

QuantumGates

Generated by Doxygen 1.8.17

1 Namespace Index	1
1.1 Namespace List	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Namespace Documentation	7
4.1 quantum Namespace Reference	7
5 Class Documentation	9
5.1 quantum::QuantumComputer Struct Reference	9
5.1.1 Detailed Description	10
5.1.2 Constructor & Destructor Documentation	10
5.1.2.1 QuantumComputer()	10
5.1.3 Member Function Documentation	10
5.1.3.1 countNonZeroBaseVector()	10
5.1.3.2 getBaseVector()	10
5.1.3.3 measure()	11
5.1.3.4 normalizeRegister()	11
5.1.3.5 resetState()	11
5.1.3.6 validateArraySize()	11
5.1.3.7 validateProbability()	11
5.1.3.8 viewProbability()	12
5.1.3.9 viewQubitsInMathExpression()	12
5.1.4 Member Data Documentation	12
5.1.4.1 baseVector	12
5.1.4.2 baseVectorsCount	12
5.1.4.3 isMeasured	12
5.1.4.4 isNormalize	12
5.1.4.5 registerSize	12
6 File Documentation	13
6.1 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/matrixOperation.h File Reference	13
6.1.1 Function Documentation	13
6.1.1.1 getAllocatedMatrix()	13
6.1.1.2 getRandomHermitianMatrix()	14
6.1.1.3 makeConjugateTranspose()	14
6.1.1.4 showMatrix()	14
6.2 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/quantumComputer.h File Reference	15

6.3 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/quantumGate.h File Reference	15
6.3.1 Function Documentation	16
6.3.1.1 getAllocatedQuantumGate()	16
6.3.1.2 getCnotGate()	16
6.3.1.3 getFredkinGate()	17
6.3.1.4 getHadamardGate()	17
6.3.1.5 getMultidimensionalHadamardGate()	17
6.3.1.6 getNotGate()	17
6.3.1.7 getPauliXGate()	18
6.3.1.8 getPauliYGate()	18
6.3.1.9 getPauliZGate()	18
6.3.1.10 getPhaseShiftGate()	18
6.3.1.11 getSqrtNotGate()	19
6.3.1.12 getSwapGate()	19
6.3.1.13 getToffoliGate()	19
6.3.1.14 makeCnotOnQubit()	19
6.3.1.15 makeFredkinOnQubit()	20
6.3.1.16 makeHadamardOnQubit()	20
6.3.1.17 makeMultidimensionalHadamardOnQubit()	20
6.3.1.18 makeNotOnQubit()	21
6.3.1.19 makePauliXOnQubit()	21
6.3.1.20 makePauliYOnQubit()	22
6.3.1.21 makePauliZOnQubit()	22
6.3.1.22 makePhaseShiftOnQubit()	22
6.3.1.23 makeSqrtNotOnQubit()	23
6.3.1.24 makeSwapOnQubit()	23
6.3.1.25 makeToffoliOnQubit()	24
6.3.1.26 showMultidimensionalHadamardGate()	24
6.3.1.27 showPhaseShiftQuantumGate()	24
6.3.1.28 showQuantumGate()	25
6.3.2 Variable Documentation	25
6.3.2.1 ONE_ARGUMENT_GATE_SIZE	25
6.3.2.2 THREE_ARGUMENTS_GATE_SIZE	25
6.3.2.3 TWO_ARGUMENTS_GATE_SIZE	25
6.4 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/quantumGateOperation.h File Reference	26
6.4.1 Function Documentation	26
6.4.1.1 composeQuantumGates()	26
6.4.1.2 getIdentityMatrix()	26
6.4.1.3 isComposeOfGatesGivesIdentityMatrix()	27
6.4.1.4 isIdentityMatrixAndComposedGatesAreEqual()	27
6.4.1.5 isMatrixUnitary()	28

6.5 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/qubitOperation.h File Reference	28
6.5.1 Function Documentation	29
6.5.1.1 getAllocatedQubit()	29
6.5.1.2 getQubitRepresentation()	29
6.5.1.3 makeDotProductOfQubits()	29
6.5.1.4 showDotProduct()	30
6.5.1.5 showQubit()	30
6.5.1.6 showQubitAfterConjugateTranspose()	30
6.5.2 Variable Documentation	31
6.5.2.1 QUBIT_NUMBER_OF_COLUMNS	31
6.5.2.2 SINGLE_QUBIT_NUMBER_OF_ROWS	31
6.5.2.3 THREE_QUBITS_NUMBER_OF_ROWS	31
6.5.2.4 TWO_QUBITS_NUMBER_OF_ROWS	31
Index	33

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

quantum	7
-------------------------	-------	-------------------

Chapter 2

Class Index

2.1 Class List

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

quantum::QuantumComputer	
QuantumComputer structure	9

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/ matrixOperation.h	13
/home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/ quantumComputer.h	15
/home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/ quantumGate.h	15
/home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/ quantumGateOperation.h	26
/home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/ qubitOperation.h	28

Chapter 4

Namespace Documentation

4.1 quantum Namespace Reference

Classes

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

Chapter 5

Class Documentation

5.1 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](#)

5.1.1 Detailed Description

[QuantumComputer](#) structure.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 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

5.1.3 Member Function Documentation

5.1.3.1 countNonZeroBaseVector()

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

Used to count elements of vector
where element not equal zero

5.1.3.2 getBaseVector()

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

Used to get created base vector

Returns

base vector

5.1.3.3 measure()

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

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

5.1.3.4 normalizeRegister()

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

Used to normalize probabilities of base vector (register)

5.1.3.5 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.

5.1.3.6 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

5.1.3.7 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.

5.1.3.8 viewProbability()

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

Used to show probabilities of base vector.

5.1.3.9 viewQubitsInMathExpression()

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

Used to show qubit as math expression

eg. for qubit 0 - $|0\rangle$

5.1.4 Member Data Documentation

5.1.4.1 baseVector

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

5.1.4.2 baseVectorsCount

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

5.1.4.3 isMeasured

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

5.1.4.4 isNormalize

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

5.1.4.5 registerSize

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

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

- </home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/quantumComputer.h>

Chapter 6

File Documentation

6.1 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/matrixOperation.h File Reference

```
#include <complex>
```

Functions

- `complex< double > ** getAllocatedMatrix (int firstDimension, int secondDimension)`
- `complex< double > ** getRandomHermitianMatrix (int dimension)`
- `complex< double > ** makeConjugateTranspose (complex< double > **matrix, int rows, int columns)`
- `void showMatrix (complex< double > **matrix, int dimension)`

6.1.1 Function Documentation

6.1.1.1 `getAllocatedMatrix()`

```
complex<double>** getAllocatedMatrix (
    int firstDimension,
    int secondDimension )
```

Used to generate and get matrix for declared dimensions

Parameters

<i>firstDimension</i>	int
<i>secondDimension</i>	int

Returns

allocated matrix

6.1.1.2 getRandomHermitianMatrix()

```
complex<double>** getRandomHermitianMatrix (
    int dimension )
```

Used to get random hermitian matrix.

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

Parameters

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

Returns

allocated matrix

6.1.1.3 makeConjugateTranspose()

```
complex<double>** makeConjugateTranspose (
    complex< double > ** matrix,
    int rows,
    int columns )
```

Used to make conjugate transpose of matrix.

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

Parameters

<i>matrix</i>	complex<double>
<i>rows</i>	int
<i>columns</i>	int

Returns

conjugate transposed matrix

6.1.1.4 showMatrix()

```
void showMatrix (
    complex< double > ** matrix,
    int dimension )
```

Used to show all elements of matrix

Parameters

<i>matrix</i>	complex<double>
<i>dimension</i>	int

6.2 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/quantumComputer.h File Reference

```
#include <vector>
```

Classes

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

Namespaces

- [quantum](#)

6.3 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/quantumGate.h File Reference

```
#include <complex>
```

Functions

- complex< double > ** [getAllocatedQuantumGate](#) (int dimension)
- complex< double > ** [makeNotOnQubit](#) (complex< double > **qubit)
- complex< double > ** [makeSqrtNotOnQubit](#) (complex< double > **qubit)
- complex< double > ** [makeCnotOnQubit](#) (complex< double > **qubit)
- complex< double > ** [makeSwapOnQubit](#) (complex< double > **qubit)
- complex< double > ** [makeFredkinOnQubit](#) (complex< double > **qubit)
- complex< double > ** [makeToffoliOnQubit](#) (complex< double > **qubit)
- complex< double > ** [makeHadamardOnQubit](#) (complex< double > **qubit)
- complex< double > ** [makeMultidimensionalHadamardOnQubit](#) (complex< double > **qubit, int number↵OfQubits, complex< double > **hadamardGate, int indexNumber)
- complex< double > ** [makePhaseShiftOnQubit](#) (complex< double > **qubit, double angle)
- complex< double > ** [makePauliXOnQubit](#) (complex< double > **qubit)
- complex< double > ** [makePauliYOnQubit](#) (complex< double > **qubit)

- `complex< double > ** makePauliZOnQubit (complex< double > **qubit)`
- `complex< double > ** getNotGate ()`
- `complex< double > ** getSqrtNotGate ()`
- `complex< double > ** getCnotGate ()`
- `complex< double > ** getSwapGate ()`
- `complex< double > ** getFredkinGate ()`
- `complex< double > ** getToffoliGate ()`
- `complex< double > ** getHadamardGate ()`
- `complex< double > ** getMultidimensionalHadamardGate (int indexNumber)`
- `complex< double > ** getPhaseShiftGate (double angle)`
- `complex< double > ** getPauliXGate ()`
- `complex< double > ** getPauliYGate ()`
- `complex< double > ** getPauliZGate ()`
- `void showQuantumGate (complex< double > **quantumGate, const int gateSize)`
- `void showPhaseShiftQuantumGate (complex< double > **phaseShiftGate, const int gateSize)`
- `void showMultidimensionalHadamardGate (complex< double > **hadamardGate, int indexNumber)`

Variables

- `const int ONE_ARGUMENT_GATE_SIZE = 2`
- `const int TWO_ARGUMENTS_GATE_SIZE = 4`
- `const int THREE_ARGUMENTS_GATE_SIZE = 8`

6.3.1 Function Documentation

6.3.1.1 getAllocatedQuantumGate()

```
complex<double>** getAllocatedQuantumGate (
    int dimension )
```

Used to generate and get quantum gate for declared dimensions

Parameters

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

6.3.1.2 getCnotGate()

```
complex<double>** getCnotGate ( )
```

Used to get CNOT quantum gate

Returns

CNOT quantum gate

6.3.1.3 getFredkinGate()

```
complex<double>** getFredkinGate ( )
```

Used to get FREDKIN quantum gate

Returns

FREDKIN quantum gate

6.3.1.4 getHadamardGate()

```
complex<double>** getHadamardGate ( )
```

Used to get HADAMARD quantum gate

Returns

HADAMARD quantum gate

6.3.1.5 getMultidimensionalHadamardGate()

```
complex<double>** getMultidimensionalHadamardGate (
    int indexNumber )
```

Used to get multidimensional HADAMARD quantum gate

Parameters

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

Returns

multidimensional HADAMARD quantum gate

6.3.1.6 getNotGate()

```
complex<double>** getNotGate ( )
```

Used to get NOT quantum gate

Returns

NOT quantum gate

6.3.1.7 getPauliXGate()

```
complex<double>** getPauliXGate ( )
```

Used to get PAULI X quantum gate

Returns

PAULI X quantum gate

6.3.1.8 getPauliYGate()

```
complex<double>** getPauliYGate ( )
```

Used to get PAULI Y quantum gate

Returns

PAULI Y quantum gate

6.3.1.9 getPauliZGate()

```
complex<double>** getPauliZGate ( )
```

Used to get PAULI Z quantum gate

Returns

PAULI Z quantum gate

6.3.1.10 getPhaseShiftGate()

```
complex<double>** 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

6.3.1.11 getSqrtNotGate()

```
complex<double>** getSqrtNotGate ( )
```

Used to get SQRT(NOT) quantum gate

Returns

SQRT(NOT) quantum gate

6.3.1.12 getSwapGate()

```
complex<double>** getSwapGate ( )
```

Used to get SWAP quantum gate

Returns

SWAP quantum gate

6.3.1.13 getToffoliGate()

```
complex<double>** getToffoliGate ( )
```

Used to get TOFFOLI quantum gate

Returns

TOFFOLI quantum gate

6.3.1.14 makeCnotOnQubit()

```
complex<double>** makeCnotOnQubit (
    complex< double > ** qubit )
```

Used to make CNOT quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.15 makeFredkinOnQubit()

```
complex<double>** makeFredkinOnQubit (
    complex< double > ** qubit )
```

Used to make FREDKIN quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.16 makeHadamardOnQubit()

```
complex<double>** makeHadamardOnQubit (
    complex< double > ** qubit )
```

Used to make HADAMARD quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.17 makeMultidimensionalHadamardOnQubit()

```
complex<double>** makeMultidimensionalHadamardOnQubit (
    complex< double > ** qubit,
```

```
int numberOfQubits,  
complex< double > ** hadamardGate,  
int indexNumber )
```

Used to make multidimensional HADAMARD quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
<i>numberOfQubits</i>	int
<i>hadamardGate</i>	complex<double>
<i>indexNumber</i>	int

Returns

updated qubit

6.3.1.18 makeNotOnQubit()

```
complex<double>** makeNotOnQubit (  
    complex< double > ** qubit )
```

Used to make NOT quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.19 makePauliXOnQubit()

```
complex<double>** makePauliXOnQubit (  
    complex< double > ** qubit )
```

Used to make PAULI X quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.20 makePauliYOnQubit()

```
complex<double>** makePauliYOnQubit (
    complex< double > ** qubit )
```

Used to make PAULI Y quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.21 makePauliZOnQubit()

```
complex<double>** makePauliZOnQubit (
    complex< double > ** qubit )
```

Used to make PAULI Z quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.22 makePhaseShiftOnQubit()

```
complex<double>** makePhaseShiftOnQubit (
    complex< double > ** qubit,
    double angle )
```

Used to make PHASE SHIFT quantum gate on qubit

Parameters

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

Returns

updated qubit

6.3.1.23 makeSqrtNotOnQubit()

```
complex<double>** makeSqrtNotOnQubit (
    complex< double > ** qubit )
```

Used to make SQRT(NOT) quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.24 makeSwapOnQubit()

```
complex<double>** makeSwapOnQubit (
    complex< double > ** qubit )
```

Used to make SWAP quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.25 makeToffoliOnQubit()

```
complex<double>** makeToffoliOnQubit (
    complex< double > ** qubit )
```

Used to make TOFFOLI quantum gate on qubit

Parameters

<i>qubit</i>	complex<double>
--------------	-----------------

Returns

updated qubit

6.3.1.26 showMultidimensionalHadamardGate()

```
void showMultidimensionalHadamardGate (
    complex< double > ** hadamardGate,
    int indexNumber )
```

Used to show all elements of multidimensional HADAMARD quantum gate

Parameters

<i>hadamardGate</i>	complex<double>
<i>indexNumber</i>	int

6.3.1.27 showPhaseShiftQuantumGate()

```
void showPhaseShiftQuantumGate (
    complex< double > ** phaseShiftGate,
    const int gateSize )
```

Used to show all elements of PHASE SHIFT quantum gate

Parameters

<i>phaseShiftGate</i>	complex<double>
<i>gateSize</i>	const int

6.3.1.28 showQuantumGate()

```
void showQuantumGate (
    complex< double > ** quantumGate,
    const int gateSize )
```

Used to show all elements of quantum gate

Parameters

<i>quantumGate</i>	complex<double>
<i>gateSize</i>	const int

6.3.2 Variable Documentation

6.3.2.1 ONE_ARGUMENT_GATE_SIZE

```
const int ONE_ARGUMENT_GATE_SIZE = 2
```

Parameters

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

6.3.2.2 THREE_ARGUMENTS_GATE_SIZE

```
const int THREE_ARGUMENTS_GATE_SIZE = 8
```

Parameters

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

6.3.2.3 TWO_ARGUMENTS_GATE_SIZE

```
const int TWO_ARGUMENTS_GATE_SIZE = 4
```

Parameters

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

6.4 /home/sebastian/Projects/CLionProjects/QuantumGates/quantum/headers/quantumGateOperation.h File Reference

```
#include <complex>
```

Functions

- `complex< double > ** composeQuantumGates` (`complex< double > **firstGate`, `complex< double > **secondGate`, `int gateSize`)
- `complex< double > ** getIdentityMatrix` (`int gateSize`)
- `bool isIdentityMatrixAndComposedGatesAreEqual` (`complex< double > **identityMatrix`, `complex< double > **composedGates`, `int gateSize`)
- `bool isComposeOfGatesGivesIdentityMatrix` (`complex< double > **firstGate`, `complex< double > **secondGate`, `int gateSize`)
- `bool isMatrixUnitary` (`complex< double > **quantumGate`, `complex< double > **conjugateTransposedQuantumGate`, `int gateSize`)

6.4.1 Function Documentation

6.4.1.1 `composeQuantumGates()`

```
complex<double>** composeQuantumGates (
    complex< double > ** firstGate,
    complex< double > ** secondGate,
    int gateSize )
```

Used to compose two quantum gates.

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

Parameters

<i>firstGate</i>	<code>complex<double></code>
<i>secondGate</i>	<code>complex<double></code>
<i>gateSize</i>	<code>int</code>

Returns

composed quantum gates

6.4.1.2 `getIdentityMatrix()`

```
complex<double>** 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

6.4.1.3 isComposeOfGatesGivesIdentityMatrix()

```
bool isComposeOfGatesGivesIdentityMatrix (
    complex< double > ** firstGate,
    complex< double > ** secondGate,
    int gateSize )
```

Function used to check
if composed gates given identity matrix.

Parameters

<i>firstGate</i>	complex<double>
<i>secondGate</i>	complex<double>
<i>gateSize</i>	int

Returns

true or false

6.4.1.4 isIdentityMatrixAndComposedGatesAreEqual()

```
bool isIdentityMatrixAndComposedGatesAreEqual (
    complex< double > ** identityMatrix,
    complex< double > ** composedGates,
    int gateSize )
```

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

Parameters

<i>identityMatrix</i>	complex<double>
<i>composedGates</i>	complex<double>
<i>gateSize</i>	int

Returns

true or false

6.4.1.5 isMatrixUnitary()

```
bool isMatrixUnitary (
    complex< double > ** quantumGate,
    complex< double > ** conjugateTransposedQuantumGate,
    int gateSize )
```

Function used to check if matrix is unitary.

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

Parameters

<i>quantumGate</i>	complex<double>
<i>conjugateTransposedQuantumGate</i>	complex<double>
<i>gateSize</i>	int

Returns

true or false

6.5 /home/sebastian/Projects/CLionProjects/Quantum↵ Gates/quantum/headers/qubitOperation.h File Reference

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

Functions

- complex< double > ** [getAllocatedQubit](#) (int rows)
- complex< double > ** [makeDotProductOfQubits](#) (complex< double > **firstQubit, complex< double > **secondQubit, int rows, int columns)
- complex< double > ** [getQubitRepresentation](#) (vector< double > baseVector)
- void [showQubit](#) (complex< double > **qubit, const int qubitRows)
- void [showQubitAfterConjugateTranspose](#) (complex< double > **qubit, const int qubitRows)
- void [showDotProduct](#) (complex< double > **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

6.5.1 Function Documentation

6.5.1.1 getAllocatedQubit()

```
complex<double>** getAllocatedQubit (
    int rows )
```

Used to get allocated qubit

Parameters

<i>rows</i>	int
-------------	-----

Returns

allocated qubit

6.5.1.2 getQubitRepresentation()

```
complex<double>** getQubitRepresentation (
    vector< double > baseVector )
```

Used to get qubit as complex type 2D array

Parameters

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

Returns

qubit

6.5.1.3 makeDotProductOfQubits()

```
complex<double>** makeDotProductOfQubits (
    complex< double > ** firstQubit,
    complex< double > ** secondQubit,
    int rows,
    int columns )
```

Used to make dot product of two qubits.

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

Parameters

<i>firstQubit</i>	<code>complex<double></code>
<i>secondQubit</i>	<code>complex<double></code>
<i>rows</i>	<code>int</code>
<i>columns</i>	<code>int</code>

Returns

dot product of qubits

6.5.1.4 showDotProduct()

```
void showDotProduct (
    complex< double > ** dotProduct )
```

Used to show dot product

Parameters

<i>dotProduct</i>	<code>complex<double></code>
-------------------	------------------------------------

6.5.1.5 showQubit()

```
void showQubit (
    complex< double > ** qubit,
    const int qubitRows )
```

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

Parameters

<i>qubit</i>	<code>complex<double></code>
<i>qubitRows</i>	<code>const int</code>

6.5.1.6 showQubitAfterConjugateTranspose()

```
void showQubitAfterConjugateTranspose (
    complex< double > ** qubit,
    const int qubitRows )
```

Used to show qubit after conjugate transpose (reversed columns and rows number)

Parameters

<i>qubit</i>	complex<double>
<i>qubitRows</i>	const int

6.5.2 Variable Documentation

6.5.2.1 QUBIT_NUMBER_OF_COLUMNS

```
const int QUBIT_NUMBER_OF_COLUMNS = 1
```

Parameters

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

6.5.2.2 SINGLE_QUBIT_NUMBER_OF_ROWS

```
const int SINGLE_QUBIT_NUMBER_OF_ROWS = 2
```

Parameters

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

6.5.2.3 THREE_QUBITS_NUMBER_OF_ROWS

```
const int THREE_QUBITS_NUMBER_OF_ROWS = 8
```

Parameters

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

6.5.2.4 TWO_QUBITS_NUMBER_OF_ROWS

```
const int TWO_QUBITS_NUMBER_OF_ROWS = 4
```

Parameters

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

Index

```

/home/sebastian/Projects/CLionProjects/QuantumGates/quantumGateOperation.h,
13 getRandomHermitianMatrix
/home/sebastian/Projects/CLionProjects/QuantumGates/quantumMatrixOperation.h,
15 makeCnotOnQubit
/home/sebastian/Projects/CLionProjects/QuantumGates/quantumGate.h,
15 getSqrtNotGate
/home/sebastian/Projects/CLionProjects/QuantumGates/quantumGate.h,
15 getSwapGate
/home/sebastian/Projects/CLionProjects/QuantumGates/quantumGate.h,
26 getToffoliGate
/home/sebastian/Projects/CLionProjects/QuantumGates/quantumGate.h,
28 makeCnotOnQubit
isComposeOfGatesGivesIdentityMatrix
quantumGateOperation.h, 27
baseVector
quantum::QuantumComputer, 12
isIdentityMatrixAndComposedGatesAreEqual
baseVectorsCount
quantumGateOperation.h, 27
quantum::QuantumComputer, 12
isMatrixUnitary
composeQuantumGates
quantumGateOperation.h, 26
quantumGateOperation.h, 28
countNonZeroBaseVector
quantum::QuantumComputer, 10
isMeasured
quantum::QuantumComputer, 12
isNormalize
quantum::QuantumComputer, 12
getAllocatedMatrix
matrixOperation.h, 13
makeCnotOnQubit
getAllocatedQuantumGate
quantumGate.h, 16
quantumGate.h, 19
makeConjugateTranspose
getAllocatedQubit
qubitOperation.h, 29
matrixOperation.h, 14
getBaseVector
quantum::QuantumComputer, 10
makeDotProductOfQubits
qubitOperation.h, 29
getCnotGate
quantumGate.h, 16
makeFredkinOnQubit
quantumGate.h, 20
getFredkinGate
quantumGate.h, 16
makeHadamardOnQubit
quantumGate.h, 20
getHadamardGate
quantumGate.h, 17
makeMultidimensionalHadamardOnQubit
quantumGate.h, 20
getIdentityMatrix
quantumGateOperation.h, 26
makeNotOnQubit
quantumGate.h, 21
getMultidimensionalHadamardGate
quantumGate.h, 17
makePauliXOnQubit
quantumGate.h, 21
getNotGate
quantumGate.h, 17
makePauliYOnQubit
quantumGate.h, 22
getPauliXGate
quantumGate.h, 17
makePauliZOnQubit
quantumGate.h, 22
getPauliYGate
quantumGate.h, 18
makeSqrtNotOnQubit
quantumGate.h, 23
getPauliZGate
quantumGate.h, 18
makeSwapOnQubit
quantumGate.h, 23
getPhaseShiftGate
quantumGate.h, 18
makeToffoliOnQubit
quantumGate.h, 23
getQubitRepresentation
matrixOperation.h

```

- getAllocatedMatrix, 13
- getRandomHermitianMatrix, 14
- makeConjugateTranspose, 14
- showMatrix, 14
- measure
 - quantum::QuantumComputer, 10
- normalizeRegister
 - quantum::QuantumComputer, 11
- ONE_ARGUMENT_GATE_SIZE
 - quantumGate.h, 25
- quantum, 7
- quantum::QuantumComputer, 9
 - baseVector, 12
 - baseVectorsCount, 12
 - countNonZeroBaseVector, 10
 - getBaseVector, 10
 - isMeasured, 12
 - isNormalize, 12
 - measure, 10
 - normalizeRegister, 11
 - QuantumComputer, 10
 - registerSize, 12
 - resetState, 11
 - validateArraySize, 11
 - validateProbability, 11
 - viewProbability, 11
 - viewQubitsInMathExpression, 12
- QuantumComputer
 - quantum::QuantumComputer, 10
- quantumGate.h
 - getAllocatedQuantumGate, 16
 - getCnotGate, 16
 - getFredkinGate, 16
 - getHadamardGate, 17
 - getMultidimensionalHadamardGate, 17
 - getNotGate, 17
 - getPauliXGate, 17
 - getPauliYGate, 18
 - getPauliZGate, 18
 - getPhaseShiftGate, 18
 - getSqrtNotGate, 19
 - getSwapGate, 19
 - getToffoliGate, 19
 - makeCnotOnQubit, 19
 - makeFredkinOnQubit, 20
 - makeHadamardOnQubit, 20
 - makeMultidimensionalHadamardOnQubit, 20
 - makeNotOnQubit, 21
 - makePauliXOnQubit, 21
 - makePauliYOnQubit, 22
 - makePauliZOnQubit, 22
 - makePhaseShiftOnQubit, 22
 - makeSqrtNotOnQubit, 23
 - makeSwapOnQubit, 23
 - makeToffoliOnQubit, 23
 - ONE_ARGUMENT_GATE_SIZE, 25
 - showMultidimensionalHadamardGate, 24
 - showPhaseShiftQuantumGate, 24
 - showQuantumGate, 24
 - THREE_ARGUMENTS_GATE_SIZE, 25
 - TWO_ARGUMENTS_GATE_SIZE, 25
- quantumGateOperation.h
 - composeQuantumGates, 26
 - getIdentityMatrix, 26
 - isComposeOfGatesGivesIdentityMatrix, 27
 - isIdentityMatrixAndComposedGatesAreEqual, 27
 - isMatrixUnitary, 28
- QUBIT_NUMBER_OF_COLUMNS
 - qubitOperation.h, 31
- qubitOperation.h
 - getAllocatedQubit, 29
 - getQubitRepresentation, 29
 - makeDotProductOfQubits, 29
 - QUBIT_NUMBER_OF_COLUMNS, 31
 - showDotProduct, 30
 - showQubit, 30
 - showQubitAfterConjugateTranspose, 30
 - SINGLE_QUBIT_NUMBER_OF_ROWS, 31
 - THREE_QUBITS_NUMBER_OF_ROWS, 31
 - TWO_QUBITS_NUMBER_OF_ROWS, 31
- registerSize
 - quantum::QuantumComputer, 12
- resetState
 - quantum::QuantumComputer, 11
- showDotProduct
 - qubitOperation.h, 30
- showMatrix
 - matrixOperation.h, 14
- showMultidimensionalHadamardGate
 - quantumGate.h, 24
- showPhaseShiftQuantumGate
 - quantumGate.h, 24
- showQuantumGate
 - quantumGate.h, 24
- showQubit
 - qubitOperation.h, 30
- showQubitAfterConjugateTranspose
 - qubitOperation.h, 30
- SINGLE_QUBIT_NUMBER_OF_ROWS
 - qubitOperation.h, 31
- THREE_ARGUMENTS_GATE_SIZE
 - quantumGate.h, 25
- THREE_QUBITS_NUMBER_OF_ROWS
 - qubitOperation.h, 31
- TWO_ARGUMENTS_GATE_SIZE
 - quantumGate.h, 25
- TWO_QUBITS_NUMBER_OF_ROWS
 - qubitOperation.h, 31
- validateArraySize
 - quantum::QuantumComputer, 11
- validateProbability

- [quantum::QuantumComputer](#), [11](#)
- [viewProbability](#)
 - [quantum::QuantumComputer](#), [11](#)
- [viewQubitsInMathExpression](#)
 - [quantum::QuantumComputer](#), [12](#)