + (4\sin b\cos b)i}{(e^{2a} + e^{-2a})(\cos^2 b + \sin^2 b) + 2(\cos^2 b - \sin^2 b)} =$$
$$\frac{2\left(\frac{e^{2a}-e^{-2a}}{2} + 2i\sin b\cos b\right)}{2\left(\frac{e^{2a}+e^{-2a}}{2} + \cos^2 b - \sin^2 b\right)} =$$
$$\frac{\sinh 2a}{\cosh 2a + \cos 2b} + \frac{\sin 2b}{\cosh 2a + \cos 2b} i$$

Parameters

in	*z*	The ComplexNumber of which to compute the hyperbolic tangent

Returns

A ComplexNumber that is the hyperbolic tangent of a ComplexNumber

Definition at line 452 of file ComplexMath.cpp.

5.1.5 Member Data Documentation

5.1.5.1 const ComplexNumber<double> ComplexNumber::i [static]

A public constant representing the value i.

This constant value is the imaginary unit, i, defined as $i^2 = -1$.

Definition at line 495 of file ComplexNumber.h.

5.1.5.2 T ComplexNumber::imaginary [private]

The value of the non-real part of the complex number.

This value is the non-real part of the complex number in rectangular form

Definition at line 517 of file ComplexNumber.h.

5.1.5.3 T ComplexNumber::real [private]

The value of the real part of the complex number.

This value is the real part of the complex number in rectangular form

Definition at line 509 of file ComplexNumber.h.

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

- include/ComplexNumber.h
- include/ComplexMath.h
- src/ComplexNumber.cpp

5.2 DivisionByZeroException Class Reference

A division by zero exception.

`#include <MathExceptions.h>`

Private Member Functions

- const char * what () const throw ()

5.2.1 Detailed Description

A division by zero exception.

"include/MathExceptions.h" This class represents a division by zero exception

5.2.2 Member Function Documentation

5.2.2.1 const char* DivisionByZeroException::what ( ) const throw () [inline, private]

Definition at line 75 of file MathExceptions.h.

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

- include/MathExceptions.h

5.3 LogarithmOfZeroException Class Reference

A logarithm of zero exception.

`#include <MathExceptions.h>`

Private Member Functions

- const char * what () const throw ()

5.3.1 Detailed Description

A logarithm of zero exception.

"include/MathExceptions.h" This class represents a logarithm of zero exception

5.3.2 Member Function Documentation

5.3.2.1 const char* LogarithmOfZeroException::what ( ) const throw () [inline, private]

Definition at line 88 of file MathExceptions.h.

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

- include/MathExceptions.h

5.4 ZeroToComplexPowerException Class Reference

A zero to a complex power exception.

`#include <MathExceptions.h>`

Private Member Functions

- const char * what () const throw ()

5.4.1 Detailed Description

A zero to a complex power exception.

"include/MathExceptions.h" This class represents a zero to a complex power exception

5.4.2 Member Function Documentation

5.4.2.1 const char* ZeroToComplexPowerException::what ( ) const throw () [inline, private]

Definition at line 101 of file MathExceptions.h.

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

- include/MathExceptions.h

5.5 ZeroToTheZerothPowerException Class Reference

A zero to the zeroth power exception.

`#include <MathExceptions.h>`

Private Member Functions

- const char * what () const throw ()

5.5.1 Detailed Description

A zero to the zeroth power exception.

"include/MathExceptions.h" This class represents a zero to the zeroth power exception

5.5.2 Member Function Documentation

5.5.2.1 const char* ZeroToTheZerothPowerException::what ( ) const throw () [inline, private]

Definition at line 114 of file MathExceptions.h.

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

- include/MathExceptions.h

Chapter 6

File Documentation

6.1 include/ComplexMath.h File Reference

Elementary complex mathematical functions.

```
#include "ComplexNumber.h" #include "Namespace.h" #include
<vector>
```

Namespaces

- namespace math

  Math namespace.
- namespace complex

  Complex namespace.

Defines

- #define M_El 2.71828182845904523536028747135266 25L

  long double e
- #define M_LOG2El 1.44269504088896340735992468100 18921L

  long double $\log_2 e$
- #define M_LOG10El 0.43429448190325182765112891891 66051L

  long double $\log_{10} e$
- #define M_LN2l 0.69314718055994530941723212145 81766L

  long double $\ln 2$
- #define M_LN10l 2.30258509299404568401799145468 43642L

  long double $\ln 10$

- #define M_PIl 3.14159265358979323846264338327950 29L

  long double π
- #define M_PI_2l 1.57079632679489661923132169163975 14L

  long double $\frac{\pi}{2}$
- #define M_PI_4l 0.78539816339744830961566084581987 57L

  long double $\frac{\pi}{4}$
- #define M_1_PIl 0.31830988618379067153776752674502 87L

  long double $\frac{1}{\pi}$
- #define M_2_PIl 0.63661977236758134307553505349005 74L

  long double $\frac{2}{\pi}$
- #define M_2_SQRTPIl 1.12837916709551257389615890312154 52L

  long double $\frac{1}{\sqrt{\pi}}$
- #define M_SQRT2l 1.41421356237309504880168872420969 81L

  long double $\sqrt{2}$
- #define M_SQRT1_2l 0.70710678118654752440084436210484 90L

  long double $\frac{1}{\sqrt{2}}$

6.1.1 Detailed Description

Elementary complex mathematical functions.

Author

Matthew Krupcale

Version

0.1.2

Date

2010

This file contains the definitions or function prototypes for elementary complex mathematical functions. It also contains several mathematical constants that are used internally

Definition in file ComplexMath.h.

6.1.2 Define Documentation

6.1.2.1 #define M_1_PIl 0.318309886183790671537767526745028 7L

long double $\frac{1}{\pi}$

Constant in long-double format using 128-bit IEEE quad.

$\frac{1}{\pi}$

Definition at line 162 of file ComplexMath.h.

6.1.2.2 #define M_2_PIl 0.636619772367581343075535053490057 4L

long double $\frac{2}{\pi}$

Constant in long-double format using 128-bit IEEE quad.

$\frac{2}{\pi}$

Definition at line 174 of file ComplexMath.h.

6.1.2.3 #define M_2_SQRTPIl 1.128379167095512573896158903121545 2L

long double $\frac{1}{\sqrt{\pi}}$

Constant in long-double format using 128-bit IEEE quad.

$\frac{1}{\sqrt{\pi}}$

Definition at line 186 of file ComplexMath.h.

6.1.2.4 #define M_El 2.718281828459045235360287471352662 5L

long double e

Constant in long-double format using 128-bit IEEE quad.

Euler's number, e, which can be represented in several ways, one of which follows,

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$$

Definition at line 61 of file ComplexMath.h.

6.1.2.5 #define M_LN10l 2.302585092994045684017991454684364 2L

long double $\ln 10$

Constant in long-double format using 128-bit IEEE quad.

Natural logarithm of 10

Definition at line 109 of file ComplexMath.h.

6.1.2.6 #define M_LN2l 0.693147180559945309417232121458176 6L

long double $\ln 2$

Constant in long-double format using 128-bit IEEE quad.

Natural logarithm of 2

Definition at line 97 of file ComplexMath.h.

6.1.2.7 #define M_LOG10El 0.434294481903251827651128918916605 1L

long double $\log_{10} e$

Constant in long-double format using 128-bit IEEE quad.

Logarithm base 10 of e

Definition at line 85 of file ComplexMath.h.

6.1.2.8 #define M_LOG2El 1.442695040888963407359924681001892 1L

long double $\log_2 e$

Constant in long-double format using 128-bit IEEE quad.

Logarithm base 2 of e

Definition at line 73 of file ComplexMath.h.

6.1.2.9 #define M_PI_2l 1.570796326794896619231321691639751 4L

long double $\frac{\pi}{2}$

Constant in long-double format using 128-bit IEEE quad.

$\frac{\pi}{2}$

Definition at line 138 of file ComplexMath.h.

6.1.2.10 #define M_PI_4l 0.785398163397448309615660845819875 7L

long double $\frac{\pi}{4}$

Constant in long-double format using 128-bit IEEE quad.

$\frac{\pi}{4}$

Definition at line 150 of file ComplexMath.h.

6.1.2.11 #define M_PIl 3.14159265358979323846264338327950291

long double π

Constant in long-double format using 128-bit IEEE quad.

π, which is defined as the ratio of a circle's circumference to its diameter,

$$\pi = \frac{C}{d}$$

Definition at line 126 of file ComplexMath.h.

6.1.2.12 #define M_SQRT1_2l 0.70710678118654752440084436210484901

long double $\frac{1}{\sqrt{2}}$

Constant in long-double format using 128-bit IEEE quad.

$\frac{1}{\sqrt{2}}$

Definition at line 210 of file ComplexMath.h.

6.1.2.13 #define M_SQRT2l 1.41421356237309504880168872420969811

long double $\sqrt{2}$

Constant in long-double format using 128-bit IEEE quad.

$\sqrt{2}$

Definition at line 198 of file ComplexMath.h.

6.2 include/ComplexNumber.h File Reference

Representation of a complex number.

```
#include "Namespace.h" #include <iosfwd>
```

Classes

- class ComplexNumber

  A class template which supports complex numbers.

Namespaces

- namespace math

  Math namespace.
- namespace complex

  Complex namespace.

6.2.1 Detailed Description

Representation of a complex number.

Author

Matthew Krupcale

Version

0.1.2

Date

2010

This file contains the class template declaration that represents a complex number, its function prototypes as well as several declarations for overloaded operators

Definition in file ComplexNumber.h.

6.3 include/MathExceptions.h File Reference

Classes that represent mathematical exceptions.

```
#include "Namespace.h" #include <exception>
```

Classes

- class DivisionByZeroException

  A division by zero exception.
- class LogarithmOfZeroException

  A logarithm of zero exception.
- class ZeroToComplexPowerException

  A zero to a complex power exception.
- class ZeroToTheZerothPowerException

  A zero to the zeroth power exception.

Namespaces

- namespace math

  Math namespace.
- namespace complex

  Complex namespace.

6.3.1 Detailed Description

Classes that represent mathematical exceptions.

Author

Matthew Krupcale

Version

0.1.0

Date

2010

This file contains classes that represent mathematical exceptions, such as domain errors

Definition in file MathExceptions.h.

6.4 include/Namespace.h File Reference

Defines namesapce macros.

Defines

- #define BEGIN_NAMESPACE(X) namespace X {

  Begin a namespace.
- #define END_NAMESPACE(X) }

  End a namespace.

6.4.1 Detailed Description

Defines namesapce macros.

Author

Matthew Krupcale

Version

0.1.0

Date

2010

This file defines macros that can be used to begin an end namespace

Definition in file Namespace.h.

6.4.2 Define Documentation

6.4.2.1 #define BEGIN_NAMESPACE(*X*) namespace X {

Begin a namespace.

This macro defines a new namespace with the name *X*

Definition at line 52 of file Namespace.h.

6.4.2.2 #define END_NAMESPACE(*X*) }

End a namespace.

This macro defines the end of a namespace with the name *X*

Definition at line 62 of file Namespace.h.

6.5 src/ComplexMath.cpp File Reference

Elementary complex mathematical functions.

```
#include "ComplexNumber.h" #include "ComplexMath.h" #include
"MathExceptions.h" #include "Namespace.h" #include <cmath> ×
#include <vector>
```

Namespaces

- namespace math

  Math namespace.
- namespace complex

  Complex namespace.

Defines

- #define COMPLEX_MATH_CPP

Functions

- template<class T >
  T abs (const ComplexNumber< T > &z)
- template<class T >
  T abs2 (const ComplexNumber< T > &z)
- template<class T >
  T arg (const ComplexNumber< T > &z)
- template<class T >
  T logabs (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > conjugate (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > exp (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > inverse (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > log (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > log (const T &x)
- template<class T >
  ComplexNumber< T > log10 (const ComplexNumber< T > &z)

- template<class T >
  ComplexNumber< T > log10 (const T &x)
- template<class T >
  ComplexNumber< T > logb (const ComplexNumber< T > &z, const ComplexNumber< T > &b)
- template<class T >
  ComplexNumber< T > logb (const ComplexNumber< T > &z, const T &b)
- template<class T >
  ComplexNumber< T > logb (const T &x, const ComplexNumber< T > &b)
- template<class T >
  ComplexNumber< T > logb (const T &x, const T &b)
- template<class T >
  ComplexNumber< T > pow (const ComplexNumber< T > &a, const ComplexNumber< T > &b)
- template<class T >
  ComplexNumber< T > pow (const ComplexNumber< T > &a, const T &b)
- template<class T >
  ComplexNumber< T > pow (const T &a, const ComplexNumber< T > &b)
- template<class T >
  ComplexNumber< T > sqrt (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > sqrt (const T &x)
- template<class T >
  ComplexNumber< T > cbrt (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > nthRoot (const ComplexNumber< T > &z, const int n)
- template<class T >
  std::vector< ComplexNumber< T > > nthRoots (const ComplexNumber< T > &z, const int n)
- template<class T >
  void nthRoots (const ComplexNumber< T > &z, std::vector< ComplexNumber< T > > &roots, const int n)
- template<class T >
  ComplexNumber< T > sin (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > cos (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > tan (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > csc (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > sec (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > cot (const ComplexNumber< T > &z)

- template<class T >
  ComplexNumber< T > asin (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > asin (const T &x)
- template<class T >
  ComplexNumber< T > acos (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > acos (const T &x)
- template<class T >
  ComplexNumber< T > atan (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > atan (const T &x)
- template<class T >
  ComplexNumber< T > acsc (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > acsc (const T &x)
- template<class T >
  ComplexNumber< T > asec (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > asec (const T &x)
- template<class T >
  ComplexNumber< T > acot (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > acot (const T &x)
- template<class T >
  ComplexNumber< T > sinh (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > cosh (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > tanh (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > csch (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > sech (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > coth (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > asinh (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > asinh (const T &x)
- template<class T >
  ComplexNumber< T > acosh (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > acosh (const T &x)

- template<class T >
  ComplexNumber< T > atanh (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > atanh (const T &x)
- template<class T >
  ComplexNumber< T > acsch (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > acsch (const T &x)
- template<class T >
  ComplexNumber< T > asech (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > asech (const T &x)
- template<class T >
  ComplexNumber< T > acoth (const ComplexNumber< T > &z)
- template<class T >
  ComplexNumber< T > acoth (const T &x)

6.5.1 Detailed Description

Elementary complex mathematical functions.

Author

Matthew Krupcale

Version

0.1.2

Date

2010

This file contains the implementations for elementary complex mathematical functions

Definition in file ComplexMath.cpp.

6.5.2 Define Documentation

6.5.2.1 #define COMPLEX_MATH_CPP

Definition at line 25 of file ComplexMath.cpp.

6.5.3 Function Documentation

6.5.3.1 template<class T > T abs (const ComplexNumber< T > & z) [related]

Definition at line 73 of file ComplexMath.cpp.

6.5.3.2 template<class T > T abs2 (const ComplexNumber< T > & z) [related]

Definition at line 80 of file ComplexMath.cpp.

6.5.3.3 template<class T > ComplexNumber<T> acos (const ComplexNumber< T > & z) [related]

Definition at line 361 of file ComplexMath.cpp.

6.5.3.4 template<class T > ComplexNumber<T> acos (const T & x) [related]

Definition at line 371 of file ComplexMath.cpp.

6.5.3.5 template<class T > ComplexNumber<T> acosh (const ComplexNumber< T > & z) [related]

Definition at line 497 of file ComplexMath.cpp.

6.5.3.6 template<class T > ComplexNumber<T> acosh (const T & x) [related]

Definition at line 507 of file ComplexMath.cpp.

6.5.3.7 template<class T > ComplexNumber<T> acot (const ComplexNumber< T > & z) [related]

Definition at line 426 of file ComplexMath.cpp.

6.5.3.8 template<class T > ComplexNumber<T> acot (const T & x) [related]

Definition at line 432 of file ComplexMath.cpp.

6.5.3.9 template<class T > ComplexNumber<T> acoth (const ComplexNumber< T > & z) [related]

Definition at line 569 of file ComplexMath.cpp.

6.5.3.10 template<class T > ComplexNumber<T> acoth (const T & x) [related]

Definition at line 575 of file ComplexMath.cpp.

6.5.3.11 template<class T > ComplexNumber<T> acsc (const ComplexNumber< T > & z) [related]

Definition at line 402 of file ComplexMath.cpp.

6.5.3.12 template<class T > ComplexNumber<T> acsc (const T & x) [related]

Definition at line 408 of file ComplexMath.cpp.

6.5.3.13 template<class T > ComplexNumber<T> acsch (const ComplexNumber< T > & z) [related]

Definition at line 541 of file ComplexMath.cpp.

6.5.3.14 template<class T > ComplexNumber<T> acsch (const T & x) [related]

Definition at line 547 of file ComplexMath.cpp.

6.5.3.15 template<class T > T arg (const ComplexNumber< T > & z) [related]

Definition at line 87 of file ComplexMath.cpp.

6.5.3.16 template<class T > ComplexNumber<T> asec (const ComplexNumber< T > & z) [related]

Definition at line 414 of file ComplexMath.cpp.

6.5.3.17 template<class T > ComplexNumber<T> asec (const T & x) [related]

Definition at line 420 of file ComplexMath.cpp.

6.5.3.18 template<class T > ComplexNumber<T> asech (const ComplexNumber< T > & z) [related]

Definition at line 553 of file ComplexMath.cpp.

6.5.3.19 template<class T > ComplexNumber<T> asech (const T & x) [related]

Definition at line 559 of file ComplexMath.cpp.

6.5.3.20 template<class T > ComplexNumber<T> asin (const ComplexNumber< T > & z) [related]

Definition at line 336 of file ComplexMath.cpp.

6.5.3.21 template<class T > ComplexNumber<T> asin (const T & x) [related]

Definition at line 347 of file ComplexMath.cpp.

6.5.3.22 template<class T > ComplexNumber<T> asinh (const ComplexNumber< T > & z) [related]

Definition at line 481 of file ComplexMath.cpp.

6.5.3.23 template<class T > ComplexNumber<T> asinh (const T & x) [related]

Definition at line 491 of file ComplexMath.cpp.

6.5.3.24 template<class T > ComplexNumber<T> atan (const ComplexNumber< T > & z) [related]

Definition at line 385 of file ComplexMath.cpp.

6.5.3.25 template<class T > ComplexNumber<T> atan (const T & x) [related]

Definition at line 396 of file ComplexMath.cpp.

6.5.3.26 template<class T > ComplexNumber<T> atanh (const ComplexNumber< T > & z) [related]

Definition at line 521 of file ComplexMath.cpp.

6.5.3.27 template<class T > ComplexNumber<T> atanh (const T & *x*) [related]

Definition at line 531 of file ComplexMath.cpp.

6.5.3.28 template<class T > ComplexNumber<T> cbrt (const ComplexNumber< T > & *z*) [related]

Definition at line 242 of file ComplexMath.cpp.

6.5.3.29 template<class T > ComplexNumber<T> conjugate (const ComplexNumber< T > & *z*) [related]

Definition at line 105 of file ComplexMath.cpp.

6.5.3.30 template<class T > ComplexNumber<T> cos (const ComplexNumber< T > & *z*) [related]

Definition at line 300 of file ComplexMath.cpp.

6.5.3.31 template<class T > ComplexNumber<T> cosh (const ComplexNumber< T > & *z*) [related]

Definition at line 445 of file ComplexMath.cpp.

6.5.3.32 template<class T > ComplexNumber<T> cot (const ComplexNumber< T > & *z*) [related]

Definition at line 330 of file ComplexMath.cpp.

6.5.3.33 template<class T > ComplexNumber<T> coth (const ComplexNumber< T > & *z*) [related]

Definition at line 475 of file ComplexMath.cpp.

6.5.3.34 template<class T > ComplexNumber<T> csc (const ComplexNumber< T > & *z*) [related]

Definition at line 318 of file ComplexMath.cpp.

6.5.3.35 template<class T > ComplexNumber<T> csch (const ComplexNumber< T > & z) [related]

Definition at line 463 of file ComplexMath.cpp.

6.5.3.36 template<class T > ComplexNumber<T> exp (const ComplexNumber< T > & z) [related]

Definition at line 111 of file ComplexMath.cpp.

6.5.3.37 template<class T > ComplexNumber<T> inverse (const ComplexNumber< T > & z) [related]

Definition at line 118 of file ComplexMath.cpp.

6.5.3.38 template<class T > ComplexNumber<T> log (const ComplexNumber< T > & z) [related]

Definition at line 128 of file ComplexMath.cpp.

6.5.3.39 template<class T > ComplexNumber<T> log (const T & x) [related]

Definition at line 137 of file ComplexMath.cpp.

6.5.3.40 template<class T > ComplexNumber<T> log10 (const ComplexNumber< T > & z) [related]

Definition at line 146 of file ComplexMath.cpp.

6.5.3.41 template<class T > ComplexNumber<T> log10 (const T & x) [related]

Definition at line 152 of file ComplexMath.cpp.

6.5.3.42 template<class T > T logabs (const ComplexNumber< T > & z) [related]

Definition at line 96 of file ComplexMath.cpp.

6.5.3.43 template<class T > ComplexNumber<T> logb (const ComplexNumber< T > & *z*, const ComplexNumber< T > & *b*) [related]

Definition at line 162 of file ComplexMath.cpp.

6.5.3.44 template<class T > ComplexNumber<T> logb (const ComplexNumber< T > & *z*, const T & *b*) [related]

Definition at line 168 of file ComplexMath.cpp.

6.5.3.45 template<class T > ComplexNumber<T> logb (const T & *x*, const ComplexNumber< T > & *b*) [related]

Definition at line 174 of file ComplexMath.cpp.

6.5.3.46 template<class T > ComplexNumber<T> logb (const T & *x*, const T & *b*) [related]

Definition at line 180 of file ComplexMath.cpp.

6.5.3.47 template<class T > ComplexNumber<T> nthRoot (const ComplexNumber< T > & *z*, const int *n*) [related]

Definition at line 253 of file ComplexMath.cpp.

6.5.3.48 template<class T > std::vector<ComplexNumber<T> > nthRoots (const ComplexNumber< T > & *z*, const int *n*) [related]

Definition at line 267 of file ComplexMath.cpp.

6.5.3.49 template<class T > void nthRoots (const ComplexNumber< T > & *z*, std::vector< ComplexNumber< T > > & *roots*, const int *n*)

Definition at line 275 of file ComplexMath.cpp.

6.5.3.50 template<class T > ComplexNumber<T> pow (const ComplexNumber< T > & *a*, const ComplexNumber< T > & *b*) [related]

Definition at line 186 of file ComplexMath.cpp.

6.5.3.51 template<class T > ComplexNumber<T> pow (const ComplexNumber< T > & *a,* const T & *b*) [related]

Definition at line 199 of file ComplexMath.cpp.

6.5.3.52 template<class T > ComplexNumber<T> pow (const T & *a,* const ComplexNumber< T > & *b*) [related]

Definition at line 208 of file ComplexMath.cpp.

6.5.3.53 template<class T > ComplexNumber<T> sec (const ComplexNumber< T > & *z*) [related]

Definition at line 324 of file ComplexMath.cpp.

6.5.3.54 template<class T > ComplexNumber<T> sech (const ComplexNumber< T > & *z*) [related]

Definition at line 469 of file ComplexMath.cpp.

6.5.3.55 template<class T > ComplexNumber<T> sin (const ComplexNumber< T > & *z*) [related]

Definition at line 293 of file ComplexMath.cpp.

6.5.3.56 template<class T > ComplexNumber<T> sinh (const ComplexNumber< T > & *z*) [related]

Definition at line 438 of file ComplexMath.cpp.

6.5.3.57 template<class T > ComplexNumber<T> sqrt (const ComplexNumber< T > & *z*) [related]

Definition at line 221 of file ComplexMath.cpp.

6.5.3.58 template<class T > ComplexNumber<T> sqrt (const T & *x*) [related]

Definition at line 232 of file ComplexMath.cpp.

6.5.3.59 template<class T > ComplexNumber<T> tan (const ComplexNumber< T > & *z*) [related]

Definition at line 307 of file ComplexMath.cpp.

6.5.3.60 template<class T > ComplexNumber<T> tanh (const ComplexNumber< T > & *z*) [related]

Definition at line 452 of file ComplexMath.cpp.

6.6 src/ComplexNumber.cpp File Reference

Representation of a complex number.

```
#include "ComplexNumber.h"  #include "MathExceptions.h"  #include "Namespace.h"  #include <iostream>  #include <cmath>
```

Namespaces

- namespace math

  Math namespace.
- namespace complex

  Complex namespace.

Defines

- #define COMPLEX_NUMBER_CPP

Functions

- template<class T >
  ComplexNumber< T > operator+ (const ComplexNumber< T > &lhs, const ComplexNumber< T > &rhs)
- template<class T >
  ComplexNumber< T > operator+ (const ComplexNumber< T > &lhs, const T &rhs)
- template<class T >
  ComplexNumber< T > operator+ (const T &lhs, const ComplexNumber< T > &rhs)

- template<class T >
  ComplexNumber< T > operator- (const ComplexNumber< T > &lhs, const ComplexNumber< T > &rhs)
- template<class T >
  ComplexNumber< T > operator- (const ComplexNumber< T > &lhs, const T &rhs)
- template<class T >
  ComplexNumber< T > operator- (const T &lhs, const ComplexNumber< T > &rhs)
- template<class T >
  ComplexNumber< T > operator* (const ComplexNumber< T > &lhs, const ComplexNumber< T > &rhs)
- template<class T >
  ComplexNumber< T > operator* (const ComplexNumber< T > &lhs, const T &rhs)
- template<class T >
  ComplexNumber< T > operator* (const T &lhs, const ComplexNumber< T > &rhs)
- template<class T >
  ComplexNumber< T > operator/ (const ComplexNumber< T > &lhs, const ComplexNumber< T > &rhs)
- template<class T >
  ComplexNumber< T > operator/ (const ComplexNumber< T > &lhs, const T &rhs)
- template<class T >
  ComplexNumber< T > operator/ (const T &lhs, const ComplexNumber< T > &rhs)
- template<class T , class U >
  bool operator== (const ComplexNumber< T > &lhs, const ComplexNumber< U > &rhs)
- template<class T , class U >
  bool operator== (const ComplexNumber< T > &lhs, const U &rhs)
- template<class T , class U >
  bool operator== (const T &lhs, const ComplexNumber< U > &rhs)
- template<class T , class U >
  bool operator!= (const ComplexNumber< T > &lhs, const ComplexNumber< U > &rhs)
- template<class T , class U >
  bool operator!= (const ComplexNumber< T > &lhs, const U &rhs)
- template<class T , class U >
  bool operator!= (const T &lhs, const ComplexNumber< U > &rhs)
- template<class U >
  std::ostream & operator<< (std::ostream &out, const ComplexNumber< U > &z)

- template<class U >
  std::istream & operator>> (std::istream &in, ComplexNumber< U > &z)

6.6.1 Detailed Description

Representation of a complex number.

Author

Matthew Krupcale

Version

0.1.2

Date

2010

This file contains the implementation of the class template that represents a complex number, its function template implementations as well as its overloaded operators

Definition in file ComplexNumber.cpp.

6.6.2 Define Documentation

6.6.2.1 #define COMPLEX_NUMBER_CPP

Definition at line 25 of file ComplexNumber.cpp.

6.6.3 Function Documentation

6.6.3.1 template<class T , class U > bool operator!= (const ComplexNumber< T > & *lhs*, const ComplexNumber< U > & *rhs*) [related]

Definition at line 355 of file ComplexNumber.cpp.

6.6.3.2 template<class T , class U > bool operator!= (const ComplexNumber< T > & *lhs*, const U & *rhs*) [related]

Definition at line 361 of file ComplexNumber.cpp.

6.6.3.3 template<class T , class U > bool operator!= (const T & *lhs*, const ComplexNumber< U > & *rhs*) [related]

Definition at line 366 of file ComplexNumber.cpp.

6.6.3.4 template<class T > ComplexNumber<T> operator* (const ComplexNumber< T > & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Definition at line 280 of file ComplexNumber.cpp.

6.6.3.5 template<class T > ComplexNumber<T> operator* (const ComplexNumber< T > & *lhs*, const T & *rhs*) [related]

Definition at line 289 of file ComplexNumber.cpp.

6.6.3.6 template<class T > ComplexNumber<T> operator* (const T & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Definition at line 294 of file ComplexNumber.cpp.

6.6.3.7 template<class T > ComplexNumber<T> operator+ (const ComplexNumber< T > & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Definition at line 246 of file ComplexNumber.cpp.

6.6.3.8 template<class T > ComplexNumber<T> operator+ (const ComplexNumber< T > & *lhs*, const T & *rhs*) [related]

Definition at line 253 of file ComplexNumber.cpp.

6.6.3.9 template<class T > ComplexNumber<T> operator+ (const T & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Definition at line 258 of file ComplexNumber.cpp.

6.6.3.10 template<class T > ComplexNumber<T> operator- (const ComplexNumber< T > & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Definition at line 263 of file ComplexNumber.cpp.

6.6.3.11 template<class T > ComplexNumber<T> operator- (const ComplexNumber< T > & *lhs*, const T & *rhs*) [related]

Definition at line 270 of file ComplexNumber.cpp.

6.6.3.12 template<class T > ComplexNumber<T> operator- (const T & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Definition at line 275 of file ComplexNumber.cpp.

6.6.3.13 template<class T > ComplexNumber<T> operator/ (const ComplexNumber< T > & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Definition at line 299 of file ComplexNumber.cpp.

6.6.3.14 template<class T > ComplexNumber<T> operator/ (const ComplexNumber< T > & *lhs*, const T & *rhs*) [related]

Definition at line 316 of file ComplexNumber.cpp.

6.6.3.15 template<class T > ComplexNumber<T> operator/ (const T & *lhs*, const ComplexNumber< T > & *rhs*) [related]

Definition at line 325 of file ComplexNumber.cpp.

6.6.3.16 template<class U > std::ostream& operator<< (std::ostream & *out*, const ComplexNumber< U > & *z*)

Sends a representation of this ComplexNumber into the ostream

Parameters

in	*out*	The ostream reference into which the ComplexNumber will be inserted
in	*z*	The ComplexNumber from which the data will be obtained

Returns

A reference to the modified ostream

Definition at line 371 of file ComplexNumber.cpp.

6.6.3.17 template<class T , class U > bool operator== (const ComplexNumber< T > & *lhs,* const ComplexNumber< U > & *rhs*) [related]

Definition at line 339 of file ComplexNumber.cpp.

6.6.3.18 template<class T , class U > bool operator== (const ComplexNumber< T > & *lhs,* const U & *rhs*) [related]

Definition at line 345 of file ComplexNumber.cpp.

6.6.3.19 template<class T , class U > bool operator== (const T & *lhs,* const ComplexNumber< U > & *rhs*) [related]

Definition at line 350 of file ComplexNumber.cpp.

6.6.3.20 template<class U > std::istream& operator>> (std::istream & *in,* ComplexNumber< U > & *z*)

Initializes a ComplexNumber from values extracted from the istream

Parameters

in	*in*	The istream reference from which the ComplexNumber will be extracted
in	*z*	The ComplexNumber into which the data will be inserted

Returns

A reference to the modified istream

Definition at line 383 of file ComplexNumber.cpp.