BitcoinAnalyzerwithC++

Generated by Doxygen 1.8.17

Chapter 1

BitcoinAnalyzer with Qt Creator and C++

A program coded with QT Creator and C++ in my spare time. The program fetches bitcoin price data from Goincecko API https://www.coingecko.com/en/api/documentation and calculates 3 queries based on the fetched data. The GET request functionality is made by SSL-protocol to read, parse, clean and transform the raw data into usable format.

The three queries that the user can run (as of now) are the following:

A) The longest bearish (downward) trend in days within the given date range.

B) Date (in format dd.mm.yyyy) with the highest trading volume within the given date range.

C) Best days to sell, buy or hold bitcoin to maximize profits withing the given date range.

1.1 Doumentation

- PDF-documentation file in repository root (bitcoinAnalyzerDocumentation.pfd) OR
- Doxygen documentation (HTML) in ./DoxygenDocumentation/html/index.html

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 bitcoinAnalyzer Namespace Reference

Are the classes of this program are in the bitcoinAnalyzer (main program) namespace.

Classes

- class mainUI

  The mainUI class is the class that contains all the functionality of the main menu.

5.1.1 Detailed Description

Are the classes of this program are in the bitcoinAnalyzer (main program) namespace.

Chapter 6

Class Documentation

6.1 bitcoinAnalyzer::mainUI Class Reference

The mainUI class is the class that contains all the functionality of the main menu.

`#include <mainui.h>`

Inherits QMainWindow.

Classes

- struct times

Public Member Functions

- **mainUI** (QWidget *parent=nullptr)
- void initializeGUI ()

  initializeGUI function initializes the mainMenu GUI. For example it sets the start date to 1.3.2020 0.00 and the end date to 1.8.2021 0.00 so that we can test our programs first query in a convenient way.
- void setTimes ()

  setTimes sets the times given by the user in the mainMenu and stores them into private attributes.
- std::string unixTimeToHumanReadable (long int seconds, bool showTime=false)

  unixTimeToHumanReadable converts unix timestamp to human readable UCT-time string.
- std::map< double, double > readData (QJsonArray array)

  readData reads the raw data from the API into QJsonArray.
- void loadData ()

  loadData loads the parsed and cleaned data into pricesMap_, totalVolumesMap_ and marketCapsmap_.
- void clearCachedData ()

  clearCachedData clears the cached data between queries from all of the maps and arrays (private attributes) to avoid data corruption. We want to start from a clean slate in every new query made by the user.
- void arrayElemsToArray (QJsonArray jsonArray)

  arrayElemsToArray converts all the elements of the QJsonArray (given as a parameter) to Arrays.
- std::pair< double, double > findHighestEntry (std::map< double, double > targetMap)

  findHighestEntry finds and returns the key, value -pair that contains the highest value of the whole map given as a parameter.

- std::pair< double, double > findLowestEntry (std::map< double, double > targetMap)

  findLowestEntry finds and returns the key, value -pair that contains the lowest value of the whole map given as a parameter.
- void calculateLongestBearTrend ()

  calculateLongestBearTrend calculates/finds the longest BTC bear (downward) trend from the given date range and updates the result to the GUI instantly.
- void findHighestVolumeDay ()

  findHighestVolumeDay calculates/finds the highest trading volume day from the given date range and updates the result to the GUI instantly.
- void giveInvestmentRecommendation ()

  giveInvestmentRecommendation calculates/finds the optimal days to sell and buy or HOLD BTC from the given date range and updates the result to the GUI instantly.
- void executeQueries ()

  executeQueries makes function call to the three previosly mentioned query-functions calculateLongestBearTrend(), findHighestVolumeDay() and giveInvestmentRecommendation() thus executing all the three queries and updates the results to the GUI for the user to see.

Public Attributes

- const std::string **FIAT_CURRENCY** = "eur"
- const std::string **CRYPTO_ID** = "bitcoin"
- std::string **REQUEST_URL**
- struct times **timeVars**

Private Slots

- void on_executeButton_clicked ()

  on_executeButton_clicked is a private slot that stores the necessary times in right formats from the user input, calculates the delta of the start and end date and makes get request from the coinGecko API based on the correct URL when the user presses the "Execute" -button in the GUI.
- void on_closeButton_clicked ()

  on_closeButton_clicked closes the main program when the user clicks the "Close" -button in the GUI.
- void onResult (QNetworkReply *reply)

  onResult is a private slot to which we connect to in the mainUI class constructor. It basically listens to QNetwork↩ Reply and if the reply from the GET request is successful, it reads the data from the API to initial placeholder data storages, then calls the parsing and cleaning functions that transform the data into usable format, and finally calls the executeQueries() function to execute all the three queries and show the results to the user.

Private Attributes

- Ui::mainUI * **ui**
- QNetworkAccessManager * **networkManager_**
- QUrl **coingeckoUrl_**
- QDate **startDate_**
- QDate **endDate_**
- unsigned int **daysBetween_**
- int **longestBearTrend_**
- double **highestBitcoinPrice_**
- double **lowestBitcoinPrice_**
- QJsonArray **pricesArray_**
- QJsonArray **marketCapsArray_**
- QJsonArray **totalVolumesArray_**
- std::map< double, double > **pricesMap_**
- std::map< double, double > **marketCapsMap_**
- std::map< double, double > **totalVolumesMap_**
- std::map< QDateTime, double > **uctDatePrices_**
- std::map< double, double >::iterator **pricesIterator_** = pricesMap_.begin()

6.1.1 Detailed Description

The mainUI class is the class that contains all the functionality of the main menu.

6.1.2 Member Function Documentation

6.1.2.1 arrayElemsToArray()

```
void bitcoinAnalyzer::mainUI::arrayElemsToArray (
            QJsonArray jsonArray )
```

arrayElemsToArray converts all the elements of the QJsonArray (given as a parameter) to Arrays.

Parameters

jsonArray	is a QJsonArray (as of now either pricesArray_, totalVolumesArray_ or marketCapsArray_).

6.1.2.2 findHighestEntry()

```
std::pair< double, double > bitcoinAnalyzer::mainUI::findHighestEntry (
            std::map< double, double > targetMap )
```

findHighestEntry finds and returns the key, value -pair that contains the highest value of the whole map given as a parameter.

Parameters

targetMap	is a STL map data structure from which we find the pair with highest entry.

Returns

STL pair that has the UNIX-timestamp as first element and some highest value (e.g. BTC price, market cap or total trading volume) as the second element.

6.1.2.3 findLowestEntry()

```
std::pair< double, double > bitcoinAnalyzer::mainUI::findLowestEntry (
            std::map< double, double > targetMap )
```

findLowestEntry finds and returns the key, value -pair that contains the lowest value of the whole map given as a parameter.

Parameters

targetMap	is a STL map data structure from which we find the pair with lowest entry.

Returns

STL pair that has the UNIX-timestamp as first element and some lowest value (e.g. BTC price, market cap or total trading volume) as the second element.

6.1.2.4 readData()

```
std::map< double, double > bitcoinAnalyzer::mainUI::readData (
            QJsonArray array )
```

readData reads the raw data from the API into QJsonArray.

Parameters

array	is a QJsonArray that contains the desired data which is later transformed to usable data.

Returns

STL map that has the UNIX-timestamp as a key and either BTC price, total volume or market cap as a value.

6.1.2.5 unixTimeToHumanReadable()

```
std::string bitcoinAnalyzer::mainUI::unixTimeToHumanReadable (
            long int seconds,
            bool showTime = false )
```

unixTimeToHumanReadable converts unix timestamp to human readable UCT-time string.

Parameters

amount	of seconds from the UNIX Epoch time (January 1st, 1970 at 00:00:00 UTC)
showTime	is by default false but when set to true, it also outputs the time in the result query in the GUI.

Returns

Time (string) as a human readable UCT-time.

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

- BitcoinAnalyzer/mainui.h
- BitcoinAnalyzer/mainui.cpp

6.2 bitcoinAnalyzer::mainUI::times Struct Reference

Public Attributes

- QDateTime **uctStartTime_**
- QDateTime **uctEndTime_**
- uint **unixStartTime_**
- uint **unixEndTime_**
- std::string **strUnixStartTime_**
- std::string **strUnixEndTime_**

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

- BitcoinAnalyzer/mainui.h