QuantumGates

Generated by Doxygen 1.8.17

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 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

registerSize	int
probabilities	double[]
arraysize	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

arraySize	int
registerSize	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>

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/Quantum↩ Gates/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

firstDimension	int
secondDimension	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

dimension	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

matrix	complex<double>
rows	int
columns	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

matrix	complex<double>
dimension	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 numberOfQubits, 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

dimension	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

indexNumber	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

angle	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

qubit	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

qubit	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

qubit	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

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

Returns

updated qubit

6.3.1.18 makeNotOnQubit()

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

Used to make NOT quantum gate on qubit

Parameters

qubit	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

qubit	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

qubit	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

qubit	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

qubit	complex<double>
angle	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

qubit	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

qubit	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

qubit	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

hadamardGate	complex<double>
indexNumber	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

phaseShiftGate	complex<double>
gateSize	const int

6.3.1.28 showQuantumGate()

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

Used to show all elements of quantum gate

Parameters

quantumGate	complex<double>
gateSize	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/Quantum↩ Gates/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 > **conjugateTransposed↩ QuantumGate, 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

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

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

gateSize	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

firstGate	complex<double>
secondGate	complex<double>
gateSize	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

identityMatrix	complex<double>
composedGates	complex<double>
gateSize	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

quantumGate	complex<double>
conjugateTransposedQuantumGate	complex<double>
gateSize	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

rows	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

baseVector	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

firstQubit	complex<double>
secondQubit	complex<double>
rows	int
columns	int

Returns

dot product of qubits

6.5.1.4 showDotProduct()

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

Used to show dot product

Parameters

dotProduct	complex<double>

6.5.1.5 showQubit()

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

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

Parameters

qubit	complex<double>
qubitRows	const int

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

qubit	complex<double>
qubitRows	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