Graphpp

1.0

Generated by Doxygen 1.9.1

Chapter 1

Graph++ Documentation

1.1 Introduction

Graph++ is a C++ project that provides a comprehensive toolkit for graph analysis and exploration. The project consists of three main components: a library, a graphical user interface (GUI) application, and a unit test project. Its primary aim is to facilitate graph-related tasks such as calculating vertex degrees, chromatic numbers, Hamiltonian and Eulerian paths, and other properties. Graph++ offers a valuable platform for researchers, students, and graph enthusiasts to analyze and explore graphs effectively. Its library, GUI application, and unit test project work together to provide an instrument for graph analysis, making it a useful resource for various graph-related tasks.

1.1.1 Graph++ Library

The Graph++ library serves as the foundation of the project, offering efficient implementations of various graph data structures and algorithms. It provides developers with a straightforward API to create and modify graphs. Additionally, the library includes algorithms for graph analysis, enabling users to gain insights into the structure of their graphs.

1.1.2 Graph++ GUI App

The GUI application complements the library by providing a user-friendly interface for interacting with graphs. It allows users to create, edit, and visualize graphs, applying algorithms and obtaining visual feedback on graph properties. The application includes features such as graph editing tools, customizable layouts, and interactive displays of graph properties.

1.2 Usage

```
// Create some vertices
int vertices[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
// Create a graph for those vertices
Graph<int>* graph = new Graph<int>();
// Add the vertices to the graph
for(int i : vertices){
    graph->addVertex(&i);
}
// Add some edges to the graph (Circular graph)
for(int i : vertices){
    int nextIndex = i + 1;
    if(nextIndex >= nbVertices){
        nextIndex = 0;
    }
    graph->addDoubleEdge(&vertices[i], &vertices[nextIndex]);
}
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Class Documentation

4.1 BasicGraphTest Class Reference

This class tests the different properties of a circular graph of size 10.

Inheritance diagram for BasicGraphTest:

d0/ddc/classBasicGraphTest-eps-converted-to.pdf

4.1.1 Detailed Description

This class tests the different properties of a circular graph of size 10.

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

- sources/Graphpp/Tests/BasicGraphTest/tst_basicgraphtest.cpp

4.2 ComplexGraphTest Class Reference

This class tests the different properties of a complete graph of size 10.

Inheritance diagram for ComplexGraphTest:

d7/da9/classComplexGraphTest-eps-converted-to.pdf

4.2.1 Detailed Description

This class tests the different properties of a complete graph of size 10.

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

- sources/Graphpp/Tests/ComplexGraphTest/tst_complexgraphtest.cpp

4.3 Edge< T > Class Template Reference

Represents a graph edge as a member of an adjacency list and allows to handle the edges of a graph.

```
#include <edge.h>
```

Public Member Functions

- Edge (T ∗target, int weight=1)

  A simple edge constructor.
- T ∗ getTarget ()

  Returns the target vertex of the edge.
- int getWeight ()

  Returns the weight of the edge.
- void setWeight (int weight)

  Sets the weight of the edge.

4.3.1 Detailed Description

template<typename T>
class Edge< T >

Represents a graph edge as a member of an adjacency list and allows to handle the edges of a graph.

Author

The Graph++ Development Team

Date

spring 2023

An Edge is represented as a combination of a target vertex and a weight. The source vertex is not included as the edge belongs to the adjacency list corresponding to the source vertex.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Edge()

```
template<typename T >
Edge< T >::Edge (
            T ∗ target,
            int weight = 1 )
```

A simple edge constructor.

Parameters

target	The target vertex
weight	The weight of the edge

Author

Damien Tschan

Date

17.04.2023

4.3.3 Member Function Documentation

4.3.3.1 getTarget()

```
template<typename T >
T * Edge< T >::getTarget
```

Returns the target vertex of the edge.

Returns

The target vertex of the edge

Author

Damien Tschan

Date

17.04.2023

4.3.3.2 getWeight()

```
template<typename T >
int Edge< T >::getWeight
```

Returns the weight of the edge.

Returns

The weight of the edge

Author

Damien Tschan

Date

17.04.2023

4.3.3.3 setWeight()

```
template<typename T >
void Edge< T >::setWeight (
             int weight )
```

Sets the weight of the edge.

Parameters

weight	The weight to set

Author

Damien Tschan

Date

17.04.2023

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

- sources/Graphpp/Lib/edge.h

4.4 Graph< T > Class Template Reference

Represents a mathematical graph and allows to handle the creation, modification and analysis of it.

```
#include <graph.h>
```

Public Member Functions

- Graph ()

  Initializes a new graph.
- ∼Graph ()

  Deletes the current graph AND ALL THE VERTICES/EDGES CONTAINED.
- void addVertex (T ∗vertex)

  Adds a vertex to the graph.
- void addEdge (T ∗source, T ∗target, int weight=1)

  Adds an edge between two vertices.
- void addDoubleEdge (T ∗vertex1, T ∗vertex2, int weight=1)

  Adds two edges in opposite directions between two vertices.
- void addPrebuiltEdge (T ∗source, Edge< T > ∗)

  Adds a prebuilt edge to the graph.
- void removeVertex (T ∗vertex)

  Removes a vertex and its linked edges from the graph and deletes them.
- std::list< Edge< T > ∗ > popVertex (T ∗vertex)

  Removes a vertex and its linked edges from the graph but doesn't delete anything.

- void removeEdge (Edge< T > *edge)

  Removes an edge from the graph and deletes it completely.
- void popEdge (Edge< T > *edge)

  Removes an edge from the graph but doesn't delete it.
- bool isEmpty ()

  Returns whether the graph is empty.
- bool isEulerian ()

  Returns whether the graph is eulerian.
- bool isHamiltonian ()

  Returns whether the graph is hamiltonian.
- bool isConnected ()

  Returns whether the graph is connected.
- bool isStronglyConnected ()

  Returns whether the graph is strongly connected.
- bool isOriented ()

  Returns whether the graph is oriented.
- bool isWeighted ()

  Returns whether the graph is weighted.
- int getChromaticNumber ()

  Returns an estimation of the chromatic number of the graph as an integer.
- int getNbEdges ()

  Returns the amount of edges in the graph as an integer.
- int getNbVertices ()

  Returns the amount of vertices in the graph as an integer.
- int getVertexIndegree (T *vertex)

  Returns the indegree of a vertex.
- int getVertexOutdegree (T *vertex)

  Returns the outdegree of a vertex.
- Graph< T > * getMinimumSpanningTree ()

  Returns a new graph which is a minimum spanning tree of the initial graph.
- Graph< T > * getMinimumDistanceGraph (T *startingVertex)

  Returns a new graph which is the shortest paths graph of the initial graph.
- Graph< T > * getHamiltonianPath ()

  Returns the hamiltonian path of a graph.
- std::string exportToDOT ()

  Serializes a graph into the DOT format.

Public Attributes

- std::unordered_map< T *, std::list< Edge< T > * > > adjacencyList

  Represents a graph as an adjacency list.

Friends

- template<typename T2 >
  std::ostream & **operator<<** (std::ostream &os, const Graph< T2 > &p)
- template<typename T2 >
  std::istream & **operator>>** (std::istream &is, Graph< T2 > &p)

4.4.1 Detailed Description

template<typename T>
class Graph< T >

Represents a mathematical graph and allows to handle the creation, modification and analysis of it.

Author

The Graph++ Development Team

Date

spring 2023

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Graph()

```
template<typename T >
Graph< T >::Graph
```

Initializes a new graph.

Author

Damien Tschan

Date

17.04.2023

4.4.2.2 ∼Graph()

```
template<typename T >
Graph< T >::∼Graph
```

Deletes the current graph AND ALL THE VERTICES/EDGES CONTAINED.

Author

Jonas Flückiger

Date

16.05.2023

4.4.3 Member Function Documentation

4.4.3.1 addDoubleEdge()

```
template<typename T >
void Graph< T >::addDoubleEdge (
             T * vertex1,
             T * vertex2,
             int weight = 1 )
```

Adds two edges in opposite directions between two vertices.

Parameters

vertex1	A vertex
vertex2	Another vertex
weight	The weight of the edge

Author

Damien Tschan

Date

17.04.2023

4.4.3.2 addEdge()

```
template<typename T >
void Graph< T >::addEdge (
             T * source,
             T * target,
             int weight = 1 )
```

Adds an edge between two vertices.

Parameters

source	Source vertex
target	Target vertex
weight	The weight of the edge

Author

Damien Tschan

Date

17.04.2023

4.4.3.3 addPrebuiltEdge()

```
template<typename T >
void Graph< T >::addPrebuiltEdge (
             T * source,
             Edge< T > * edge )
```

Adds a prebuilt edge to the graph.

Parameters

The	source vertex
The	prebuilt edge

Author

Jonas Flückiger

Date

15.05.2023

4.4.3.4 addVertex()

```
template<typename T >
void Graph< T >::addVertex (
             T * vertex )
```

Adds a vertex to the graph.

Parameters

vertex	A vertex

Author

Damien Tschan

Date

17.04.2023

4.4.3.5 exportToDOT()

```
template<typename T >
std::string Graph< T >::exportToDOT
```

Serializes a graph into the DOT format.

Author

Jonas Flückiger

Date

26.05.2023

4.4.3.6 getChromaticNumber()

```
template<typename T >
int Graph< T >::getChromaticNumber
```

Returns an estimation of the chromatic number of the graph as an integer.

Author

Jonas Flückiger

Date

12.05.2023

This method uses the greedy algorithm, where each vertex is colored one after the other with the first possible color. This method can perform well depending on the order in which the vertices are colored, and depends on the shape of the graph. In this implementation, vertices are colored in order of descending degree.

4.4.3.7 getHamiltonianPath()

```
template<typename T >
Graph< T > * Graph< T >::getHamiltonianPath
```

Returns the hamiltonian path of a graph.

Returns

A subgraph containing the hamiltonian path if it exists, an empty graph otherwise

Author

Damien Tschan

Date

09.06.2023

Since the algorithm and deduction rules are oriented toward directed graphs, each Edge is counted twice. To prevent that, when an Edge is added to the partial path, the opposite Edge is deleted. This means that this method has to be updated to fully support oriented graphs.

As not all rules are implemented, it is almost a brute force algorithm, with a complexity of O(n!)

4.4.3.8 getMinimumDistanceGraph()

```
template<typename T >
Graph< T > * Graph< T >::getMinimumDistanceGraph (
             T * startingVertex )
```

Returns a new graph which is the shortest paths graph of the initial graph.

This method uses Dijkstra's algorithm.

Author

Jonas Flückiger

Date

26.05.2023

4.4.3.9 getMinimumSpanningTree()

```
template<typename T >
Graph< T > * Graph< T >::getMinimumSpanningTree
```

Returns a new graph which is a minimum spanning tree of the initial graph.

This method uses Prim's algorithm.

Author

Jonas Flückiger

Date

15.05.2023

4.4.3.10 getNbEdges()

```
template<typename T >
int Graph< T >::getNbEdges
```

Returns the amount of edges in the graph as an integer.

Returns

The amount of edges in the graph as an integer

Author

Damien Tschan

Date

17.04.2023

4.4.3.11 getNbVertices()

```
template<typename T >
int Graph< T >::getNbVertices
```

Returns the amount of vertices in the graph as an integer.

Returns

The amount of vertices in the graph as an integer

Author

Damien Tschan

Date

17.04.2023

4.4.3.12 getVertexIndegree()

```
template<typename T >
int Graph< T >::getVertexIndegree (
             T * vertex )
```

Returns the indegree of a vertex.

Parameters

vertex	A vertex

Returns

The indegree of the vertex

Author

Damien Tschan

Date

17.04.2023

4.4.3.13 getVertexOutdegree()

```
template<typename T >
int Graph< T >::getVertexOutdegree (
             T * vertex )
```

Returns the outdegree of a vertex.

Parameters

vertex	A vertex

Returns

The outdegree of the vertex

Author

Damien Tschan

Date

17.04.2023

4.4.3.14 isConnected()

```
template<typename T >
bool Graph< T >::isConnected
```

Returns whether the graph is connected.

Returns

Whether the graph is connected

Author

Damien Tschan

Date

08.05.2023

4.4.3.15 isEmpty()

```
template<typename T >
bool Graph< T >::isEmpty
```

Returns whether the graph is empty.

Returns

Whether the graph is empty

Author

Damien Tschan

Date

05.06.2023

4.4.3.16 isEulerian()

```
template<typename T >
bool Graph< T >::isEulerian
```

Returns whether the graph is eulerian.

Returns

Whether the graph is eulerian

Author

Damien Tschan

Date

08.05.2023

4.4.3.17 isHamiltonian()

```
template<typename T >
bool Graph< T >::isHamiltonian
```

Returns whether the graph is hamiltonian.

Returns

Whether the graph is hamiltonian

Author

Damien Tschan

Date

05.06.2023

4.4.3.18 isOriented()

```
template<typename T >
bool Graph< T >::isOriented
```

Returns whether the graph is oriented.

Returns

Whether the graph is oriented

Author

Damien Tschan

Date

24.04.2023

4.4.3.19 isStronglyConnected()

```
template<typename T >
bool Graph< T >::isStronglyConnected
```

Returns whether the graph is strongly connected.

Returns

Whether the graph is strongly connected

Author

Damien Tschan

Date

24.04.2023

4.4.3.20 isWeighted()

```
template<typename T >
bool Graph< T >::isWeighted
```

Returns whether the graph is weighted.

Returns

Whether the graph is weighted

Author

Damien Tschan

Date

24.04.2023

4.4.3.21 popEdge()

```
template<typename T >
void Graph< T >::popEdge (
             Edge< T > * edge )
```

Removes an edge from the graph but doesn't delete it.

Parameters

edge	An edge

Author

Damien Tschan

Date

01.06.2023

As the targeted edge is a parameter of the function, it doesn't need to be returned (the caller already knows it)

4.4.3.22 popVertex()

```
template<typename T >
std::list< Edge< T > * > Graph< T >::popVertex (
             T * vertex )
```

Removes a vertex and its linked edges from the graph but doesn't delete anything.

Parameters

vertex	A vertex

Author

Damien Tschan

Date

01.06.2023

Returns

a list of all removed edges (removed from the graph but not deleted)

As the targeted vertex is a parameter of the function, it doesn't need to be returned (the caller already knows it)

4.4.3.23 removeEdge()

```
template<typename T >
void Graph< T >::removeEdge (
             Edge< T > * edge )
```

Removes an edge from the graph and deletes it completely.

Parameters

edge	An edge

Author

Damien Tschan

Date

01.06.2023

4.4.3.24 removeVertex()

```
template<typename T >
void Graph< T >::removeVertex (
             T * vertex )
```

Removes a vertex and its linked edges from the graph and deletes them.

Parameters

vertex	A vertex

Author

Damien Tschan

Date

01.06.2023

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

- sources/Graphpp/Lib/graph.h

4.5 GraphDockWidget Class Reference

QWidget displaying all informations about graph analysis. Like eulerian, oriented,...

```
#include <graphdockwidget.h>
```

Inheritance diagram for GraphDockWidget:

de/d4b/classGraphDockWidget-eps-converted-to.pdf

Public Member Functions

- void setSelectedGraph (Graph< QVertex > ∗graph)
  Set the selected graph.
- GraphDockWidget (QWidget ∗parent)
  Constructor of the dock widget about graph analysis.

4.5.1 Detailed Description

QWidget displaying all informations about graph analysis. Like eulerian, oriented,...

Author

Plumey Simon

Date

spring 2023

4.5.2 Constructor & Destructor Documentation

4.5.2.1 GraphDockWidget()

```
GraphDockWidget::GraphDockWidget (
            QWidget * parent )
```

Constructor of the dock widget about graph analysis.

Parameters

QWidget	parent

Author

Plumey Simon

4.5.3 Member Function Documentation

4.5.3.1 setSelectedGraph()

```
void GraphDockWidget::setSelectedGraph (
            Graph< QVertex > * graph )
```

Set the selected graph.

Parameters

Graph<QVertex>∗	graph of QVertex

Author

Plumey Simon

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

- sources/Graphpp/App/graphdockwidget.h
- sources/Graphpp/App/graphdockwidget.cpp

4.6 MainWindow Class Reference

MainWindow of the application. It contains a MDI, menu, tools,...

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:

d6/d1a/classMainWindow-eps-converted-to.pdf

Public Member Functions

- MainWindow ()

    Constructor main window.

4.6.1 Detailed Description

MainWindow of the application. It contains a MDI, menu, tools,...

Author

Plumey Simon & Jonas Flückiger

Date

spring 2023

4.6.2 Constructor & Destructor Documentation

4.6.2.1 MainWindow()

```
MainWindow::MainWindow ( )
```

Constructor main window.

Author

Plumey Simon

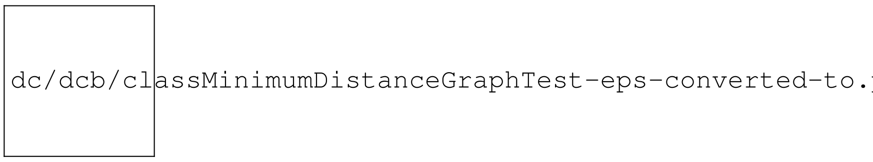
The documentation for this class was generated from the following files:

- sources/Graphpp/App/mainwindow.h
- sources/Graphpp/App/mainwindow.cpp

4.7 MinimumDistanceGraphTest Class Reference

This class tests the minimum distance graph algorithm.

Inheritance diagram for MinimumDistanceGraphTest:



4.7.1 Detailed Description

This class tests the minimum distance graph algorithm.

The tested graph is the complete graph K10, where each edge's weight is equal to the index of the source vertex multiplied by the index of the target vertex (indices range from 1 to 10). Meaning that the minimum distance graph of this graph, starting from any vertex, is composed only of the edges connected to the '1' vertex.

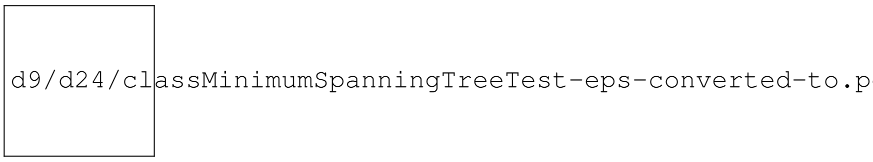
The documentation for this class was generated from the following file:

- sources/Graphpp/Tests/MinimumDistanceGraphTest/tst_minimumdistancegraphtest.cpp

4.8 MinimumSpanningTreeTest Class Reference

This class tests the minimum spanning tree algorithm.

Inheritance diagram for MinimumSpanningTreeTest:



4.8.1 Detailed Description

This class tests the minimum spanning tree algorithm.

The tested graph is the complete graph K10, where each edge's weight is equal to the index of the source vertex multiplied by the index of the target vertex (indices range from 1 to 10), meaning that the minimum spanning tree is composed only of the edges connected to the '1' vertex.

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

- sources/Graphpp/Tests/MinimumSpanningTreeTest/tst_minimumspanningtreetest.cpp

4.9 QBoard Class Reference

QBoard is a widget on which we paint the graph. Like a whiteboard.

```
#include <qboard.h>
```

Inheritance diagram for QBoard:

Public Member Functions

- QBoard (VertexDockWidget ∗vertexDockWidget, QWidget ∗parent=nullptr)

  Constructor of the QBoard.
- virtual ∼QBoard ()

  Destructor of the QBoard.
- void setSelectedTool (Tool selectedTool)

  Set the selected Tool.
- void exportToPng (QString path)

  Able to convert QPainter to a PNG image.
- void exportToDOT (QString path)

  Exports the graph of the board in a DOT text file.
- void saveToFile (QString path)

  Save current graph to a file.
- void openFile (QString path)

  Imports the graph contained in the file.
- void highlightMinimumDistanceGraph ()

  Highlights the current graph's minimum distance graph.
- void highlightMinimumSpanningTree ()

  Highlights the current graph's minimum spanning tree.
- void highlightHamiltonianPath ()

  Highlights the current graph's hamiltonian path.
- QMemento save ()

  Function to save current QBoard state and return a new QMemento.
- void restore (QMemento memento)

  Function to restore QBoard state from a QMemento.
- QCaretaker ∗ getQCaretaker ()

  Return the caretaker.

Public Attributes

- Graph< QVertex > ∗ **graph**
- Graph< QVertex > ∗ **highlightedGraph**

Protected Member Functions

- void paintEvent (QPaintEvent ∗) override

  Paint event method. Called on every graph update. This method is not supposed to be called. Use this->update() to refresh scene.
- void mousePressEvent (QMouseEvent ∗event) override

  Method to handle the right behaviour on click depending on which tool is selected.
- void mouseMoveEvent (QMouseEvent ∗event) override

  Method to handle the right behaviour on mouse moving depending on which tool is selected.
- void mouseReleaseEvent (QMouseEvent ∗event) override

  Method to handle the right behaviour on release click depending on which tool is selected.
- void wheelEvent (QWheelEvent ∗event) override

  Zoom when mouse wheel is used.

4.9.1 Detailed Description

QBoard is a widget on which we paint the graph. Like a whiteboard.

Author

Plumey Simon & Jonas Flückiger

Date

spring 2023

4.9.2 Constructor & Destructor Documentation

4.9.2.1 QBoard()

```
QBoard::QBoard (
            VertexDockWidget * vertexDockWidget,
            QWidget * parent = nullptr )
```

Constructor of the QBoard.

Parameters

QWidget	parent

Author

Plumey Simon

4.9.2.2 ∼QBoard()

```
QBoard::∼QBoard ( )  [virtual]
```

Destructor of the QBoard.

Author

Plumey Simon

4.9.3 Member Function Documentation

4.9.3.1 exportToDOT()

```
void QBoard::exportToDOT (
             QString path )
```

Exports the graph of the board in a DOT text file.

Author

Flückiger Jonas

4.9.3.2 exportToPng()

```
void QBoard::exportToPng (
             QString path )
```

Able to convert QPainter to a PNG image.

Author

Plumey Simon

4.9.3.3 getQCaretaker()

```
QCaretaker * QBoard::getQCaretaker ( )
```

Return the caretaker.

Author

Plumey Simon

4.9.3.4 highlightHamiltonianPath()

```
void QBoard::highlightHamiltonianPath ( )
```

Highlights the current graph's hamiltonian path.

Author

Tschan Damien

4.9.3.5 highlightMinimumDistanceGraph()

```
void QBoard::highlightMinimumDistanceGraph ( )
```

Highlights the current graph's minimum distance graph.

Author

Flückiger Jonas

4.9.3.6 highlightMinimumSpanningTree()

```
void QBoard::highlightMinimumSpanningTree ( )
```

Highlights the current graph's minimum spanning tree.

Author

Flückiger Jonas

4.9.3.7 mouseMoveEvent()

```
void QBoard::mouseMoveEvent (
            QMouseEvent * event )  [override], [protected]
```

Method to handle the right behaviour on mouse moving depending on which tool is selected.

Parameters

QMouseEvent	Mouse event

Author

Plumey Simon

4.9.3.8 mousePressEvent()

```
void QBoard::mousePressEvent (
            QMouseEvent * event )  [override], [protected]
```

Method to handle the right behaviour on click depending on which tool is selected.

Parameters

QMouseEvent	Mouse event

Author

Plumey Simon

4.9.3.9 mouseReleaseEvent()

```
void QBoard::mouseReleaseEvent (
            QMouseEvent * event )  [override], [protected]
```

Method to handle the right behaviour on release click depending on which tool is selected.

Parameters

QMouseEvent	Mouse event

Author

Plumey Simon

4.9.3.10 openFile()

```
void QBoard::openFile (
            QString path )
```

Imports the graph contained in the file.

Parameters

path	the path to the file

Author

Flückiger Jonas

4.9.3.11 paintEvent()

```
void QBoard::paintEvent (
            QPaintEvent * )  [override], [protected]
```

Paint event method. Called on every graph update. This method is not supposed to be called. Use this->update() to refresh scene.

Parameters

QPaintEvent∗	

Author

Plumey Simon

4.9.3.12 restore()

```
void QBoard::restore (
            QMemento memento )
```

Function to restore QBoard state from a QMemento.

Parameters

QMemento	used to restore state of QBoard

Author

Plumey Simon

4.9.3.13 save()

```
QMemento QBoard::save ( )
```

Function to save current QBoard state and return a new QMemento.

Author

Plumey Simon

4.9.3.14 saveToFile()

```
void QBoard::saveToFile (
            QString path )
```

Save current graph to a file.

Parameters

path	the path to the file

Author

Flückiger Jonas

4.9.3.15 setSelectedTool()

```
void QBoard::setSelectedTool (
            Tool selectedTool )
```

Set the selected Tool.

Parameters

Tool	Selected

Author

Plumey Simon

4.9.3.16 wheelEvent()

```
void QBoard::wheelEvent (
            QWheelEvent * event )  [override], [protected]
```

Zoom when mouse wheel is used.

Parameters

QWheelEvent	Mouse event

Author

Plumey Simon

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

- sources/Graphpp/App/qboard.h
- sources/Graphpp/App/clickBehaviours.cpp
- sources/Graphpp/App/moveBehaviours.cpp
- sources/Graphpp/App/qboard.cpp

4.10 QCaretaker Class Reference

Caretaker of qboard state. It's part of the memento design pattern.

`#include <qcaretaker.h>`

Inheritance diagram for QCaretaker:

`d4/d02/classQCaretaker-eps-converted-to.pdf`

Signals

- void **backupAction** ()

Public Member Functions

- QCaretaker (QBoard ∗qboard)

    Constructor of QCaretaker.
- void backup ()

    Save current stat of graph in a QMemento object.
- void undo ()

    Restore the current state from the previous memento.
- void redo ()

    Redo the last operation undid.
- bool canUndo ()
- bool canRedo ()
- void deleteDifferences (std::unordered_map< QVertex ∗, std::list< Edge< QVertex > ∗ >> graphMap, std↩ ::unordered_map< QVertex ∗, std::list< Edge< QVertex > ∗ >> mementoMap)

    Delete in memory all objects present in mapToRemove and not in currentMap.

4.10.1 Detailed Description

Caretaker of qboard state. It's part of the memento design pattern.

Author

Plumey Simon

Date

spring 2023

4.10.2 Constructor & Destructor Documentation

4.10.2.1 QCaretaker()

```
QCaretaker::QCaretaker (
            QBoard * qboard )
```

Constructor of QCaretaker.

Author

Plumey Simon

4.10.3 Member Function Documentation

4.10.3.1 backup()

```
void QCaretaker::backup ( )
```

Save current stat of graph in a QMemento object.

Author

Plumey Simon

4.10.3.2 canRedo()

```
bool QCaretaker::canRedo ( )
```

Returns

true if the caretaker can redo something. Otherwise, return false

Author

Plumey Simon

4.10.3.3 canUndo()

```
bool QCaretaker::canUndo ( )
```

Returns

true if the caretaker can undo something. Otherwise, return false

Author

Plumey Simon

4.10.3.4 deleteDifferences()

```
void QCaretaker::deleteDifferences (
            std::unordered_map< QVertex *, std::list< Edge< QVertex > * >> mapToRemove,
            std::unordered_map< QVertex *, std::list< Edge< QVertex > * >> currentMap )
```

Delete in memory all objects present in mapToRemove and not in currentMap.

Parameters

std::unordered_map<QVertex∗,std::list<Edge<↩ QVertex>∗>>	mapToRemove map with all object more will be deleted (often memento map)
std::unordered_map<QVertex∗,std::list<Edge<↩ QVertex>∗>>	currentMap the reference map to compare the mapToRemove (often current graph map)

Author

Plumey Simon

4.10.3.5 redo()

```
void QCaretaker::redo ( )
```

Redo the last operation undid.

Author

Plumey Simon

4.10.3.6 undo()

```
void QCaretaker::undo ( )
```

Restore the current state from the previous memento.

Author

Plumey Simon

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

- sources/Graphpp/App/qcaretaker.h
- sources/Graphpp/App/qcaretaker.cpp

4.11 QMemento Class Reference

Memento of qboard state. It's part of the memento design pattern.

```
#include <qmemento.h>
```

Public Member Functions

- QMemento ()

  Default constructor for data structs.
- QMemento (std::unordered_map< QVertex ∗, std::list< Edge< QVertex > ∗ >> adjencyList)

  Constructor of QMemento.
- std::unordered_map< QVertex ∗, std::list< Edge< QVertex > ∗ > > getAdjencyList ()

  Getter of attribut adjencyList.

4.11.1 Detailed Description

Memento of qboard state. It's part of the memento design pattern.

Author

Plumey Simon

Date

spring 2023

4.11.2 Constructor & Destructor Documentation

4.11.2.1 QMemento() [1/2]

```
QMemento::QMemento ( )
```

Default constructor for data structs.

Author

Flückiger Jonas

4.11.2.2 QMemento() [2/2]

```
QMemento::QMemento (
            std::unordered_map< QVertex ∗, std::list< Edge< QVertex > ∗ >> adjencyList )
```

Constructor of QMemento.

Author

Plumey Simon

4.11.3 Member Function Documentation

4.11.3.1 getAdjencyList()

```
std::unordered_map< QVertex *, std::list< Edge< QVertex > * > > QMemento::getAdjencyList ( )
```

Getter of attribut adjencyList.

Returns

std::unordered_map<QVertex *, std::list<Edge<QVertex> *>>

Author

Plumey Simon

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

- sources/Graphpp/App/qmemento.h
- sources/Graphpp/App/qmemento.cpp

4.12 QMultipleInputDialog Class Reference

This class create a multiple input dialog. It can manage inputs from a list of spinBox.

```
#include <qmultipleinputdialog.h>
```

Inheritance diagram for QMultipleInputDialog:

d2/d05/classQMultipleInputDialog-eps-converted-to.pdf

Public Member Functions

- QMultipleInputDialog (QString title, QList< QPair< QLabel *, QSpinBox * >> elements, QWidget *parent=nullptr)

  Constructor of QMultipleInputDialog.

Static Public Member Functions

- static QList< int > getInts (QString title, QList< QPair< QLabel *, QSpinBox * >> elements, bool *ok, QWidget *parent=nullptr)

  Get all spinbox values and put in a list.

4.12.1 Detailed Description

This class create a multiple input dialog. It can manage inputs from a list of spinBox.

Author

Plumey Simon, inspired by Bobur (https://stackoverflow.com/a/53332748)

Date

spring 2023

4.12.2 Constructor & Destructor Documentation

4.12.2.1 QMultipleInputDialog()

```
QMultipleInputDialog::QMultipleInputDialog (
            QString title,
            QList< QPair< QLabel *, QSpinBox * >> elements,
            QWidget * parent = nullptr )  [explicit]
```

Constructor of QMultipleInputDialog.

Parameters

QString	title
QList<QPair<QLabel,QSpinBox*>>*	list of pair of labels and spinbox
QWidget	parent

Author

Plumey Simon, inspired by Bobur

4.12.3 Member Function Documentation

4.12.3.1 getInts()

```
QList< int > QMultipleInputDialog::getInts (
            QString title,
            QList< QPair< QLabel *, QSpinBox * >> elements,
            bool * ok,
            QWidget * parent = nullptr )  [static]
```

Get all spinbox values and put in a list.

Author

Plumey Simon, inspired by Bobur

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

- sources/Graphpp/App/qmultipleinputdialog.h
- sources/Graphpp/App/qmultipleinputdialog.cpp

4.13 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_↩ MainWindow_t Struct Reference

Public Attributes

- uint **offsetsAndSizes** [40]
- char **stringdata0** [11]
- char **stringdata1** [9]
- char **stringdata2** [1]
- char **stringdata3** [10]
- char **stringdata4** [10]
- char **stringdata5** [12]
- char **stringdata6** [12]
- char **stringdata7** [5]
- char **stringdata8** [5]
- char **stringdata9** [6]
- char **stringdata10** [30]
- char **stringdata11** [29]
- char **stringdata12** [22]
- char **stringdata13** [19]
- char **stringdata14** [9]
- char **stringdata15** [7]
- char **stringdata16** [19]
- char **stringdata17** [15]
- char **stringdata18** [5]
- char **stringdata19** [5]

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

- sources/build-Graphpp-Desktop_Qt_6_4_2_MinGW_64_bit-Release/App/release/moc_mainwindow.cpp

4.14 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_↩ QMultipleInputDialog_t Struct Reference

Public Attributes

- uint **offsetsAndSizes** [2]
- char **stringdata0** [21]

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

- sources/build-Graphpp-Desktop_Qt_6_4_2_MinGW_64_bit-Release/App/release/moc_qmultipleinputdialog.↩ cpp

4.15 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_↩ SelectColorButton_t Struct Reference

Public Attributes

- uint **offsetsAndSizes** [10]
- char **stringdata0** [18]
- char **stringdata1** [13]
- char **stringdata2** [1]
- char **stringdata3** [12]
- char **stringdata4** [12]

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

- sources/build-Graphpp-Desktop_Qt_6_4_2_MinGW_64_bit-Release/App/release/moc_selectcolorbutton.↩ cpp

4.16 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_↩ VertexDockWidget_t Struct Reference

Public Attributes

- uint **offsetsAndSizes** [6]
- char **stringdata0** [17]
- char **stringdata1** [14]
- char **stringdata2** [1]

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

- sources/build-Graphpp-Desktop_Qt_6_4_2_MinGW_64_bit-Release/App/release/moc_vertexdockwidget.↩ cpp

4.17 queue_element< T > Struct Template Reference

A structure to hold vertices in a priority queue.

```
#include <queue_element.h>
```

Public Member Functions

- queue_element (int priority, T ∗source, Edge< T > ∗edge)

  Builds a new queue element.

Public Attributes

- int priority

    The priority of this element in the queue.
- T ∗ source

    The vertex from which the element was discovered.
- Edge< T > ∗ edge

    The edge connecting the source and the element.

4.17.1 Detailed Description

template<typename T>
struct queue_element< T >

A structure to hold vertices in a priority queue.

This structure is used for different graph search algorithms where vertices and the way (source and egde) through which they were discovered must be saved in a priority queue.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 queue_element()

```
template<typename T >
queue_element< T >::queue_element (
             int priority,
             T ∗ source,
             Edge< T > ∗ edge )  [inline]
```

Builds a new queue element.

Parameters

The	priority of this element in the queue.
The	vertex from which the element was discovered.
The	edge connecting the source and the element.

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

- sources/Graphpp/Lib/queue_element.h

4.18 QVertex Class Reference

Represent a vertex in the graph.

```
#include <qvertex.h>
```

Public Member Functions

- QVertex (QString name, QPointF position, QColor textColor=Qt::black, QColor backgroundColor=Qt::black, QColor borderColor=Qt::black)

    Constructor of QVertex.
- QString **getName** ()
- QPointF **getPosition** ()
- QColor **getTextColor** ()
- QColor **getBackgroundColor** ()
- QColor **getBorderColor** ()
- bool **isSelected** ()
- void **setName** (QString name)
- void **setPosition** (QPointF position)
- void **setTextColor** (QColor color)
- void **setBackgroundColor** (QColor color)
- void **setBorderColor** (QColor color)
- void **setSelected** (bool selected)

4.18.1 Detailed Description

Represent a vertex in the graph.

Author

Plumey Simon

Date

spring 2023

4.18.2 Constructor & Destructor Documentation

4.18.2.1 QVertex()

```
QVertex::QVertex (
            QString name,
            QPointF position,
            QColor textColor = Qt::black,
            QColor backgroundColor = Qt::black,
            QColor borderColor = Qt::black )
```

Constructor of QVertex.

Parameters

QString	name
QPointF	position
QColor	color of text
QColor	color of background
QColor	color of border

Author

Plumey Simon, inspired by Bobur

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

- sources/Graphpp/App/qvertex.h
- sources/Graphpp/App/qvertex.cpp

4.19 SelectColorButton Class Reference

A simple button to select color This code comes from here: `https://stackoverflow.com/questions/18257281/qt-`

`#include <selectcolorbutton.h>`

Inheritance diagram for SelectColorButton:

`df/d6c/classSelectColorButton-eps-converted-to.pdf`

Public Slots

- void **updateColor** ()
- void **changeColor** ()

Signals

- void **colorChanged** ()

Public Member Functions

- **SelectColorButton** (QWidget ∗parent)
- void **setColor** (const QColor &color)
- const QColor & **getColor** () const

4.19.1 Detailed Description

A simple button to select color This code comes from here: `https://stackoverflow.com/questions/18257281/qt-`

Author

Alexis Wilke

Date

spring 2023

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

- sources/Graphpp/App/selectcolorbutton.h
- sources/build-Graphpp-Desktop_Qt_6_4_2_MinGW_64_bit-Release/App/release/moc_selectcolorbutton.↩ cpp
- sources/Graphpp/App/selectcolorbutton.cpp

4.20 VertexDockWidget Class Reference

QWidget displaying all informations about the selected vertex. Like colors, positions,...

```
#include <vertexdockwidget.h>
```

Inheritance diagram for VertexDockWidget:

df/d86/classVertexDockWidget-eps-converted-to.pdf

Signals

- void **vertexUpdated** ()

Public Member Functions

- void setSelectedVertex (QVertex ∗vertex)
  Set the selected vertex.
- void setSelectedGraph (Graph< QVertex > ∗graph)
  Set the selected graph (graph is used to know the degree of the selected vertex)
- VertexDockWidget (QWidget ∗parent)
  Constructor of the dock widget about vertex properties.

4.20.1 Detailed Description

QWidget displaying all informations about the selected vertex. Like colors, positions,...

Author

Plumey Simon

Date

spring 2023

4.20.2 Constructor & Destructor Documentation

4.20.2.1 VertexDockWidget()

```
VertexDockWidget::VertexDockWidget (
            QWidget * parent )
```

Constructor of the dock widget about vertex properties.

Parameters

QWidget	parent

Author

Plumey Simon

4.20.3 Member Function Documentation

4.20.3.1 setSelectedGraph()

```
void VertexDockWidget::setSelectedGraph (
            Graph< QVertex > * graph )
```

Set the selected graph (graph is used to know the degree of the selected vertex)

Parameters

Graph<QVertex>∗	graph of QVertex

Author

Plumey Simon

4.20.3.2 setSelectedVertex()

```
void VertexDockWidget::setSelectedVertex (
            QVertex * vertex )
```

Set the selected vertex.

Parameters

QVertex∗	vertex

Author

Plumey Simon

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

- sources/Graphpp/App/vertexdockwidget.h

- sources/build-Graphpp-Desktop_Qt_6_4_2_MinGW_64_bit-Release/App/release/moc_vertexdockwidget.←
cpp
- sources/Graphpp/App/vertexdockwidget.cpp