

QuantumGates

Generated by Doxygen 1.8.17

1 Class Index	1
1 Class Index	1
1.1 Class List	1
2 File Index	2
2.1 File List	2
3 Namespace Documentation	2
3.1 quantum Namespace Reference	2
4 Class Documentation	2
4.1 quantum::QuantumComputer Struct Reference	2
4.1.1 Detailed Description	3
4.1.2 Constructor & Destructor Documentation	3
4.1.3 Member Function Documentation	3
4.1.4 Member Data Documentation	5
5 File Documentation	5
5.1 matrixOperation.h File Reference	5
5.1.1 Typedef Documentation	6
5.1.2 Function Documentation	6
5.2 quantumComputer.h File Reference	7
5.3 quantumGate.h File Reference	7
5.3.1 Typedef Documentation	8
5.3.2 Function Documentation	8
5.3.3 Variable Documentation	16
5.4 quantumGateOperation.h File Reference	16
5.4.1 Typedef Documentation	17
5.4.2 Function Documentation	17
5.5 qubitOperation.h File Reference	19
5.5.1 Typedef Documentation	19
5.5.2 Function Documentation	20
5.5.3 Variable Documentation	21
Index	23

1 Class Index

1.1 Class List

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

quantum::QuantumComputer QuantumComputer structure	2
--	----------

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

matrixOperation.h	5
quantumComputer.h	7
quantumGate.h	7
quantumGateOperation.h	16
qubitOperation.h	19

3 Namespace Documentation

3.1 quantum Namespace Reference

Classes

- struct [QuantumComputer](#)
[QuantumComputer](#) structure.

4 Class Documentation

4.1 quantum::QuantumComputer Struct Reference

[QuantumComputer](#) structure.

```
#include <quantumComputer.h>
```

Public Member Functions

- [QuantumComputer](#) (int regSize, double probability[], int arrSize)
- void [countNonZeroBaseVector](#) ()
- void [resetState](#) ()
- void [viewProbability](#) ()

Used to show probabilities of base vector.

- void [viewQubitsInMathExpression](#) ()
- void [validateProbability](#) ()
- void [normalizeRegister](#) ()

Used to normalize probabilities of base vector (register)

- void [measure](#) ()

Used to measure probabilities of base vector (not implemented yet)

- vector< double > [getBaseVector](#) ()

Static Public Member Functions

- static void [validateArraySize](#) (int arrSize, int regSize)

Public Attributes

- int [registerSize](#)
- int [baseVectorsCount](#)
- bool [isNormalize](#)
- bool [isMeasured](#)
- vector< double > [baseVector](#)

4.1.1 Detailed Description

[QuantumComputer](#) structure.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 QuantumComputer() `quantum::QuantumComputer::QuantumComputer (int regSize, double probability[], int arrSize)`

[QuantumComputer](#) constructor

Parameters

<i>registerSize</i>	int
<i>probabilities</i>	double[]
<i>arraysize</i>	int

4.1.3 Member Function Documentation

4.1.3.1 countNonZeroBaseVector() `void quantum::QuantumComputer::countNonZeroBaseVector ()`

Used to count elements of vector
where element not equal zero

4.1.3.2 getBaseVector() `vector<double> quantum::QuantumComputer::getBaseVector ()`

Used to get created base vector

Returns

base vector

4.1.3.3 measure() `void quantum::QuantumComputer::measure ()`

Used to measure probabilities of base vector (not implemented yet)

4.1.3.4 normalizeRegister() `void quantum::QuantumComputer::normalizeRegister ()`

Used to normalize probabilities of base vector (register)

4.1.3.5 resetState() `void quantum::QuantumComputer::resetState ()`

Used to reset state of base vector.

First element of vector is set to one, other elements are set to 0.

4.1.3.6 validateArraySize() `static void quantum::QuantumComputer::validateArraySize (`
 `int arrSize,`
 `int regSize) [static]`

Used to check register size and array size from input

Parameters

<i>arraySize</i>	int
<i>registerSize</i>	int

4.1.3.7 validateProbability() `void quantum::QuantumComputer::validateProbability ()`

Used to check probabilities of base vector.

If sum of square probabilities is not equal one, [normalizeRegister\(\)](#) and [resetState\(\)](#) is executed.

4.1.3.8 viewProbability() `void quantum::QuantumComputer::viewProbability ()`

Used to show probabilities of base vector.

4.1.3.9 viewQubitsInMathExpression() `void quantum::QuantumComputer::viewQubitsInMathExpression ()`

Used to show qubit as math expression
eg. for qubit 0 - $|0\rangle$

4.1.4 Member Data Documentation

4.1.4.1 baseVector `vector<double> quantum::QuantumComputer::baseVector`

4.1.4.2 baseVectorsCount `int quantum::QuantumComputer::baseVectorsCount`

4.1.4.3 isMeasured `bool quantum::QuantumComputer::isMeasured`

4.1.4.4 isNormalize `bool quantum::QuantumComputer::isNormalize`

4.1.4.5 registerSize `int quantum::QuantumComputer::registerSize`

5 File Documentation

5.1 matrixOperation.h File Reference

```
#include <complex>
```

Typedefs

- `typedef vector< vector< complex< double > > > vector2d`
Used as alias for declaration of two dimensional vector.

Functions

- `vector2d getPreparedVectorForHermitianMatrix` (int dimension)
- `vector2d getRandomHermitianMatrix` (int dimension)
- `vector2d makeConjugateTranspose` (vector2d matrix)
- `void showMatrix` (vector2d matrix)

5.1.1 Typedef Documentation

5.1.1.1 **vector2d** `typedef vector<vector<complex<double> > > vector2d`

Used as alias for declaration of two dimensional vector.

5.1.2 Function Documentation

5.1.2.1 **getPreparedVectorForHermitianMatrix()** `vector2d getPreparedVectorForHermitianMatrix (int dimension)`

Used to get defined vector for declared dimension

Parameters

<i>dimension</i>	int
------------------	-----

Returns

defined matrix as two dimensional vector

5.1.2.2 **getRandomHermitianMatrix()** `vector2d getRandomHermitianMatrix (int dimension)`

Used to get random hermitian matrix.

Hermitian matrix - https://pl.wikipedia.org/wiki/Macierz_hermitowska

Parameters

<i>dimension</i>	int
------------------	-----

Returns

hermitian matrix as two dimensional vector

5.1.2.3 **makeConjugateTranspose()** `vector2d makeConjugateTranspose (vector2d matrix)`

Used to make conjugate transpose of matrix.

Conjugate transpose - https://en.wikipedia.org/wiki/Conjugate_transpose#Example

Parameters

<i>matrix</i>	vector2d
---------------	----------

Returns

conjugate transposed matrix as two dimensional vector

5.1.2.4 showMatrix() `void showMatrix (`
 `vector2d matrix)`

Used to show all elements of matrix

Parameters

<i>matrix</i>	vector2d
<i>dimension</i>	int

5.2 quantumComputer.h File Reference

```
#include <vector>
```

Classes

- struct [quantum::QuantumComputer](#)
QuantumComputer structure.

Namespaces

- [quantum](#)

5.3 quantumGate.h File Reference

```
#include <complex>
#include <vector>
```

Typedefs

- typedef `vector< vector< complex< double > > >` [vector2d](#)
Used as alias for declaration of two dimensional vector.

Functions

- [vector2d getPreparedContainerForQuantumGate](#) (int dimension)
- [vector2d makeNotOnQubit](#) ([vector2d](#) qubit)
- [vector2d makeSqrtNotOnQubit](#) ([vector2d](#))
- [vector2d makeCnotOnQubit](#) ([vector2d](#) qubit)
- [vector2d makeSwapOnQubit](#) ([vector2d](#) qubit)
- [vector2d makeFredkinOnQubit](#) ([vector2d](#) qubit)
- [vector2d makeToffoliOnQubit](#) ([vector2d](#) qubit)
- [vector2d makeHadamardOnQubit](#) ([vector2d](#) qubit)
- [vector2d makeMultidimensionalHadamardOnQubit](#) ([vector2d](#) qubit, [vector2d](#) hadamardGate, int index↔Number)
- [vector2d makePhaseShiftOnQubit](#) ([vector2d](#) qubit, double angle)
- [vector2d makePauliXOnQubit](#) ([vector2d](#) qubit)
- [vector2d makePauliYOnQubit](#) ([vector2d](#) qubit)
- [vector2d makePauliZOnQubit](#) ([vector2d](#) qubit)
- [vector2d getNotGate](#) ()
- [vector2d getSqrtNotGate](#) ()
- [vector2d getCnotGate](#) ()
- [vector2d getSwapGate](#) ()
- [vector2d getFredkinGate](#) ()
- [vector2d getToffoliGate](#) ()
- [vector2d getHadamardGate](#) ()
- [vector2d getMultidimensionalHadamardGate](#) (int indexNumber)
- [vector2d getPhaseShiftGate](#) (double angle)
- [vector2d getPauliXGate](#) ()
- [vector2d getPauliYGate](#) ()
- [vector2d getPauliZGate](#) ()
- void [showQuantumGate](#) ([vector2d](#) quantumGate)
- void [showPhaseShiftQuantumGate](#) ([vector2d](#) phaseShiftGate)

Variables

- const int [ONE_ARGUMENT_GATE_SIZE](#) = 2
- const int [TWO_ARGUMENTS_GATE_SIZE](#) = 4
- const int [THREE_ARGUMENTS_GATE_SIZE](#) = 8

5.3.1 Typedef Documentation

5.3.1.1 [vector2d](#) `typedef vector<vector<complex<double> > > vector2d`

Used as alias for declaration of two dimensional vector.

5.3.2 Function Documentation

5.3.2.1 `getCnotGate()` `vector2d` `getCnotGate ()`

Used to get CNOT quantum gate

Returns

CNOT quantum gate

5.3.2.2 `getFredkinGate()` `vector2d` `getFredkinGate ()`

Used to get FREDKIN quantum gate

Returns

FREDKIN quantum gate

5.3.2.3 `getHadamardGate()` `vector2d` `getHadamardGate ()`

Used to get HADAMARD quantum gate

Returns

HADAMARD quantum gate

5.3.2.4 `getMultidimensionalHadamardGate()` `vector2d` `getMultidimensionalHadamardGate (` `int indexNumber)`

Used to get multidimensional HADAMARD quantum gate

Parameters

<i>indexNumber</i>	int
--------------------	-----

Returns

multidimensional HADAMARD quantum gate

5.3.2.5 `getNotGate()` `vector2d` `getNotGate ()`

Used to get NOT quantum gate

Returns

NOT quantum gate

5.3.2.6 getPauliXGate() `vector2d` getPauliXGate ()

Used to get PAULI X quantum gate

Returns

PAULI X quantum gate

5.3.2.7 getPauliYGate() `vector2d` getPauliYGate ()

Used to get PAULI Y quantum gate

Returns

PAULI Y quantum gate

5.3.2.8 getPauliZGate() `vector2d` getPauliZGate ()

Used to get PAULI Z quantum gate

Returns

PAULI Z quantum gate

5.3.2.9 getPhaseShiftGate() `vector2d` getPhaseShiftGate (
 `double angle`)

Used to get PHASE SHIFT quantum gate

Parameters

<i>angle</i>	double - angle as value eg. PI or -PI
--------------	---------------------------------------

Returns

PHASE SHIFT quantum gate

5.3.2.10 getPreparedContainerForQuantumGate() `vector2d` getPreparedContainerForQuantumGate (
 `int` *dimension*)

Used to generate and get quantum gate for declared dimensions

Parameters

<i>dimension</i>	<code>int</code>
------------------	------------------

5.3.2.11 getSqrtNotGate() `vector2d` getSqrtNotGate ()

Used to get SQRT(NOT) quantum gate

Returns

SQRT(NOT) quantum gate

5.3.2.12 getSwapGate() `vector2d` getSwapGate ()

Used to get SWAP quantum gate

Returns

SWAP quantum gate

5.3.2.13 getToffoliGate() `vector2d` getToffoliGate ()

Used to get TOFFOLI quantum gate

Returns

TOFFOLI quantum gate

5.3.2.14 makeCnotOnQubit() `vector2d` makeCnotOnQubit (
 `vector2d` *qubit*)

Used to make CNOT quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.15 makeFredkinOnQubit() `vector2d` makeFredkinOnQubit (
 `vector2d` *qubit*)

Used to make FREDKIN quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.16 makeHadamardOnQubit() `vector2d` makeHadamardOnQubit (
 `vector2d` *qubit*)

Used to make HADAMARD quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.17 makeMultidimensionalHadamardOnQubit() `vector2d` makeMultidimensionalHadamardOnQubit (
 `vector2d` *qubit*,
 `vector2d` *hadamardGate*,
 `int` *indexNumber*)

Used to make multidimensional HADAMARD quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
<i>hadamardGate</i>	vector2d
<i>indexNumber</i>	int

Returns

updated qubit

5.3.2.18 makeNotOnQubit() `vector2d` makeNotOnQubit (
 `vector2d` *qubit*)

Used to make NOT quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.19 makePauliXOnQubit() `vector2d` makePauliXOnQubit (
 `vector2d` *qubit*)

Used to make PAULI X quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.20 makePauliYOnQubit() `vector2d` makePauliYOnQubit (
 `vector2d` *qubit*)

Used to make PAULI Y quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.21 makePauliZOnQubit() `vector2d` makePauliZOnQubit (
 `vector2d` *qubit*)

Used to make PAULI Z quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.22 makePhaseShiftOnQubit() `vector2d` makePhaseShiftOnQubit (
 `vector2d` *qubit*,
 `double` *angle*)

Used to make PHASE SHIFT quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
<i>angle</i>	double - angle as value eg. PI or -PI

Returns

updated qubit

5.3.2.23 makeSqrtNotOnQubit() `vector2d` makeSqrtNotOnQubit (
 `vector2d`)

Used to make SQRT(NOT) quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.24 makeSwapOnQubit() `vector2d makeSwapOnQubit (`
`vector2d qubit)`

Used to make SWAP quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.25 makeToffoliOnQubit() `vector2d makeToffoliOnQubit (`
`vector2d qubit)`

Used to make TOFFOLI quantum gate on qubit

Parameters

<i>qubit</i>	vector2d
--------------	----------

Returns

updated qubit

5.3.2.26 showPhaseShiftQuantumGate() `void showPhaseShiftQuantumGate (`
`vector2d phaseShiftGate)`

Used to show all elements of PHASE SHIFT quantum gate

Parameters

<i>phaseShiftGate</i>	vector2d
<i>gateSize</i>	const int

5.3.2.27 showQuantumGate() `void showQuantumGate (`
`vector2d quantumGate)`

Used to show all elements of quantum gate

Parameters

<i>quantumGate</i>	vector2d
<i>gateSize</i>	const int

5.3.3 Variable Documentation

5.3.3.1 ONE_ARGUMENT_GATE_SIZE `const int ONE_ARGUMENT_GATE_SIZE = 2`

Parameters

-	size of one argument quantum gates
---	------------------------------------

5.3.3.2 THREE_ARGUMENTS_GATE_SIZE `const int THREE_ARGUMENTS_GATE_SIZE = 8`

Parameters

-	size of three argument quantum gates
---	--------------------------------------

5.3.3.3 TWO_ARGUMENTS_GATE_SIZE `const int TWO_ARGUMENTS_GATE_SIZE = 4`

Parameters

-	size of two argument quantum gates
---	------------------------------------

5.4 quantumGateOperation.h File Reference

```
#include <complex>
#include <vector>
```

Typedefs

- typedef vector< vector< complex< double > > > [vector2d](#)

Used as alias for declaration of two dimensional vector.

Functions

- [vector2d](#) [composeQuantumGates](#) ([vector2d](#) firstGate, [vector2d](#) secondGate)
- [vector2d](#) [getIdentityMatrix](#) (int gateSize)
- bool [isIdentityMatrixAndComposedGatesAreEqual](#) ([vector2d](#) identityMatrix, [vector2d](#) composedGates)
- bool [isComposeOfGatesGivesIdentityMatrix](#) ([vector2d](#) firstGate, [vector2d](#) secondGate)
- bool [isMatrixUnitary](#) ([vector2d](#) quantumGate, [vector2d](#) conjugateTransposedQuantumGate)

5.4.1 Typedef Documentation

5.4.1.1 [vector2d](#) typedef vector<vector<complex<double> > > [vector2d](#)

Used as alias for declaration of two dimensional vector.

5.4.2 Function Documentation

5.4.2.1 [composeQuantumGates\(\)](#) [vector2d](#) [composeQuantumGates](#) ([vector2d](#) *firstGate*, [vector2d](#) *secondGate*)

Used to compose two quantum gates.

Compose - multiplication of values from two matrices, at the same indexes

Parameters

<i>firstGate</i>	vector2d
<i>secondGate</i>	vector2d

Returns

composed quantum gates

5.4.2.2 [getIdentityMatrix\(\)](#) [vector2d](#) [getIdentityMatrix](#) (int *gateSize*)

Used to get identity matrix for defined size.

Identity matrix - https://en.wikipedia.org/wiki/Identity_matrix

Parameters

<i>gateSize</i>	int
-----------------	-----

Returns

identity matrix

5.4.2.3 isComposeOfGatesGivesIdentityMatrix() `bool isComposeOfGatesGivesIdentityMatrix (`
`vector2d firstGate,`
`vector2d secondGate)`

Used to check
if composed gates gives identity matrix.

Parameters

<i>firstGate</i>	vector2d
<i>secondGate</i>	vector2d

Returns

true or false

5.4.2.4 isIdentityMatrixAndComposedGatesAreEqual() `bool isIdentityMatrixAndComposedGatesAre↵`
`Equal (`
`vector2d identityMatrix,`
`vector2d composedGates)`

Used to check
if identity matrix are the same as composed gates.

Parameters

<i>identityMatrix</i>	vector2d
<i>composedGates</i>	vector2d

Returns

true or false

5.4.2.5 isMatrixUnitary() `bool isMatrixUnitary (`
`vector2d quantumGate,`
`vector2d conjugateTransposedQuantumGate)`

Used to check if matrix is unitary.

Unitary matrix - https://en.wikipedia.org/wiki/Unitary_matrix

Parameters

<code>quantumGate</code>	<code>vector2d</code>
<code>conjugateTransposedQuantumGate</code>	<code>vector2d</code>

Returns

true or false

5.5 qubitOperation.h File Reference

```
#include <complex>
#include <vector>
```

Typedefs

- `typedef vector< vector< complex< double > > > vector2d`
Used as alias for declaration of two dimensional vector.

Functions

- `vector2d getPreparedContainerForQubit` (int qubitRows)
- `vector2d makeDotProductOfQubits` (`vector2d` firstQubit, `vector2d` secondQubit)
- `vector2d getQubitRepresentation` (vector< double > baseVector)
- void `showQubit` (`vector2d` qubit)
- void `showDotProduct` (`vector2d` dotProduct)

Variables

- const int `SINGLE_QUBIT_NUMBER_OF_ROWS` = 2
- const int `TWO_QUBITS_NUMBER_OF_ROWS` = 4
- const int `THREE_QUBITS_NUMBER_OF_ROWS` = 8
- const int `QUBIT_NUMBER_OF_COLUMNS` = 1

5.5.1 Typedef Documentation

5.5.1.1 vector2d `typedef vector<vector<complex<double> > > vector2d`

Used as alias for declaration of two dimensional vector.

5.5.2 Function Documentation

5.5.2.1 getPreparedContainerForQubit() `vector2d` getPreparedContainerForQubit (
 `int` *qubitRows*)

Used to get defined qubit for declared dimension

Parameters

<i>qubitRows</i>	<code>int</code>
------------------	------------------

Returns

allocated qubit

5.5.2.2 getQubitRepresentation() `vector2d` getQubitRepresentation (
 `vector< double >` *baseVector*)

Used to get qubit as complex type 2D array

Parameters

<i>baseVector</i>	<code>vector<double></code>
-------------------	-----------------------------------

Returns

qubit

5.5.2.3 makeDotProductOfQubits() `vector2d` makeDotProductOfQubits (
 `vector2d` *firstQubit*,
 `vector2d` *secondQubit*)

Used to make dot product of two qubits.

Dot product - https://en.wikipedia.org/wiki/Dot_product#Algebraic_definition

Parameters

<i>firstQubit</i>	<code>vector2d</code>
<i>secondQubit</i>	<code>vector2d</code>

Returns

dot product of qubits

5.5.2.4 showDotProduct() `void showDotProduct (`
`vector2d dotProduct)`

Used to show dot product

Parameters

<i>dotProduct</i>	vector2d
-------------------	----------

5.5.2.5 showQubit() `void showQubit (`
`vector2d qubit)`

Used to show all elements of qubit (2D array)

Parameters

<i>qubit</i>	vector2d
--------------	----------

5.5.3 Variable Documentation

5.5.3.1 QUBIT_NUMBER_OF_COLUMNS `const int QUBIT_NUMBER_OF_COLUMNS = 1`

Parameters

-	constant qubit column
---	-----------------------

5.5.3.2 SINGLE_QUBIT_NUMBER_OF_ROWS `const int SINGLE_QUBIT_NUMBER_OF_ROWS = 2`

Parameters

-	constant single qubit rows
---	----------------------------

5.5.3.3 THREE_QUBITS_NUMBER_OF_ROWS `const int THREE_QUBITS_NUMBER_OF_ROWS = 8`

Parameters

-	constant three qubits rows
---	----------------------------

5.5.3.4 TWO_QUBITS_NUMBER_OF_ROWS `const int TWO_QUBITS_NUMBER_OF_ROWS = 4`

Parameters

-	constant two qubits rows
---	--------------------------

Index

- baseVector
 - quantum::QuantumComputer, 5
- baseVectorsCount
 - quantum::QuantumComputer, 5
- composeQuantumGates
 - quantumGateOperation.h, 17
- countNonZeroBaseVector
 - quantum::QuantumComputer, 3
- getBaseVector
 - quantum::QuantumComputer, 3
- getCnotGate
 - quantumGate.h, 8
- getFredkinGate
 - quantumGate.h, 9
- getHadamardGate
 - quantumGate.h, 9
- getIdentityMatrix
 - quantumGateOperation.h, 17
- getMultidimensionalHadamardGate
 - quantumGate.h, 9
- getNotGate
 - quantumGate.h, 9
- getPauliXGate
 - quantumGate.h, 10
- getPauliYGate
 - quantumGate.h, 10
- getPauliZGate
 - quantumGate.h, 10
- getPhaseShiftGate
 - quantumGate.h, 10
- getPreparedContainerForQuantumGate
 - quantumGate.h, 11
- getPreparedContainerForQubit
 - qubitOperation.h, 20
- getPreparedVectorForHermitianMatrix
 - matrixOperation.h, 6
- getQubitRepresentation
 - qubitOperation.h, 20
- getRandomHermitianMatrix
 - matrixOperation.h, 6
- getSqrtNotGate
 - quantumGate.h, 11
- getSwapGate
 - quantumGate.h, 11
- getToffoliGate
 - quantumGate.h, 11
- isComposeOfGatesGivesIdentityMatrix
 - quantumGateOperation.h, 18
- isIdentityMatrixAndComposedGatesAreEqual
 - quantumGateOperation.h, 18
- isMatrixUnitary
 - quantumGateOperation.h, 18
- isMeasured
 - quantum::QuantumComputer, 5
- isNormalize
 - quantum::QuantumComputer, 5
- makeCnotOnQubit
 - quantumGate.h, 11
- makeConjugateTranspose
 - matrixOperation.h, 6
- makeDotProductOfQubits
 - qubitOperation.h, 20
- makeFredkinOnQubit
 - quantumGate.h, 12
- makeHadamardOnQubit
 - quantumGate.h, 12
- makeMultidimensionalHadamardOnQubit
 - quantumGate.h, 12
- makeNotOnQubit
 - quantumGate.h, 13
- makePauliXOnQubit
 - quantumGate.h, 13
- makePauliYOnQubit
 - quantumGate.h, 13
- makePauliZOnQubit
 - quantumGate.h, 14
- makePhaseShiftOnQubit
 - quantumGate.h, 14
- makeSqrtNotOnQubit
 - quantumGate.h, 14
- makeSwapOnQubit
 - quantumGate.h, 15
- makeToffoliOnQubit
 - quantumGate.h, 15
- matrixOperation.h, 5
 - getPreparedVectorForHermitianMatrix, 6
 - getRandomHermitianMatrix, 6
 - makeConjugateTranspose, 6
 - showMatrix, 7
 - vector2d, 6
- measure
 - quantum::QuantumComputer, 4
- normalizeRegister
 - quantum::QuantumComputer, 4
- ONE_ARGUMENT_GATE_SIZE
 - quantumGate.h, 16
- quantum, 2
- quantum::QuantumComputer, 2
 - baseVector, 5
 - baseVectorsCount, 5
 - countNonZeroBaseVector, 3
 - getBaseVector, 3
 - isMeasured, 5
 - isNormalize, 5
 - measure, 4

- normalizeRegister, 4
- QuantumComputer, 3
- registerSize, 5
- resetState, 4
- validateArraySize, 4
- validateProbability, 4
- viewProbability, 4
- viewQubitsInMathExpression, 4
- QuantumComputer
 - quantum::QuantumComputer, 3
- quantumComputer.h, 7
- quantumGate.h, 7
 - getCnotGate, 8
 - getFredkinGate, 9
 - getHadamardGate, 9
 - getMultidimensionalHadamardGate, 9
 - getNotGate, 9
 - getPauliXGate, 10
 - getPauliYGate, 10
 - getPauliZGate, 10
 - getPhaseShiftGate, 10
 - getPreparedContainerForQuantumGate, 11
 - getSqrtNotGate, 11
 - getSwapGate, 11
 - getToffoliGate, 11
 - makeCnotOnQubit, 11
 - makeFredkinOnQubit, 12
 - makeHadamardOnQubit, 12
 - makeMultidimensionalHadamardOnQubit, 12
 - makeNotOnQubit, 13
 - makePauliXOnQubit, 13
 - makePauliYOnQubit, 13
 - makePauliZOnQubit, 14
 - makePhaseShiftOnQubit, 14
 - makeSqrtNotOnQubit, 14
 - makeSwapOnQubit, 15
 - makeToffoliOnQubit, 15
 - ONE_ARGUMENT_GATE_SIZE, 16
 - showPhaseShiftQuantumGate, 15
 - showQuantumGate, 16
 - THREE_ARGUMENTS_GATE_SIZE, 16
 - TWO_ARGUMENTS_GATE_SIZE, 16
 - vector2d, 8
- quantumGateOperation.h, 16
 - composeQuantumGates, 17
 - getIdentityMatrix, 17
 - isComposeOfGatesGivesIdentityMatrix, 18
 - isIdentityMatrixAndComposedGatesAreEqual, 18
 - isMatrixUnitary, 18
 - vector2d, 17
- QUBIT_NUMBER_OF_COLUMNS
 - qubitOperation.h, 21
- qubitOperation.h, 19
 - getPreparedContainerForQubit, 20
 - getQubitRepresentation, 20
 - makeDotProductOfQubits, 20
 - QUBIT_NUMBER_OF_COLUMNS, 21
 - showDotProduct, 21
 - showQubit, 21
 - SINGLE_QUBIT_NUMBER_OF_ROWS, 21
 - THREE_QUBITS_NUMBER_OF_ROWS, 21
 - TWO_QUBITS_NUMBER_OF_ROWS, 22
 - vector2d, 19
- registerSize
 - quantum::QuantumComputer, 5
- resetState
 - quantum::QuantumComputer, 4
- showDotProduct
 - qubitOperation.h, 21
- showMatrix
 - matrixOperation.h, 7
- showPhaseShiftQuantumGate
 - quantumGate.h, 15
- showQuantumGate
 - quantumGate.h, 16
- showQubit
 - qubitOperation.h, 21
- SINGLE_QUBIT_NUMBER_OF_ROWS
 - qubitOperation.h, 21
- THREE_ARGUMENTS_GATE_SIZE
 - quantumGate.h, 16
- THREE_QUBITS_NUMBER_OF_ROWS
 - qubitOperation.h, 21
- TWO_ARGUMENTS_GATE_SIZE
 - quantumGate.h, 16
- TWO_QUBITS_NUMBER_OF_ROWS
 - qubitOperation.h, 22
- validateArraySize
 - quantum::QuantumComputer, 4
- validateProbability
 - quantum::QuantumComputer, 4
- vector2d
 - matrixOperation.h, 6
 - quantumGate.h, 8
 - quantumGateOperation.h, 17
 - qubitOperation.h, 19
- viewProbability
 - quantum::QuantumComputer, 4
- viewQubitsInMathExpression
 - quantum::QuantumComputer, 4