# Trade Strategies with Cpp - Ömer Halit Cinar

Generated by Doxygen 1.10.0

1 Class Index		1
1.1 Class List	 	1
2 File Index		3
2.1 File List	 	3
3 Class Documentation		5
3.1 Strategy Class Reference	 	5
3.1.1 Detailed Description	 	5
3.1.2 Member Function Documentation	 	5
3.1.2.1 calculateBollingerBands()	 	5
3.1.2.2 calculateLongPeriodMovingAverage()	 	6
3.1.2.3 calculateRSI()	 	6
3.1.2.4 calculateShortPeriodMovingAverage()	 	6
3.1.3 Member Data Documentation	 	7
3.1.3.1 longPeriod	 	7
3.1.3.2 shortPeriod	 	7
4 File Documentation		9
4.1 inc/strategy.hpp File Reference		9
4.2 strategy.hpp		9
4.3 inc/utils.hpp File Reference		10
4.3.1 Function Documentation		10
4.3.1.1 bollingerBandsLogger()		10
4.3.1.2 fetchAvailableStocks()		11
4.3.1.3 fetchMarketData()		11
4.3.1.4 getAvailableStocks()		11
4.3.1.5 loadEnv()		11
4.3.1.6 loadMenu()		12
4.3.1.7 movingAverageLogger()		12
		12
4.3.1.8 parseAvailableStocks()		
4.3.1.9 parseMarketData()		12
4.3.1.10 rsiLogger()		13
4.4 utils.hpp		13
4.5 src/main.cpp File Reference		13
4.5.1 Function Documentation		14
4.5.1.1 main()		14
4.6 src/strategy.cpp File Reference		14
4.7 src/utils.cpp File Reference		14
4.7.1 Function Documentation		15
4.7.1.1 bollingerBandsLogger()		15
4.7.1.2 fetchAvailableStocks()	 	15
4.7.1.3 fetchMarketData()	 	15

4.7.1.4 getAvailableStocks()	16
4.7.1.5 loadEnv()	16
4.7.1.6 loadMenu()	16
4.7.1.7 movingAverageLogger()	16
4.7.1.8 parseAvailableStocks()	16
4.7.1.9 parseMarketData()	17
4.7.1.10 rsiLogger()	17
4.7.1.11 WriteCallback()	17

# **Class Index**

## 1.1 Class List

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

## Strategy

Provides vario	us technica	l analysis	functions	for stock	c prices,	such as	moving	averages, RSI,	
and Bollinger I	3ands								5

2 Class Index

# File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

inc/strategy.hpp															 								9
inc/utils.hpp						 								 	 								10
src/main.cpp .						 								 	 								13
src/strategy.cpp															 								14
src/utils con																							14

File Index

## **Class Documentation**

## 3.1 Strategy Class Reference

Provides various technical analysis functions for stock prices, such as moving averages, RSI, and Bollinger Bands.

```
#include <strategy.hpp>
```

#### **Static Public Member Functions**

- static double calculateShortPeriodMovingAverage (const std::vector< double > &prices)
  - Calculates the short-period simple moving average (SMA).
- static double calculateLongPeriodMovingAverage (const std::vector< double > &prices)
  - Calculates the long-period simple moving average (SMA).
- static double calculateRSI (const std::vector< double > &prices, int period=14)
  - Calculates the Relative Strength Index (RSI) over a specified period.
- static std::tuple< double, double, double > calculateBollingerBands (const std::vector< double > &prices, int period=20, double numStdDev=2.0)

Calculates the Bollinger Bands for a given set of prices.

### **Static Public Attributes**

- static const int shortPeriod = 5
  - Number of periods for short period moving average.
- static const int longPeriod = 20

Number of periods for long period moving average.

## 3.1.1 Detailed Description

Provides various technical analysis functions for stock prices, such as moving averages, RSI, and Bollinger Bands.

## 3.1.2 Member Function Documentation

## 3.1.2.1 calculateBollingerBands()

```
std::tuple< double, double, double > Strategy::calculateBollingerBands (
    const std::vector< double > & prices,
    int period = 20,
    double numStdDev = 2.0 ) [static]
```

Calculates the Bollinger Bands for a given set of prices.

6 Class Documentation

#### **Parameters**

prices	Vector of historical stock prices.
period	Number of periods to use for the moving average (default is 20).
numStdDev	Number of standard deviations to calculate the upper and lower bands (default is 2.0).

#### **Returns**

A tuple containing the lower band, middle band (SMA), and upper band.

## 3.1.2.2 calculateLongPeriodMovingAverage()

Calculates the long-period simple moving average (SMA).

#### **Parameters**

prices	Vector of historical stock prices.
--------	------------------------------------

#### Returns

The long-period SMA.

## 3.1.2.3 calculateRSI()

Calculates the Relative Strength Index (RSI) over a specified period.

## **Parameters**

	prices	Vector of historical stock prices.
ſ	period	Number of periods to use for RSI calculation (default is 14).

## Returns

The RSI value.

## 3.1.2.4 calculateShortPeriodMovingAverage()

Calculates the short-period simple moving average (SMA).

## **Parameters**

prices	Vector of historical stock prices.
--------	------------------------------------

## Returns

The short-period SMA.

## 3.1.3 Member Data Documentation

## 3.1.3.1 longPeriod

```
const int Strategy::longPeriod = 20 [static]
```

Number of periods for long period moving average.

#### 3.1.3.2 shortPeriod

```
const int Strategy::shortPeriod = 5 [static]
```

Number of periods for short period moving average.

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

- inc/strategy.hpp
- src/strategy.cpp

8 Class Documentation

## **File Documentation**

## 4.1 inc/strategy.hpp File Reference

```
#include <iostream>
#include <vector>
#include <numeric>
```

#### Classes

· class Strategy

Provides various technical analysis functions for stock prices, such as moving averages, RSI, and Bollinger Bands.

## 4.2 strategy.hpp

#### Go to the documentation of this file.

```
00001 #ifndef STRATEGY_HPP
00002 #define STRATEGY_HPP
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <numeric>
00007
00012 class Strategy
00013 {
00014 public:
00016
        const static int shortPeriod = 5;
00017
         const static int longPeriod = 20;
00020
00026
         static double calculateShortPeriodMovingAverage(const std::vector<double>& prices);
00027
00033
          static double calculateLongPeriodMovingAverage(const std::vector<double>& prices);
00034
00041
          static double calculateRSI(const std::vector<double>& prices, int period = 14);
00050
         static std::tuple<double, double, double> calculateBollingerBands(const std::vector<double>&
     prices, int period = 20, double numStdDev = 2.0);
00051 };
00052
00053 #endif // STRATEGY_HPP
```

## 4.3 inc/utils.hpp File Reference

```
#include <string>
#include <vector>
```

#### **Functions**

• void loadEnv ()

Loads environment variables from the .env file.

std::string fetchMarketData (const std::string &apiKey, const std::string &symbol)

Fetches market data for a given stock symbol using an API.

std::string fetchAvailableStocks (const std::string &apiKey)

Fetches available stocks from the API.

std::vector< double > parseMarketData (const std::string &jsonData)

Parses the market data JSON string and extracts close prices.

· void movingAverageLogger (const double shortSMA, const double longSMA)

Logs buy, sell, or hold signals based on short and long simple moving averages (SMA).

void rsiLogger (const double rsi)

Logs buy, sell, or hold signals based on the Relative Strength Index (RSI).

 void bollingerBandsLogger (const double lowerBand, const double middleBand, const double upperBand, const double lastPrice)

Logs buy, sell, or hold signals based on Bollinger Bands.

• void loadMenu ()

Loads the menu options for the application.

std::vector< std::string > getAvailableStocks ()

Fetches the list of available stocks.

• std::vector< std::string > parseAvailableStocks (const std::string &jsonData)

Parses the JSON response and extracts available stock symbols.

#### 4.3.1 Function Documentation

## 4.3.1.1 bollingerBandsLogger()

Logs buy, sell, or hold signals based on Bollinger Bands.

#### **Parameters**

lowerBand	The lower Bollinger Band.
middleBand	The middle Bollinger Band (SMA).
upperBand	The upper Bollinger Band.
lastPrice	The most recent stock price.

### 4.3.1.2 fetchAvailableStocks()

```
std::string fetchAvailableStocks ( {\tt const\ std::string\ \&\ apiKey\ )}
```

Fetches available stocks from the API.

## **Parameters**

apiKey The API key for authentication.
--

#### **Returns**

The available stocks as a JSON string.

## 4.3.1.3 fetchMarketData()

Fetches market data for a given stock symbol using an API.

#### **Parameters**

apiKey	The API key for authentication.
symbol	The stock symbol to fetch data for.

## Returns

The market data as a JSON string.

## 4.3.1.4 getAvailableStocks()

```
std::vector< std::string > getAvailableStocks ( )
```

Fetches the list of available stocks.

#### Returns

A vector containing stock symbols.

## 4.3.1.5 loadEnv()

```
void loadEnv ( )
```

Loads environment variables from the .env file.

## 4.3.1.6 loadMenu()

```
void loadMenu ( )
```

Loads the menu options for the application.

## 4.3.1.7 movingAverageLogger()

Logs buy, sell, or hold signals based on short and long simple moving averages (SMA).

#### **Parameters**

shortSMA	The short-period SMA.
IongSMA	The long-period SMA.

## 4.3.1.8 parseAvailableStocks()

Parses the JSON response and extracts available stock symbols.

## **Parameters**

jsonData	The JSON data containing stock information.
----------	---

#### Returns

A vector of stock symbols.

## 4.3.1.9 parseMarketData()

Parses the market data JSON string and extracts close prices.

## **Parameters**

jsonData	The JSON data containing stock prices.

4.4 utils.hpp 13

#### Returns

A vector of close prices.

#### 4.3.1.10 rsiLogger()

```
void rsiLogger ( {\tt const\ double}\ rsi\ )
```

Logs buy, sell, or hold signals based on the Relative Strength Index (RSI).

#### **Parameters**

```
rsi The RSI value.
```

## 4.4 utils.hpp

#### Go to the documentation of this file.

```
00001 #ifndef UTILS_HPP
00002 #define UTILS_HPP
00003
00004 #include <string>
00005 #include <vector>
00006
00010 void loadEnv();
00011
00018 std::string fetchMarketData(const std::string& apiKey, const std::string& symbol);
00019
00025 std::string fetchAvailableStocks(const std::string& apiKey);
00032 std::vector<double> parseMarketData(const std::string& jsonData);
00033
00039 void movingAverageLogger(const double shortSMA, const double longSMA);
00040
00045 void rsiLogger(const double rsi);
00054 void bollingerBandsLogger(const double lowerBand, const double middleBand, const double upperBand,
      const double lastPrice);
00055
00059 void loadMenu();
00060
00065 std::vector<std::string> getAvailableStocks();
00072 std::vector<std::string> parseAvailableStocks(const std::string& jsonData);
00073
00074 #endif // UTILS_HPP
```

## 4.5 src/main.cpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include "../inc/utils.hpp"
#include "../inc/strategy.hpp"
```

## **Functions**

• int main ()

#### 4.5.1 Function Documentation

## 4.5.1.1 main()

```
int main ( )
```

## 4.6 src/strategy.cpp File Reference

```
#include "../inc/strategy.hpp"
#include <cmath>
#include <numeric>
```

## 4.7 src/utils.cpp File Reference

```
#include "../inc/utils.hpp"
#include "../inc/strategy.hpp"
#include <iostream>
#include <fstream>
#include <sstream>
#include <curl/curl.h>
#include "../json/json.hpp"
```

### **Functions**

- size t WriteCallback (void \*contents, size t size, size t nmemb, void \*userp)
- void loadEnv ()

Loads environment variables from the .env file.

• std::string fetchMarketData (const std::string &apiKey, const std::string &symbol)

Fetches market data for a given stock symbol using an API.

• std::string fetchAvailableStocks (const std::string &apiKey)

Fetches available stocks from the API.

std::vector< double > parseMarketData (const std::string &jsonData)

Parses the market data JSON string and extracts close prices.

std::vector< std::string > parseAvailableStocks (const std::string &jsonData)

Parses the JSON response and extracts available stock symbols.

std::vector< std::string > getAvailableStocks ()

Fetches the list of available stocks.

• void movingAverageLogger (const double shortSMA, const double longSMA)

Logs buy, sell, or hold signals based on short and long simple moving averages (SMA).

• void rsiLogger (const double rsi)

Logs buy, sell, or hold signals based on the Relative Strength Index (RSI).

 void bollingerBandsLogger (const double lowerBand, const double middleBand, const double upperBand, const double lastPrice)

Logs buy, sell, or hold signals based on Bollinger Bands.

• void loadMenu ()

Loads the menu options for the application.

## 4.7.1 Function Documentation

## 4.7.1.1 bollingerBandsLogger()

Logs buy, sell, or hold signals based on Bollinger Bands.

#### **Parameters**

lowerBand	The lower Bollinger Band.
middleBand	The middle Bollinger Band (SMA).
upperBand	The upper Bollinger Band.
lastPrice	The most recent stock price.

## 4.7.1.2 fetchAvailableStocks()

Fetches available stocks from the API.

#### **Parameters**

apiKey	The API key for authentication.
--------	---------------------------------

## Returns

The available stocks as a JSON string.

## 4.7.1.3 fetchMarketData()

Fetches market data for a given stock symbol using an API.

### **Parameters**

apiKey	The API key for authentication.
symbol	The stock symbol to fetch data for.

#### Returns

The market data as a JSON string.

## 4.7.1.4 getAvailableStocks()

```
std::vector< std::string > getAvailableStocks ( )
```

Fetches the list of available stocks.

## Returns

A vector containing stock symbols.

## 4.7.1.5 loadEnv()

```
void loadEnv ( )
```

Loads environment variables from the .env file.

## 4.7.1.6 loadMenu()

```
void loadMenu ( )
```

Loads the menu options for the application.

#### 4.7.1.7 movingAverageLogger()

Logs buy, sell, or hold signals based on short and long simple moving averages (SMA).

## **Parameters**

shortSMA	The short-period SMA.
IongSMA	The long-period SMA.

## 4.7.1.8 parseAvailableStocks()

Parses the JSON response and extracts available stock symbols.

#### **Parameters**

ponbata   The occit data containing stock information.	jsonData	The JSON data containing stock information.	]
--	----------	---	---

## Returns

A vector of stock symbols.

## 4.7.1.9 parseMarketData()

Parses the market data JSON string and extracts close prices.

#### **Parameters**

ata containing stock prices.	jsonData
------------------------------	----------

#### Returns

A vector of close prices.

## 4.7.1.10 rsiLogger()

Logs buy, sell, or hold signals based on the Relative Strength Index (RSI).

### **Parameters**

```
rsi The RSI value.
```

## 4.7.1.11 WriteCallback()