DOROTHY

Version 1.0

Generated by Doxygen 1.8.18

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 Utilities Namespace Reference

Functions

- **int RoundOff** (double a_value)

  Function to round off value to the lower integral value.
- double **GetAverage** (std::vector< double > a_list)

  Function to get average of a list with doubles.
- double **GetStandardDeviation** (std::vector< double > a_list, double a_average)

  Function to get the standard deviation of a list with doubles.
- void **trimBlanks** (std::string &a_str)

  Method to trim leading and trailing blanks while reading data.

4.1.1 Detailed Description

Utilities.h (p. **??**) Namespace with utility functions for the simulator.

Created by Salil Maharjan on 5/04/20. Copyright © 2020 Salil Maharjan. All rights reserved.

4.1.2 Function Documentation

4.1.2.1 GetAverage()

```
double Utilities::GetAverage (
            std::vector< double > a_list )
```

Function to get average of a list with doubles.

Utilities::GetAverage (p. **??**) Function to get average of a list with doubles.

Parameters

a_list	vector<double> List of doubles

Returns

double The mean of the list.

Author

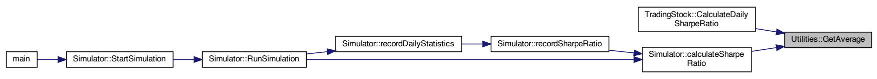
Salil Maharjan

Date

5/04/20.

Definition at line 36 of file Utilities.cpp.

Here is the caller graph for this function:


TradingStock::CalculateDaily SharpeRatio
main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordDailyStatistics
Simulator::recordSharpeRatio
Simulator::calculateSharpe Ratio
Utilities::GetAverage

4.1.2.2 GetStandardDeviation()

```
double Utilities::GetStandardDeviation (
            std::vector< double > a_list,
            double a_average )
```

Function to get the standard deviation of a list with doubles.

Utilities::GetStandardDeviation (p. **??**) Function to get the standard deviation of a list with doubles.

Parameters

a_list	vector<double> List of doubles.
a_average	double The mean of the list.

Returns

double The standard deviation of the list.

Author

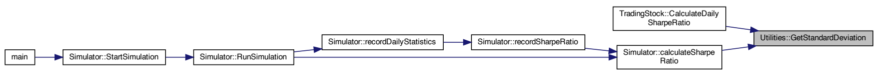
Salil Maharjan

Date

5/04/20.

Definition at line 53 of file Utilities.cpp.

Here is the caller graph for this function:


TradingStock::CalculateDaily SharpeRatio
Utilities::GetStandardDeviation
Simulator::recordDailyStatistics
Simulator::recordSharpeRatio
main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::calculateSharpe Ratio

4.1.2.3 RoundOff()

```
int Utilities::RoundOff (
            double a_value )
```

Function to round off value to the lower integral value.

Utilities.cpp (p. **??**) Implementation of **Utilities.h** (p. **??**).

Created by Salil Maharjan on 5/04/20. Copyright © 2020 Salil Maharjan. All rights reserved. **Utilities::RoundOff** (p. **??**) Function to round off value to integer value. Uses floor.

Parameters

a_value	double Value to round off.

Returns

int Integral value of a_value left rounded.

Author

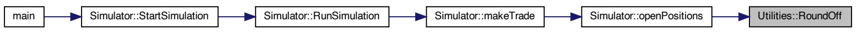
Salil Maharjan

Date

5/04/20.

Definition at line 22 of file Utilities.cpp.

Here is the caller graph for this function:



4.1.2.4 trimBlanks()

```
void Utilities::trimBlanks (
            std::string & a_str )
```

Method to trim leading and trailing blanks while reading data.

Utilities::trimBlanks (p. **??**) Method to trim leading and trailing whitespaces.

Parameters

a_str	string Reference to string to trim white spaces.

Author

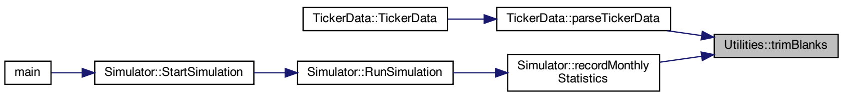
Salil Maharjan

Date

4/30/20.

Definition at line 73 of file Utilities.cpp.

Here is the caller graph for this function:


TickerData::TickerData
TickerData::parseTickerData
Utilities::trimBlanks
main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordMonthly Statistics

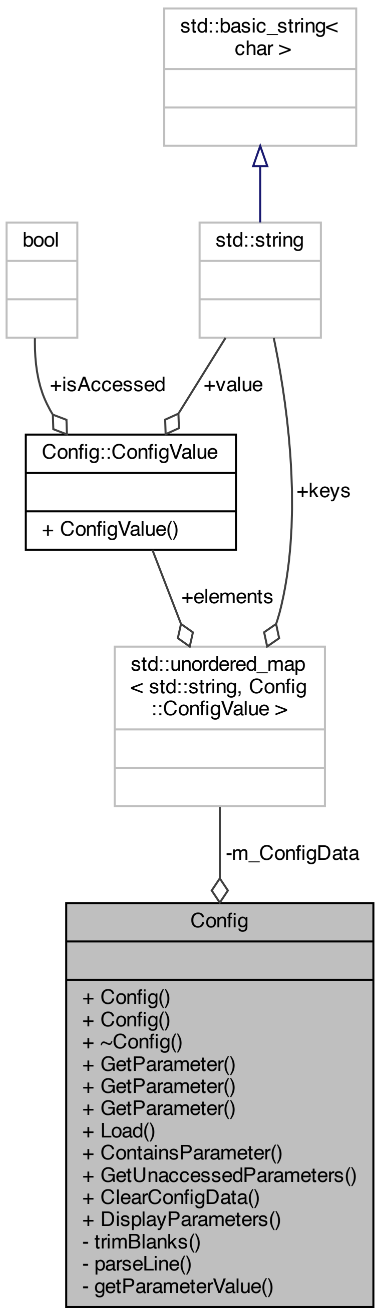
Chapter 5

Class Documentation

5.1 Config Class Reference

```
#include <Config.h>
```

Collaboration diagram for Config:



Classes

- struct **ConfigValue**

  Struct to hold configuration values and access flag.

Public Member Functions

- **Config** ()

  Default constructor.
- **Config** (const **char** ∗a_file)

  Main class constructor.
- virtual ∼**Config** ()

  Virtual deconstructor.
- template<class T >
  bool **GetParameter** (std::string a_parameter, T &a_value)

  Template function to access parameters.
- bool **GetParameter** (std::string a_parameter, **char** ∗a_value)

  Template function specializations to access parameters as char∗ and bool.
- bool **GetParameter** (std::string a_parameter, bool &a_value)
- bool **Load** (const **char** ∗a_file, bool a_displayParameters=true)

  Load in data from a specified configuration file. Allows multiple calls.
- bool **ContainsParameter** (std::string a_parameter)

  Test if a config file has a parameter for a given segment.
- void **GetUnaccessedParameters** (std::vector< std::string > &a_paramNames)

  Provides a list of the parameters that were not accessed.
- void **ClearConfigData** ()

  Clear the set of recorded parameters.
- void **DisplayParameters** ()

  Display the parameters in alphabetical order.

Private Member Functions

- void **trimBlanks** (std::string &a_str)

  Trim leading and trailing blanks.
- bool **parseLine** (const std::string &a_line, std::string &a_name, std::string &a_value)

  Get parameters by parsing line from config file.
- bool **getParameterValue** (std::string a_name, std::string &a_value)

  Get the value of a specified parameter.

Private Attributes

- std::unordered_map< std::string, **ConfigValue** > **m_ConfigData**

  Map of config values.

5.1.1 Detailed Description

Config.h (p. **??**) Interface for the **Config** (p. **??**) class. Loads configurations from file to use for simulation.

Created by Salil Maharjan on 4/25/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 18 of file Config.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Config() [1/2]

```
Config::Config ( )  [inline]
```

Default constructor.

Definition at line 27 of file Config.h.

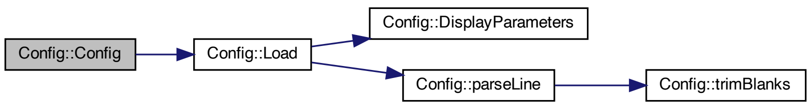
5.1.2.2 Config() [2/2]

```
Config::Config (
             const  char * a_file )  [inline]
```

Main class constructor.

Definition at line 30 of file Config.h.

Here is the call graph for this function:



5.1.2.3 ∼Config()

```
virtual Config::∼Config ( )  [inline], [virtual]
```

Virtual deconstructor.

Definition at line 33 of file Config.h.

5.1.3 Member Function Documentation

5.1.3.1 ClearConfigData()

```
void Config::ClearConfigData ( )  [inline]
```

Clear the set of recorded parameters.

Definition at line 65 of file Config.h.

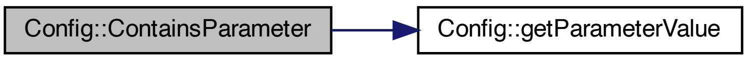
5.1.3.2 ContainsParameter()

```
bool Config::ContainsParameter (
             std::string a_parameter )  [inline]
```

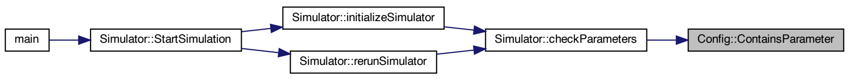
Test if a config file has a parameter for a given segment.

Definition at line 55 of file Config.h.

Here is the call graph for this function:



Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::checkParameters
Config::ContainsParameter

5.1.3.3 DisplayParameters()

```
void Config::DisplayParameters ( )
```

Display the parameters in alphabetical order.

Config::DisplayParameters (p. **??**) Method to display the parameters in alphabetical order.

Author

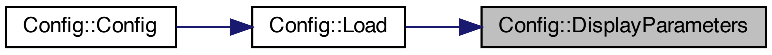
Salil Maharjan

Date

4/30/20.

Definition at line 140 of file Config.cpp.

Here is the caller graph for this function:



5.1.3.4 GetParameter() [1/3]

```
bool Config::GetParameter (
            std::string a_parameter,
            bool & a_value )
```

Config::GetParameter (p. **??**) Template function specialization of function (GetParameter) for returning parameter value as booleans.

Parameters

a_parameter	string Parameter name.
a_value	bool Variable to save parameter value.

Returns

bool Function execution status.

Author

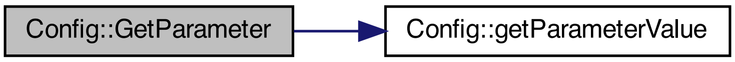
Salil Maharjan

Date

4/30/20.

Definition at line 93 of file Config.cpp.

Here is the call graph for this function:



5.1.3.5 GetParameter() [2/3]

```
bool Config::GetParameter (
             std::string a_parameter,
              char * a_value )
```

Template function specializations to access parameters as char∗ and bool.

Config::GetParameter (p. **??**) Template function specialization of function (GetParameter) for returning parameter value as char∗.

Parameters

a_parameter	string Parameter name.
a_value	char∗ Variable to save parameter value.

Returns

bool Function execution status.

Author

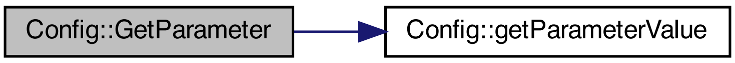
Salil Maharjan

Date

4/30/20.

Definition at line 71 of file Config.cpp.

Here is the call graph for this function:



5.1.3.6 GetParameter() [3/3]

```
template<class T >
bool Config::GetParameter (
            std::string a_parameter,
            T & a_value )
```

Template function to access parameters.

Config::GetParameter (p. **??**) Template function to get parameter value and pass it by reference. Template supports: (string, int, short, long, float, double) Explicit instantiation of template function on header file to prevent linker error. Separate template function specialization functions defined for (bool, char∗)

Parameters

a_parameter	string Parameter name.
a_value	T Variable to save parameter value. Template supported types.

Returns

bool Function execution status.

Author

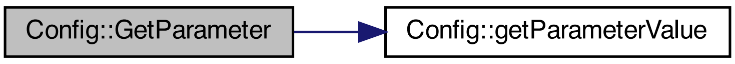
Salil Maharjan

Date

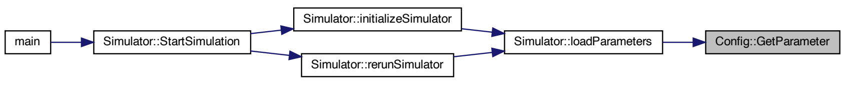
4/30/20.

Definition at line 115 of file Config.h.

Here is the call graph for this function:



Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::loadParameters
Config::GetParameter

5.1.3.7 getParameterValue()

```
bool Config::getParameterValue (
            std::string a_name,
            std::string & a_value )  [private]
```

Get the value of a specified parameter.

Config::getParameterValue (p. **??**) Private utility function to get parameter value as a string. Used by Get↩ Parameter.

Parameters

a_name	string Parameter name.
a_value	bool Parameter value.

Returns

bool Function execution status.

Author

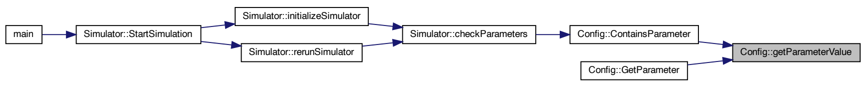
Salil Maharjan

Date

4/30/20.

Definition at line 221 of file Config.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::checkParameters
Config::ContainsParameter
Config::GetParameter
Config::getParameterValue

5.1.3.8 GetUnaccessedParameters()

```
void Config::GetUnaccessedParameters (
            std::vector< std::string > & a_paramNames )
```

Provides a list of the parameters that were not accessed.

Config::GetUnaccessedParameters (p. **??**) Method to provide the list of the parameters that were not accessed.

Parameters

a_paramNames	vector<string> Names of parameters that were not accessed.

Author

Salil Maharjan

Date

4/30/20.

Definition at line 125 of file Config.cpp.

5.1.3.9 Load()

```
bool Config::Load (
            const  char * a_file,
            bool a_displayParameters = true )
```

Load in data from a specified configuration file. Allows multiple calls.

Config.cpp (p. **??**) Implementation of **Config.h** (p. **??**).

Created by Salil Maharjan on 4/25/20. Copyright © 2020 Salil Maharjan. All rights reserved. **Config::Load** (p. **??**) Method to load configuration file. Uses parseLine method to parse lines.

Parameters

a_file	char∗ **Config** (p. **??**) file name.
a_displayParameters	bool Flag for display purposes

Returns

bool Function execution status.

Author

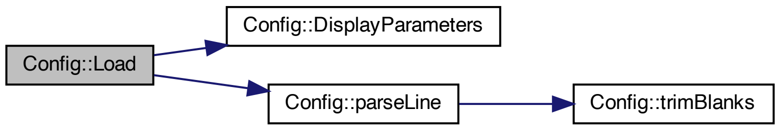
Salil Maharjan

Date

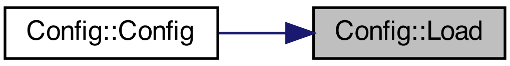
4/30/20.

Definition at line 25 of file Config.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3.10 parseLine()

```
bool Config::parseLine (
             const std::string & a_line,
             std::string & a_name,
             std::string & a_value )  [private]
```

Get parameters by parsing line from config file.

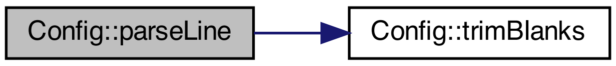
Config::parseLine (p. **??**) Method to parse configuration file line into parameter name and value.

Parameters

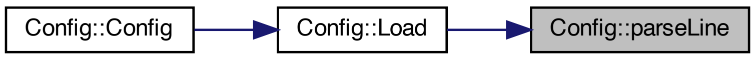
a_line	string The line read from configuration file.
a_name	string Variable to save parameter name by reference.

Definition at line 169 of file Config.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3.11 trimBlanks()

```
void Config::trimBlanks (
             std::string & a_str )  [private]
```

Trim leading and trailing blanks.

Config::trimBlanks (p. **??**) Method to trim leading and trailing whitespaces.

Parameters

a_str	string Reference to string to trim white spaces.

Author

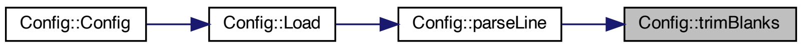
Salil Maharjan

Date

4/30/20.

Definition at line 245 of file Config.cpp.

Here is the caller graph for this function:



5.1.4 Member Data Documentation

5.1.4.1 m_ConfigData

`std::unordered_map<std::string, ConfigValue> Config::m_ConfigData [private]`

Map of config values.

Definition at line 87 of file Config.h.

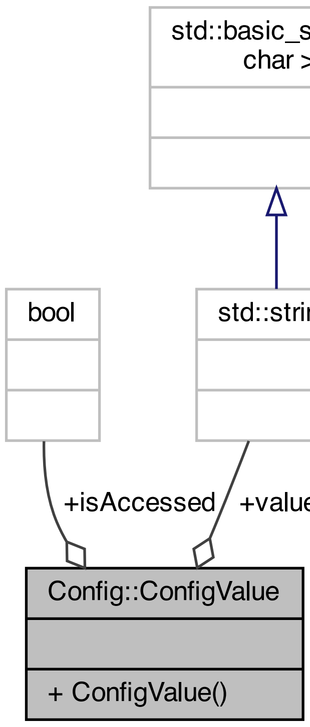
The documentation for this class was generated from the following files:

- **Config.h**
- **Config.cpp**

5.2 Config::ConfigValue Struct Reference

Struct to hold configuration values and access flag.

Collaboration diagram for Config::ConfigValue:



Public Member Functions

- **ConfigValue** ()

Public Attributes

- std::string **value**
- bool **isAccessed**

5.2.1 Detailed Description

Struct to hold configuration values and access flag.

Definition at line 77 of file Config.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 ConfigValue()

```
Config::ConfigValue::ConfigValue ( )  [inline]
```

Definition at line 83 of file Config.h.

5.2.3 Member Data Documentation

5.2.3.1 isAccessed

```
bool Config::ConfigValue::isAccessed
```

Definition at line 81 of file Config.h.

5.2.3.2 value

```
std::string Config::ConfigValue::value
```

Definition at line 79 of file Config.h.

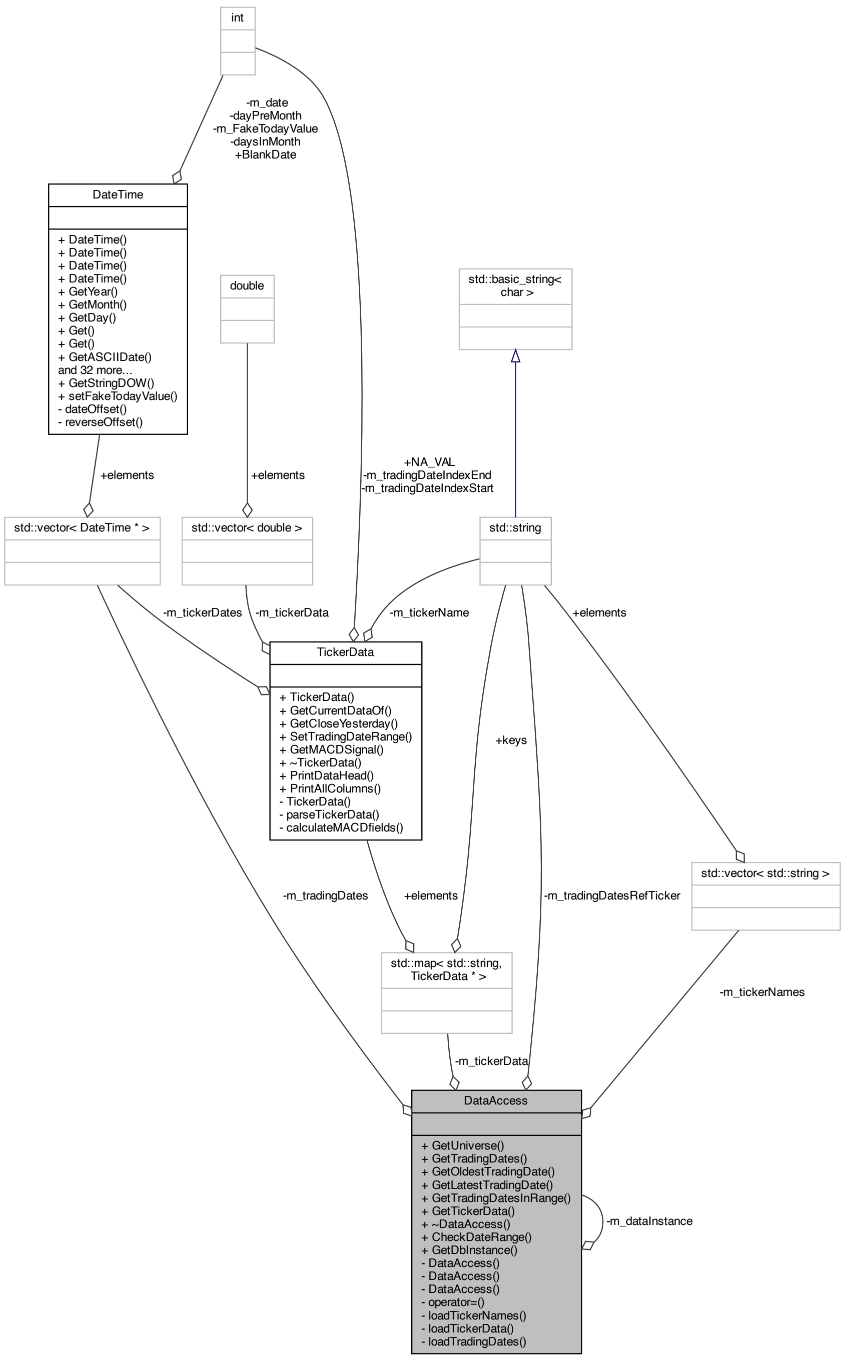
The documentation for this struct was generated from the following file:

- **Config.h**

5.3 DataAccess Class Reference

```
#include <DataAccess.h>
```

Collaboration diagram for DataAccess:



Public Member Functions

- const std::vector< std::string > **GetUniverse** ()

    Get constituent names in the universe.
- const std::vector< **DateTime** $*$ > **GetTradingDates** ()

    Get trading dates.
- **DateTime** $*$ **GetOldestTradingDate** ()

Get oldest trading date for which data is available.
- **DateTime** $*$ **GetLatestTradingDate** ()

  Get most recent trading date for which data is available.
- std::vector$<$ **DateTime** $*$ $>$ **GetTradingDatesInRange** (**DateTime** $*$a_from, **DateTime** $*$a_to)

  Get trading dates from "a_from" to "a_to".
- **TickerData** $*$ **GetTickerData** (std::string a_ticker)

  *Get **TickerData** (*p. **??***) Data.*
- ~**DataAccess** ()=default

  Destructor.
- bool **CheckDateRange** (**DateTime** $*$a_startDate, **DateTime** $*$a_endDate)

  Check date range with available data's date range.

Static Public Member Functions

- static **DataAccess** $*$ **GetDbInstance** (const std::string a_directory, const std::string a_universe)

  *Static access method to create a singleton **DataAccess** (*p. **??***) object.*

Private Member Functions

- **DataAccess** ()

  Private constructors to prevent multiple instancing:
- **DataAccess** (const std::string &a_directory, const std::string &a_universe)

  Main class parameterized constructor.
- **DataAccess** (**DataAccess** const &)

  Private copy constructor.
- **DataAccess** & **operator=** (**DataAccess** const &)=delete

  Delete assignment operator.
- void **loadTickerNames** (const std::string &a_universe)

  Method to load ticker names from the constituent universe file.
- void **loadTickerData** (const std::string &a_directory)

  Method to load price data for loaded tickers.
- void **loadTradingDates** (const std::string &a_directory)

  Method to load trading dates from the refence ticker, IBM.

Static Private Attributes

- static const std::string **m_tradingDatesRefTicker** = "IBM"

  TickerData** (*p. **??) for referencing trading dates. Uses IBM.*
- static **DataAccess** $*$ **m_dataInstance** = 0

  *Single **DataAccess** (*p. **??***) instance var.*
- static std::vector$<$ std::string $>$ **m_tickerNames**

  TickerData** (*p. **??) names specified in the universe.*
- static std::vector$<$ **DateTime** $*$ $>$ **m_tradingDates**

  Trading dates loaded from reference ticker.
- static std::map$<$ std::string, **TickerData** $*$ $>$ **m_tickerData**

  *Map variable (**TickerData** (*p. **??***) Name : **TickerData** (*p. **??***) Data)*

5.3.1 Detailed Description

DataAccess.h (p. **??**) Singleton class for storing data of ticker price data in internal memory. Cannot have multiple instances since it can use a lot of memory.

Created by Salil Maharjan on 4/29/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 18 of file DataAccess.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ∼DataAccess()

```
DataAccess::∼DataAccess ( )  [default]
```

Destructor.

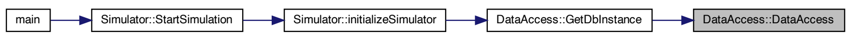
5.3.2.2 DataAccess() [1/3]

```
DataAccess::DataAccess ( )  [inline], [private]
```

Private constructors to prevent multiple instancing:

Definition at line 80 of file DataAccess.h.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
DataAccess::GetDbInstance
DataAccess::DataAccess

5.3.2.3 DataAccess() [2/3]

```
DataAccess::DataAccess (
            const std::string & a_directory,
            const std::string & a_universe )  [private]
```

Main class parameterized constructor.

DataAccess::DataAccess (p. **??**). Constructor to create single instance of **DataAccess** (p. **??**). Stores price data for the specified universe in memory.

Parameters

a_directory	string Price data directory path.
a_universe	string Directory path for ticker name universe file.

Author

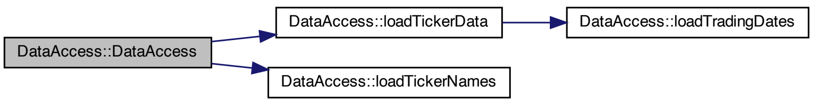
Salil Maharjan

Date

4/29/20.

Definition at line 34 of file DataAccess.cpp.

Here is the call graph for this function:


DataAccess::DataAccess
DataAccess::loadTickerData
DataAccess::loadTradingDates
DataAccess::loadTickerNames

5.3.2.4 DataAccess() [3/3]

```
DataAccess::DataAccess (
              DataAccess const &  )  [inline], [private]
```

Private copy constructor.

Definition at line 85 of file DataAccess.h.

5.3.3 Member Function Documentation

5.3.3.1 CheckDateRange()

```
bool DataAccess::CheckDateRange (
              DateTime * a_startDate,
              DateTime * a_endDate )
```

Check date range with available data's date range.

DataAccess::CheckDateRange (p. **??**). Check date range with available data's date range.

Parameters

a_startDate	DateTime∗ Starting date range.
a_endDate	DateTime∗ End of date range.

Returns

bool If data is available for the date range.

Author

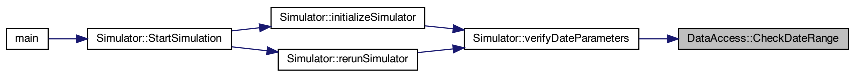
Salil Maharjan

Date

4/29/20.

Definition at line 70 of file DataAccess.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::verifyDateParameters
DataAccess::CheckDateRange

5.3.3.2 GetDbInstance()

```
DataAccess * DataAccess::GetDbInstance (
            const std::string a_directory,
            const std::string a_universe )  [static]
```

Static access method to create a singleton **DataAccess** (p. **??**) object.

DataAccess::GetDbInstance (p. **??**). Method to create a single instance of **DataAccess** (p. **??**) and prevent multiple instances.

Parameters

a_directory	string Price data directory path.
a_universe	string Directory path for file with ticker name universe.

Returns

DataAccess∗ Single instance of the path.

Author

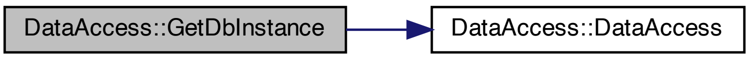
Salil Maharjan

Date

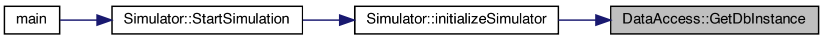
4/29/20.

Definition at line 53 of file DataAccess.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
DataAccess::GetDbInstance

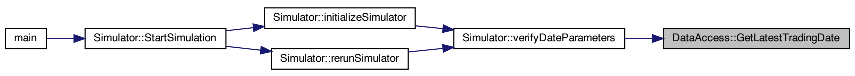
5.3.3.3 GetLatestTradingDate()

```
DateTime* DataAccess::GetLatestTradingDate ( )  [inline]
```

Get most recent trading date for which data is available.

Definition at line 36 of file DataAccess.h.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::verifyDateParameters
DataAccess::GetLatestTradingDate

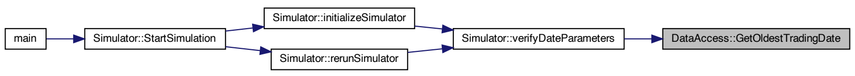
5.3.3.4 GetOldestTradingDate()

```
DateTime* DataAccess::GetOldestTradingDate ( )  [inline]
```

Get oldest trading date for which data is available.

Definition at line 33 of file DataAccess.h.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::verifyDateParameters
DataAccess::GetOldestTradingDate

5.3.3.5 GetTickerData()

```
TickerData * DataAccess::GetTickerData (
            std::string a_ticker )
```

Get **TickerData** (p. **??**) Data.

DataAccess::GetTickerData (p. **??**). Get ticker data for a_ticker.

Parameters

a_ticker	string The name of the ticker to get data.

Returns

TickerData∗ Price data of a_ticker

Author

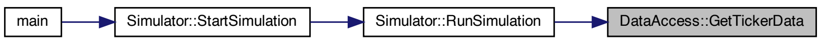
Salil Maharjan

Date

4/29/20.

Definition at line 105 of file DataAccess.cpp.

Here is the caller graph for this function:



5.3.3.6 GetTradingDates()

```
const std::vector< DateTime*> DataAccess::GetTradingDates ( )  [inline]
```

Get trading dates.

Definition at line 30 of file DataAccess.h.

5.3.3.7 GetTradingDatesInRange()

```
std::vector< DateTime * > DataAccess::GetTradingDatesInRange (
              DateTime * a_from,
              DateTime * a_to )
```

Get trading dates from "a_from" to "a_to".

DataAccess::GetTradingDatesInRange (p. **??**). Get trading dates from "a_from" to "a_to" according to reference ticker. IBM.

Parameters

a_from	DateTime∗ Starting date range.
a_to	DateTime∗ End of date range.

Returns

vector<DateTime∗> All trading dates in range.

Author

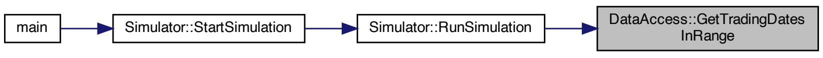
Salil Maharjan

Date

4/29/20.

Definition at line 84 of file DataAccess.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
DataAccess::GetTradingDates InRange

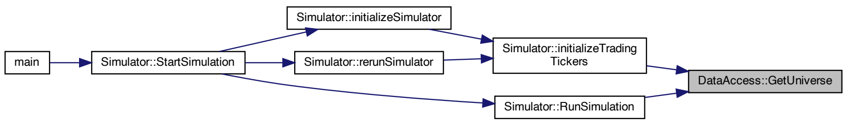
5.3.3.8 GetUniverse()

```
const std::vector<std::string> DataAccess::GetUniverse ( )  [inline]
```

Get constituent names in the universe.

Definition at line 27 of file DataAccess.h.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::RunSimulation
Simulator::initializeTrading Tickers
DataAccess::GetUniverse

5.3.3.9 loadTickerData()

```
void DataAccess::loadTickerData (
             const std::string & a_directory )  [private]
```

Method to load price data for loaded tickers.

DataAccess::loadTickerData (p. **??**). Method to load price data for each ticker in the universe.

Parameters

a_directory	string Price data directory path.

Author

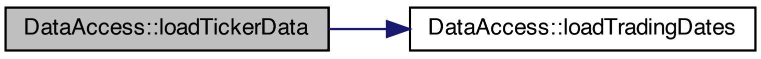
Salil Maharjan

Date

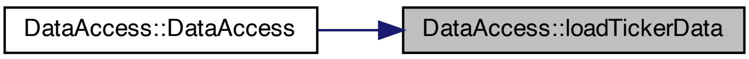
4/29/20.

Definition at line 182 of file DataAccess.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.3.10 loadTickerNames()

```
void DataAccess::loadTickerNames (
             const std::string & a_universe )  [private]
```

Method to load ticker names from the constituent universe file.

DataAccess::loadTickerNames (p. **??**). Method to load ticker names in the universe.

Parameters

a_universe	string Directory path for ticker name universe file.

Author

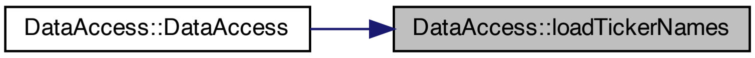
Salil Maharjan

Date

4/29/20.

Definition at line 124 of file DataAccess.cpp.

Here is the caller graph for this function:



5.3.3.11 loadTradingDates()

```
void DataAccess::loadTradingDates (
             const std::string & a_directory )  [private]
```

Method to load trading dates from the refence ticker, IBM.

DataAccess::loadTradingDates (p. **??**). Method to load trading dates from the refence ticker, IBM.

Parameters

a_directory	string Price data directory path.

Author

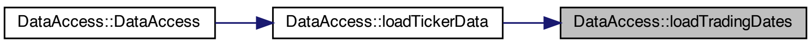
Salil Maharjan

Date

4/29/20.

Definition at line 154 of file DataAccess.cpp.

Here is the caller graph for this function:



5.3.3.12 operator=()

```
 DataAccess& DataAccess::operator= (
             DataAccess const &  )  [private], [delete]
```

Delete assignment operator.

5.3.4 Member Data Documentation

5.3.4.1 m_dataInstance

```
DataAccess * DataAccess::m_dataInstance = 0  [static], [private]
```

Single **DataAccess** (p. **??**) instance var.

DataAccess.cpp (p. **??**) Implementation of **DataAccess.h** (p. **??**)

Created by Salil Maharjan on 4/29/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 67 of file DataAccess.h.

5.3.4.2 m_tickerData

```
std::map< std::string, TickerData * > DataAccess::m_tickerData  [static], [private]
```

Map variable (**TickerData** (p. **??**) Name : **TickerData** (p. **??**) Data)

Definition at line 73 of file DataAccess.h.

5.3.4.3 m_tickerNames

```
std::vector< std::string > DataAccess::m_tickerNames  [static], [private]
```

TickerData (p. **??**) names specified in the universe.

Definition at line 69 of file DataAccess.h.

5.3.4.4 m_tradingDates

```
std::vector< DateTime * > DataAccess::m_tradingDates  [static], [private]
```

Trading dates loaded from reference ticker.

Definition at line 71 of file DataAccess.h.

5.3.4.5 m_tradingDatesRefTicker

```
const std::string DataAccess::m_tradingDatesRefTicker = "IBM"  [static], [private]
```

TickerData (p. **??**) for referencing trading dates. Uses IBM.

Definition at line 65 of file DataAccess.h.

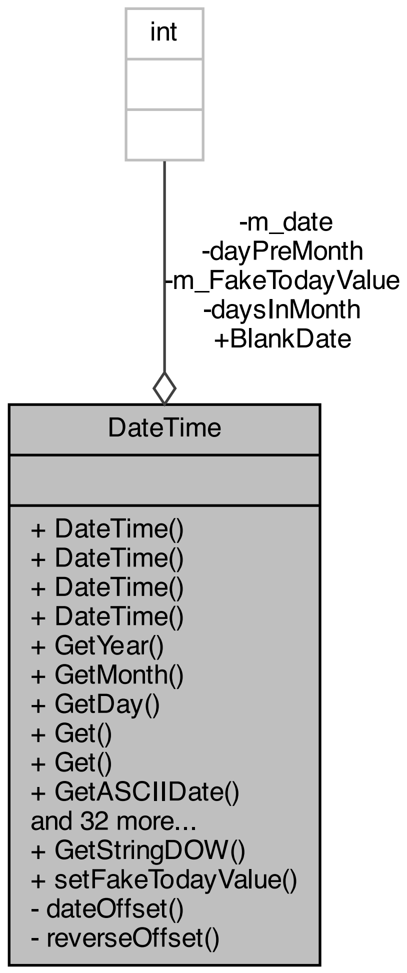
The documentation for this class was generated from the following files:

- **DataAccess.h**
- **DataAccess.cpp**

5.4 DateTime Class Reference

```
#include <DateTime.h>
```

Collaboration diagram for DateTime:



Public Types

- enum **DAY_OF_WEEK** : char {
  DAY_OF_WEEK::SUNDAY = 0, **DAY_OF_WEEK::MONDAY**, **DAY_OF_WEEK::TUESDAY**, **DAY_OF_↩ WEEK::WEDNESDAY**,
  DAY_OF_WEEK::THURSDAY, **DAY_OF_WEEK::FRIDAY**, **DAY_OF_WEEK::SATURDAY**, **DAY_OF_↩ WEEK::UNDEF_DOW** }

Public Member Functions

- **DateTime** ()

    Default constructor.
- **DateTime** (**int** a_year, **int** a_month, **int** a_day)

    General date format parameterized constructor (YYYY MM DD)
- **DateTime** (**int** a_date)

    Internal date format parameterized constructor.
- **DateTime** (const **DateTime** &a_date)

    Copy constructor.
- **int GetYear** () const
- **int GetMonth** () const
- **int GetDay** () const
- **int Get** () const

    Gets the date as one value.
- void **Get** (**int** &a_year, **int** &a_month, **int** &a_day) const

    Gets the date as component values.
- std::string **GetASCIIDate** ()

    Get date as an ASCII string in format "MM/DD/YYYY".
- **int GetDayOfWeek** () const

    Gets the Day of the week as integral value.
- void **Set** (**int** a_year, **int** a_month, **int** a_day)
- void **Set** (**int** a_date)
- void **Set** (const **DateTime** &a_date)
- bool **checkDateValueRanges** (**int** a_year, **int** a_month, **int** a_day)

    Function to assert date value ranges.
- void **SetToday** ()

    Record today's local date in this object - uses faked today's date if set.
- void **SetActualToday** ()

    Record today's local date in this object. Does not use faked today's date.
- bool **isLeapYear** () const

    Determines if the year recorded here is a leap year.
- **int getJulianDay** () const

    Gets the Julian day from this date. Julian day vary from 0 to 365.
- **int CalendarDiffDates** (const **DateTime** &a_nearDate, const **DateTime** &a_farDate)

    Computes the calendar difference between two date.
- **operator int** () const

    Conversion operators.
- **DateTime** & **operator=** (const **DateTime** &a_date)

    Assignment operators.
- **DateTime** & **operator=** (**int** a_date)
- **int operator-** (const **DateTime** &a_date)

    Finds the difference between two dates.
- **DateTime operator-** (**int** a_days)

    Subtracts a specified number of days to the date.
- **DateTime operator+** (**int** a_days)

    Adds a specified number of days to the date.
- bool **operator==** (const **DateTime** &a_date)

    Comparison operator to compare two dates:
- bool **operator==** (**int** a_date)
- bool **operator!=** (const **DateTime** &a_date)
- bool **operator!=** (**int** a_date)

- bool **operator<** (const **DateTime** &a_date)
- bool **operator<** (**int** a_date)
- bool **operator<=** (const **DateTime** &a_date)
- bool **operator<=** (**int** a_date)
- bool **operator>** (const **DateTime** &a_date)
- bool **operator>** (**int** a_date)
- bool **operator>=** (const **DateTime** &a_date)
- bool **operator>=** (**int** a_date)
- **DateTime** & **operator--** ()

  Unary minus operators. (PREFIX)
- **DateTime** **operator--** (**int**)

  Unary minus operators. (POSTFIX)
- **DateTime** & **operator++** ()

  Unary plus operators. (PREFIX)
- **DateTime** **operator++** (**int**)

  Unary plus operators. (POSTFIX)

Static Public Member Functions

- static std::string **GetStringDOW** (**DAY_OF_WEEK** a_dow)

  Get day of the week as a string.
- static void **setFakeTodayValue** (**int** a_val)

  Set the fake today value. This will be used instead of the real today.

Static Public Attributes

- static const **int** **BlankDate** = 0

Private Member Functions

- **int** **dateOffset** (const **DateTime** &a_date)

  Get the offset from the year 0.
- **DateTime** **reverseOffset** (**int** a_days)

  Reverse date offset.

Private Attributes

- **int** **m_date**

  Date stored as (year $*$ 10000 + 100 $*$ month + day)

Static Private Attributes

- static **int** **m_FakeTodayValue** = 0

  Fake date value of today.
- static **int** **dayPreMonth** [13] = { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365}

  The number of days since the beginning of the year to a given month. (non-leap year)
- static **int** **daysInMonth** [13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }

  The number of days in a month. (non-leap year)

5.4.1 Detailed Description

DateTime.h (p. **??**) Interface for the **DateTime** (p. **??**) class. General date time class. Dates represented internally as: (year ∗ 10000 + 100 ∗ month + day)

Created by Salil Maharjan on 3/22/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 17 of file DateTime.h.

5.4.2 Member Enumeration Documentation

5.4.2.1 DAY_OF_WEEK

```
enum DateTime::DAY_OF_WEEK : char [strong]
```

Enumerator

SUNDAY	
MONDAY	
TUESDAY	
WEDNESDAY	
THURSDAY	
FRIDAY	
SATURDAY	
UNDEF_DOW	

Definition at line 58 of file DateTime.h.

5.4.3 Constructor & Destructor Documentation

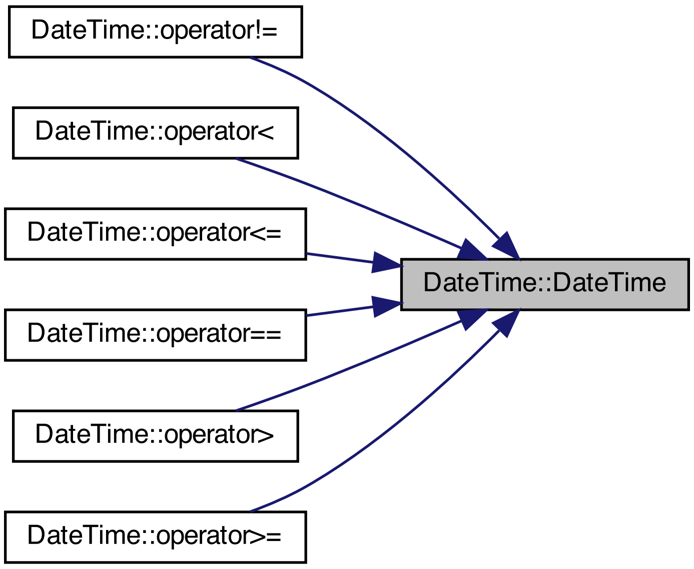
5.4.3.1 DateTime() [1/4]

```
DateTime::DateTime ( ) [inline]
```

Default constructor.

Definition at line 26 of file DateTime.h.

Here is the caller graph for this function:



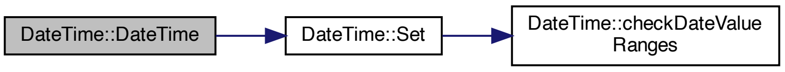
5.4.3.2 DateTime() `[2/4]`

```
DateTime::DateTime (
              int a_year,
              int a_month,
              int a_day )  [inline]
```

General date format parameterized constructor (YYYY MM DD)

Definition at line 32 of file DateTime.h.

Here is the call graph for this function:



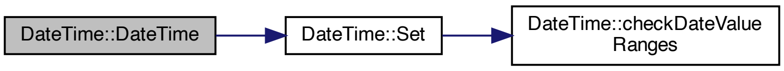
5.4.3.3 DateTime() [3/4]

```
DateTime::DateTime (
              int a_date )  [inline]
```

Internal date format parameterized constructor.

Definition at line 38 of file DateTime.h.

Here is the call graph for this function:



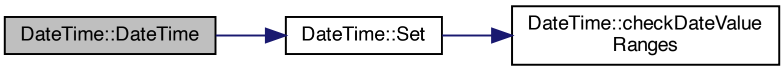
5.4.3.4 DateTime() [4/4]

```
DateTime::DateTime (
             const  DateTime & a_date )  [inline]
```

Copy constructor.

Definition at line 44 of file DateTime.h.

Here is the call graph for this function:



5.4.4 Member Function Documentation

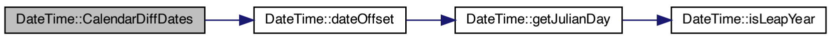
5.4.4.1 CalendarDiffDates()

```
int DateTime::CalendarDiffDates (
            const DateTime & a_nearDate,
            const DateTime & a_farDate ) [inline]
```

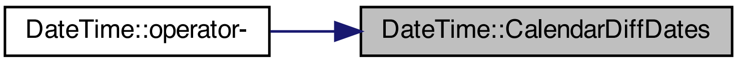
Computes the calendar difference between two date.

Definition at line 152 of file DateTime.h.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.4.2 checkDateValueRanges()

```
bool DateTime::checkDateValueRanges (
              int a_year,
              int a_month,
              int a_day )
```

Function to assert date value ranges.

DateTime::checkDateValueRanges (p. **??**) Function to assert date value ranges

Parameters

a_year	int Year
a_month	int Month
a_day	int Date

Returns

bool If passed values mark a valid date.

Author

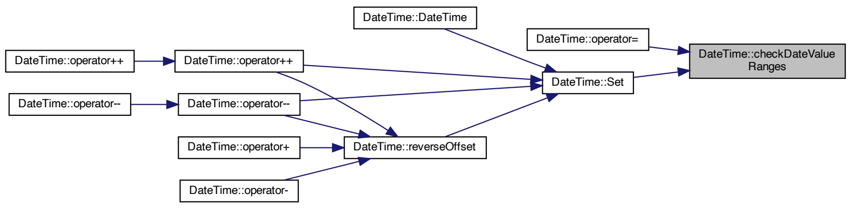
Salil Maharjan

Date

3/24/20.

Definition at line 68 of file DateTime.cpp.

Here is the caller graph for this function:



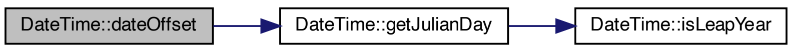
5.4.4.3 dateOffset()

```
int DateTime::dateOffset (
            const DateTime & a_date ) [inline], [private]
```

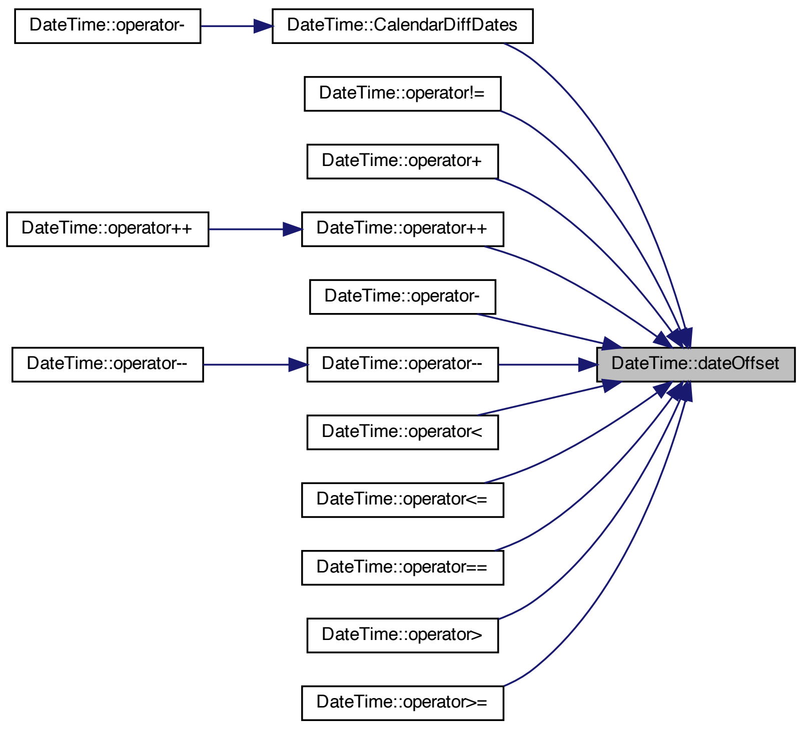
Get the offset from the year 0.

Definition at line 277 of file DateTime.h.

Here is the call graph for this function:



Here is the caller graph for this function:



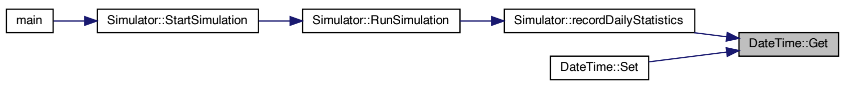
5.4.4.4 Get() `[1/2]`

```
int DateTime::Get ( ) const  [inline]
```

Gets the date as one value.

Definition at line 79 of file DateTime.h.

Here is the caller graph for this function:



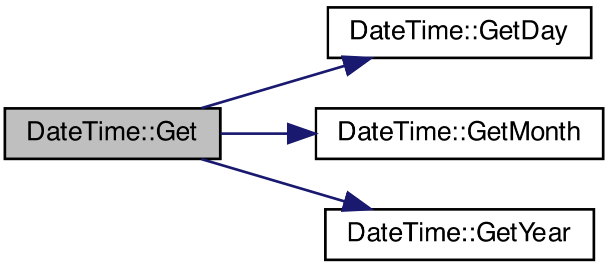
5.4.4.5 Get() [2/2]

```
void DateTime::Get (
              int & a_year,
              int & a_month,
              int & a_day ) const  [inline]
```

Gets the date as component values.

Definition at line 82 of file DateTime.h.

Here is the call graph for this function:



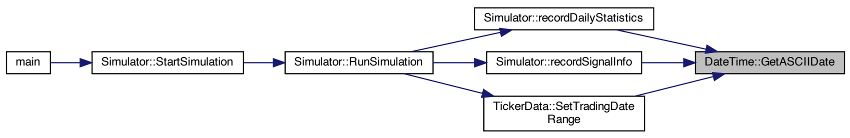
5.4.4.6 GetASCIIDate()

```
std::string DateTime::GetASCIIDate ( )  [inline]
```

Get date as an ASCII string in format "MM/DD/YYYY".

Definition at line 90 of file DateTime.h.

Here is the caller graph for this function:

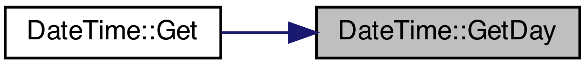

Simulator::recordDailyStatistics
main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordSignalInfo
DateTime::GetASCIIDate
TickerData::SetTradingDate Range

5.4.4.7 GetDay()

```
int DateTime::GetDay ( ) const  [inline]
```

Definition at line 76 of file DateTime.h.

Here is the caller graph for this function:



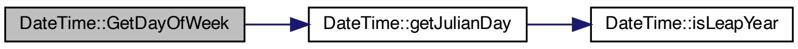
5.4.4.8 GetDayOfWeek()

```
int DateTime::GetDayOfWeek ( ) const  [inline]
```

Gets the Day of the week as integral value.

Definition at line 106 of file DateTime.h.

Here is the call graph for this function:



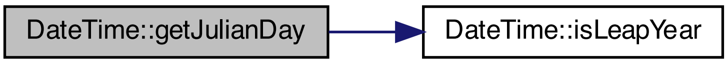
5.4.4.9 getJulianDay()

```
int DateTime::getJulianDay ( ) const  [inline]
```

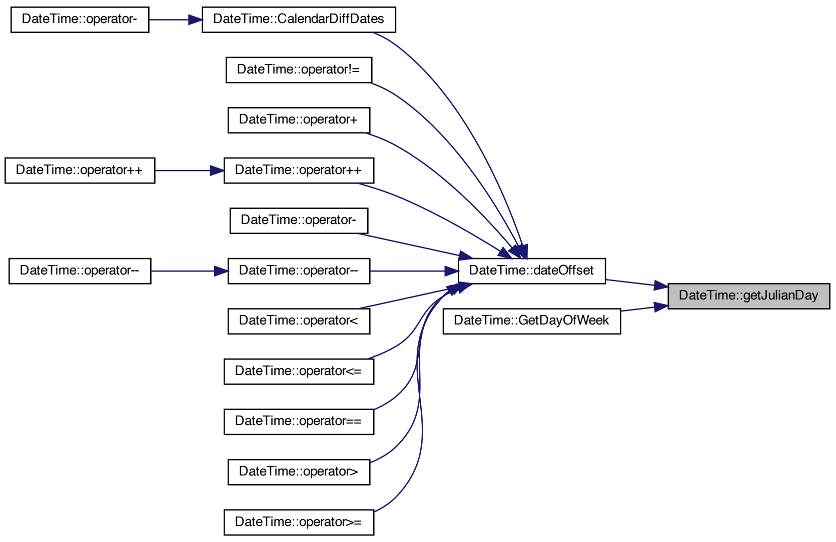
Gets the Julian day from this date. Julian day vary from 0 to 365.

Definition at line 142 of file DateTime.h.

Here is the call graph for this function:



Here is the caller graph for this function:

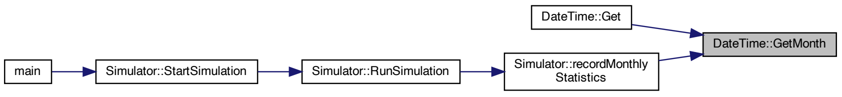


5.4.4.10 GetMonth()

```
int DateTime::GetMonth ( ) const  [inline]
```

Definition at line 75 of file DateTime.h.

Here is the caller graph for this function:


DateTime::Get
DateTime::GetMonth
main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordMonthly Statistics

5.4.4.11 GetStringDOW()

```
static std::string DateTime::GetStringDOW (
              DAY_OF_WEEK a_dow )  [inline], [static]
```

Get day of the week as a string.

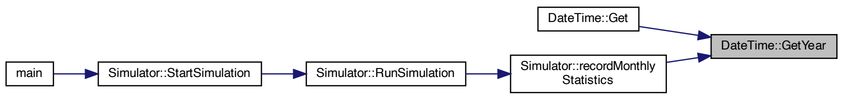
Definition at line 93 of file DateTime.h.

5.4.4.12 GetYear()

```
int DateTime::GetYear ( ) const  [inline]
```

Definition at line 74 of file DateTime.h.

Here is the caller graph for this function:


DateTime::Get
DateTime::GetYear
main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordMonthly Statistics

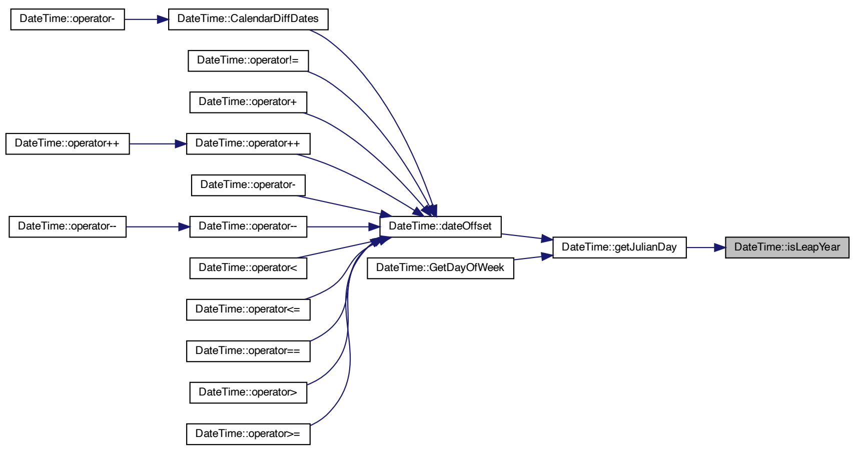
5.4.4.13 isLeapYear()

```
bool DateTime::isLeapYear ( ) const  [inline]
```

Determines if the year recorded here is a leap year.

Definition at line 139 of file DateTime.h.

Here is the caller graph for this function:



5.4.4.14 operator int()

```
DateTime::operator  int ( ) const  [inline]
```

Conversion operators.

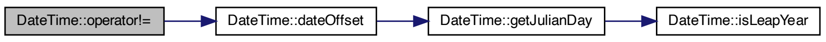
Definition at line 163 of file DateTime.h.

5.4.4.15 operator"!=() [1/2]

```
bool DateTime::operator!= (
             const  DateTime & a_date )  [inline]
```

Definition at line 190 of file DateTime.h.

Here is the call graph for this function:

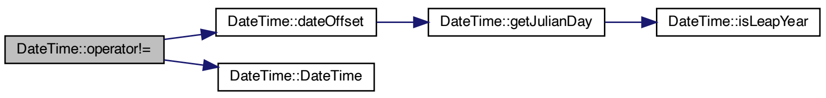


5.4.4.16 operator"!=() [2/2]

```
bool DateTime::operator!= (
              int a_date )  [inline]
```

Definition at line 191 of file DateTime.h.

Here is the call graph for this function:



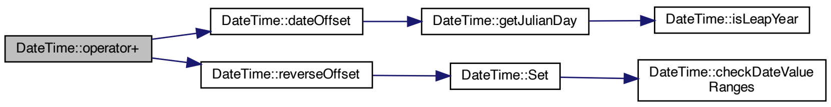
5.4.4.17 operator+()

```
DateTime DateTime::operator+ (
              int a_days )  [inline]
```

Adds a specified number of days to the date.

Definition at line 180 of file DateTime.h.

Here is the call graph for this function:



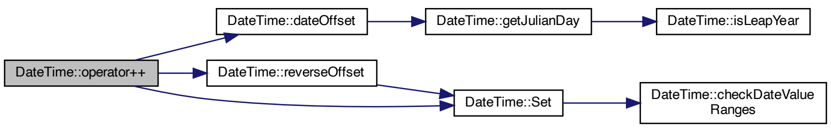
5.4.4.18 operator++() [1/2]

```
DateTime& DateTime::operator++ ( )  [inline]
```

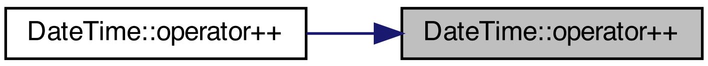
Unary plus operators. (PREFIX)

Definition at line 242 of file DateTime.h.

Here is the call graph for this function:



Here is the caller graph for this function:



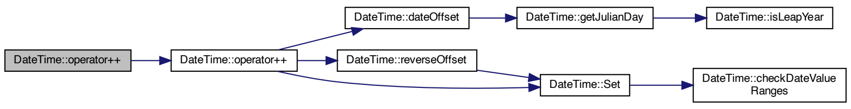
5.4.4.19 operator++() [2/2]

```
DateTime DateTime::operator++ (
             int   )  [inline]
```

Unary plus operators. (POSTFIX)

Definition at line 250 of file DateTime.h.

Here is the call graph for this function:



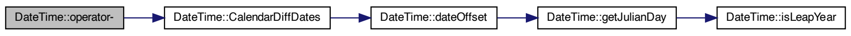
5.4.4.20 operator-() [1/2]

```
int DateTime::operator- (
            const  DateTime & a_date )  [inline]
```

Finds the difference between two dates.

Definition at line 174 of file DateTime.h.

Here is the call graph for this function:



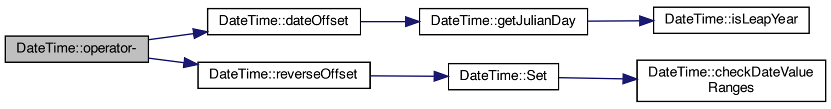
5.4.4.21 operator-() [2/2]

```
DateTime DateTime::operator- (
             int a_days )  [inline]
```

Subtracts a specified number of days to the date.

Definition at line 177 of file DateTime.h.

Here is the call graph for this function:



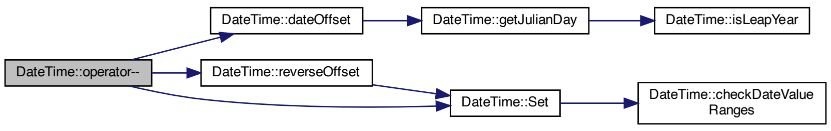
5.4.4.22 operator--() [1/2]

```
DateTime& DateTime::operator-- ( )  [inline]
```

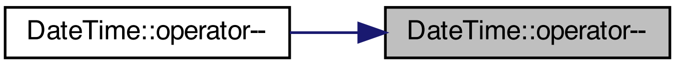
Unary minus operators. (PREFIX)

Definition at line 226 of file DateTime.h.

Here is the call graph for this function:



Here is the caller graph for this function:



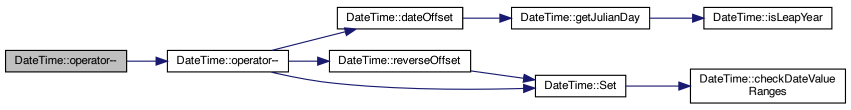
5.4.4.23 operator--() [2/2]

```
DateTime DateTime::operator-- (
            int  )  [inline]
```

Unary minus operators. (POSTFIX)

Definition at line 234 of file DateTime.h.

Here is the call graph for this function:

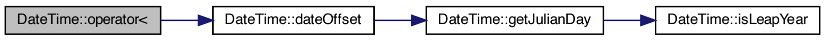


5.4.4.24 operator<() [1/2]

```
bool DateTime::operator< (
             const  DateTime & a_date )  [inline]
```

Definition at line 197 of file DateTime.h.

Here is the call graph for this function:

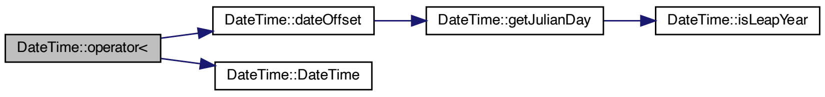


5.4.4.25 operator<() [2/2]

```
bool DateTime::operator< (
              int a_date )  [inline]
```

Definition at line 198 of file DateTime.h.

Here is the call graph for this function:

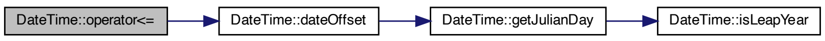


5.4.4.26 operator<=() [1/2]

```
bool DateTime::operator<= (
             const  DateTime & a_date )  [inline]
```

Definition at line 204 of file DateTime.h.

Here is the call graph for this function:

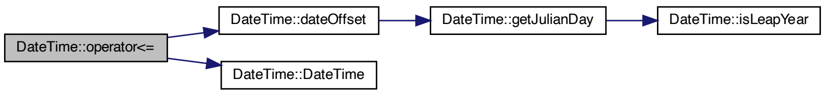


5.4.4.27 operator<=() [2/2]

```
bool DateTime::operator<= (
              int a_date )  [inline]
```

Definition at line 205 of file DateTime.h.

Here is the call graph for this function:



5.4.4.28 operator=() [1/2]

```
 DateTime& DateTime::operator= (
             const  DateTime & a_date )  [inline]
```

Assignment operators.

Definition at line 166 of file DateTime.h.

5.4.4.29 operator=() [2/2]

```
 DateTime & DateTime::operator= (
              int a_date )
```

DateTime::operator = (p. **??**) Assignment operator overload.

Parameters

a_date	**DateTime** (p. **??**) **DateTime** (p. **??**) object to assign.

Returns

DateTime (p. **??**)& New **DateTime** (p. **??**) that is assigned.

Author

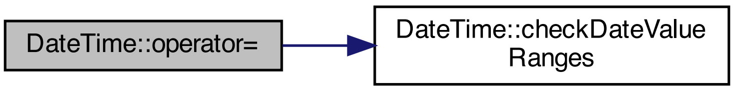
Salil Maharjan

Date

5/12/20.

Definition at line 132 of file DateTime.cpp.

Here is the call graph for this function:



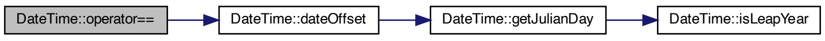
5.4.4.30 operator==() `[1/2]`

```
bool DateTime::operator== (
             const  DateTime & a_date )  [inline]
```

Comparison operator to compare two dates:

Definition at line 183 of file DateTime.h.

Here is the call graph for this function:

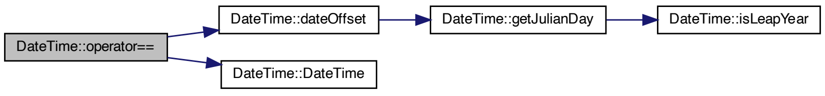


5.4.4.31 operator==() `[2/2]`

```
bool DateTime::operator== (
              int a_date )  [inline]
```

Definition at line 184 of file DateTime.h.

Here is the call graph for this function:

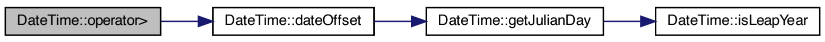


5.4.4.32 operator>() [1/2]

```
bool DateTime::operator> (
            const DateTime & a_date ) [inline]
```

Definition at line 211 of file DateTime.h.

Here is the call graph for this function:

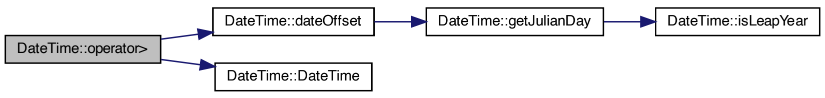


5.4.4.33 operator>() [2/2]

```
bool DateTime::operator> (
            int a_date ) [inline]
```

Definition at line 212 of file DateTime.h.

Here is the call graph for this function:

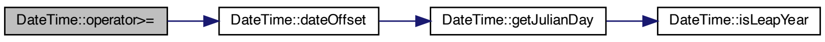


5.4.4.34 operator>=() [1/2]

```
bool DateTime::operator>= (
            const DateTime & a_date ) [inline]
```

Definition at line 218 of file DateTime.h.

Here is the call graph for this function:

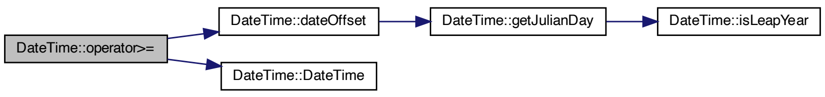


5.4.4.35 operator>=() [2/2]

```
bool DateTime::operator>= (
              int a_date )  [inline]
```

Definition at line 219 of file DateTime.h.

Here is the call graph for this function:



5.4.4.36 reverseOffset()

```
 DateTime DateTime::reverseOffset (
              int a_days )  [private]
```

Reverse date offset.

DateTime::reverseOffset (p. **??**) Reverse date offset

Parameters

a_days	int Number of days to reverse offset.

Returns

DateTime (p. **??**) The date after reversing a_days offset.

Author

Salil Maharjan

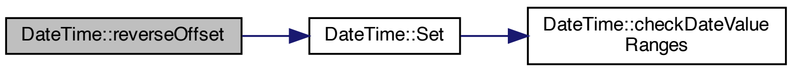
Date

5/12/20.

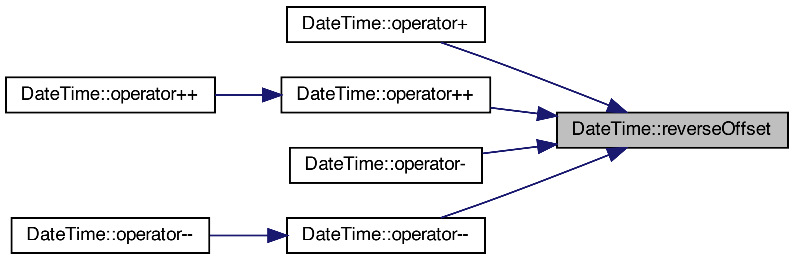
Change the return value!

Definition at line 152 of file DateTime.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

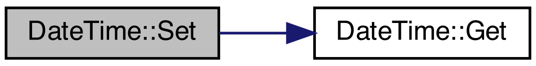


5.4.4.37 Set() `[1/3]`

```
void DateTime::Set (
             const  DateTime & a_date )  [inline]
```

Definition at line 119 of file DateTime.h.

Here is the call graph for this function:



5.4.4.38 Set() `[2/3]`

```
void DateTime::Set (
              int a_date )
```

DateTime::Set (p. **??**) Date set function for Internal date format parameterized constructor

Parameters

a_date	int Date passed in internal format YYYYMMDD

Author

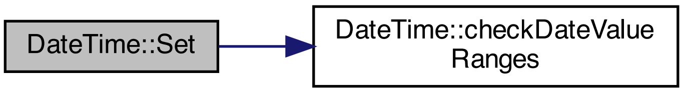
Salil Maharjan

Date

3/24/20.

Definition at line 48 of file DateTime.cpp.

Here is the call graph for this function:



5.4.4.39 Set() [3/3]

```
void DateTime::Set (
              int a_year,
              int a_month,
              int a_day )
```

DateTime::Set (p. **??**) Date set function for General date format parameterized constructor (YYYY MM DD)

Parameters

a_year	int Year
a_month	int Month
a_day	int Date

Author

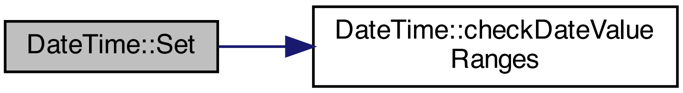
Salil Maharjan

Date

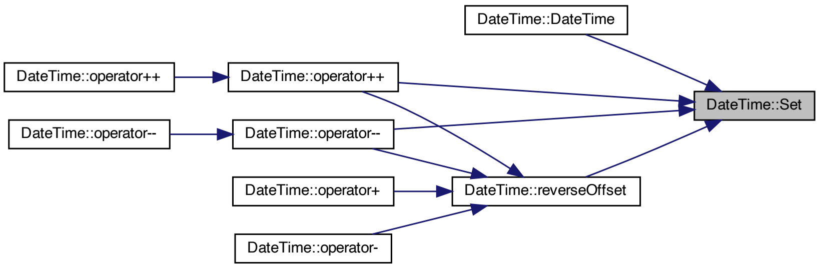
3/22/20.

Definition at line 31 of file DateTime.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.4.40 SetActualToday()

```
void DateTime::SetActualToday ( )
```

Record today's local date in this object. Does not use faked today's date.

DateTime::SetToday (p. **??**) Record today's local date in this object. Does not use faked today's date.

Author

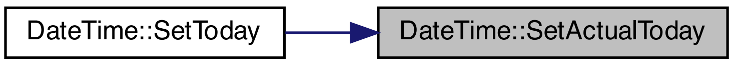
Salil Maharjan

Date

5/12/20.

Definition at line 117 of file DateTime.cpp.

Here is the caller graph for this function:



5.4.4.41 setFakeTodayValue()

```
static void DateTime::setFakeTodayValue (
              int a_val )  [inline], [static]
```

Set the fake today value. This will be used instead of the real today.

Definition at line 125 of file DateTime.h.

5.4.4.42 SetToday()

```
void DateTime::SetToday ( )
```

Record today's local date in this object - uses faked today's date if set.

DateTime::SetToday (p. **??**) Record today's local date in this object - uses faked today's date if set.

Author

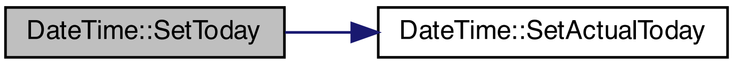
Salil Maharjan

Date

5/12/20.

Definition at line 99 of file DateTime.cpp.

Here is the call graph for this function:



5.4.5 Member Data Documentation

5.4.5.1 BlankDate

```
const int DateTime::BlankDate = 0 [static]
```

Definition at line 54 of file DateTime.h.

5.4.5.2 dayPreMonth

```
int DateTime::dayPreMonth = { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365}
[static], [private]
```

The number of days since the beginning of the year to a given month. (non-leap year)

Definition at line 268 of file DateTime.h.

5.4.5.3 daysInMonth

```
int DateTime::daysInMonth = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 } [static],
[private]
```

The number of days in a month. (non-leap year)

Definition at line 270 of file DateTime.h.

5.4.5.4 m_date

```
int DateTime::m_date [private]
```

Date stored as (year $*$ 10000 + 100 $*$ month + day)

Definition at line 264 of file DateTime.h.

5.4.5.5 m_FakeTodayValue

```
int DateTime::m_FakeTodayValue = 0  [static], [private]
```

Fake date value of today.

DateTime.cpp (p. **??**) Implementation of DateTime.hpp

Created by Salil Maharjan on 3/22/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 266 of file DateTime.h.

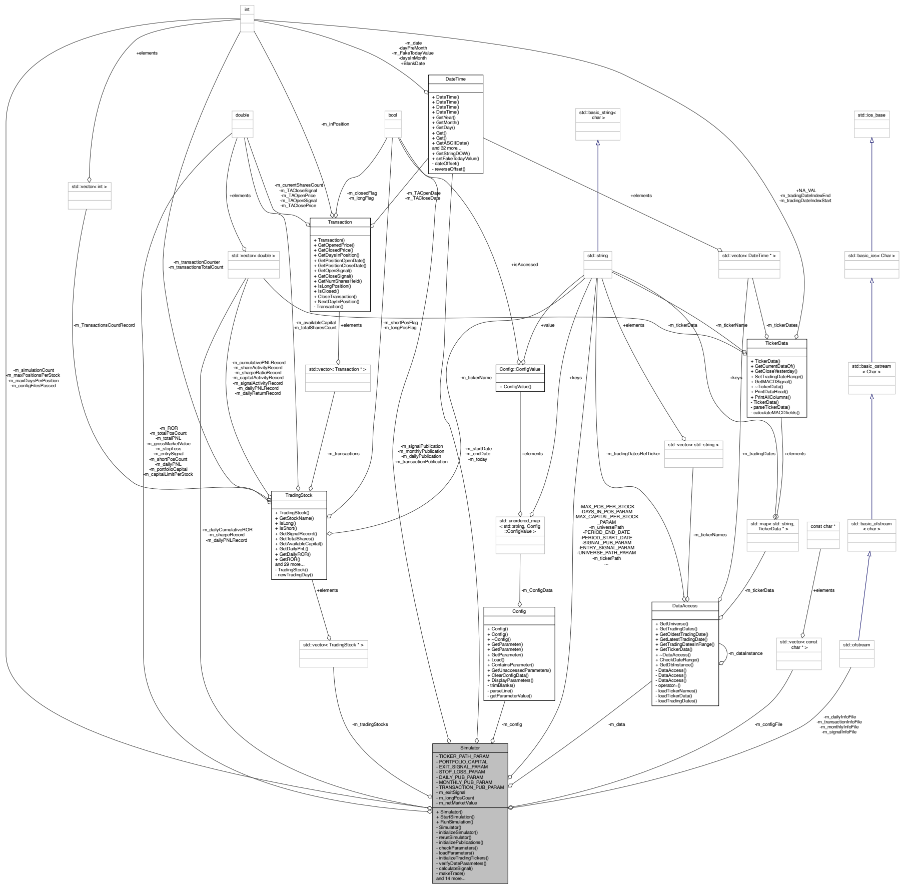
The documentation for this class was generated from the following files:

- **DateTime.h**
- **DateTime.cpp**

5.5 Simulator Class Reference

```
#include <Simulator.h>
```

Collaboration diagram for Simulator:



Public Member Functions

- **Simulator** (**int** argc, const **char** ∗argv[ ])
  Main simulator constructor.
- void **StartSimulation** ()
  Interface method to start the simulation.
- void **RunSimulation** ()
  Main method to run simulation.

Private Member Functions

- **Simulator** ()=default
  Private default constructor.
- void **initializeSimulator** (const **char** ∗a_configFile)
  Initializes the simulator by loading configurations, ticker data and trading stocks.
- void **rerunSimulator** (const **char** ∗a_configFile)
  Rerun simulator for consecutive configuration files.
- void **initializePublications** ()
  Initialize file write stream for the publications requested in config.
- bool **checkParameters** ()
  Checks if all required parameters are passed for simulator to run.
- void **loadParameters** ()
  Loads parameters required for the simulator.
- void **initializeTradingTickers** ()
  Initialize stocks that the simulator will be trading.
- void **verifyDateParameters** ()
  Verify date range.
- double **calculateSignal** (**TickerData** &a_stockData)
  Calculate trade signal.
- void **makeTrade** (**TradingStock** &a_stock, double &a_currentPrice, double &a_signalToday)
  Function to open or close positions according to the signal by checking current position.
- bool **hasEnoughCapital** (**TradingStock** &a_stock, double a_price)
  Check if available capital for a stock is enough to purchase more equities at a_price.
- void **closePositions** (**TradingStock** &a_stock, double &a_currentPrice, double &a_signal)
  Close positions for a trading stock.
- void **openPositions** (**TradingStock** &a_stock, double &a_currentPrice, double &a_signal)
  Open positions for a trading stock.
- void **recordTransaction** (**TradingStock** &a_stock)
  Function to record transactions to a file.
- void **recordSignalInfo** (double a_signal)
  Function to record signals to a file.
- void **recordDailyStatistics** ()
  Function to record daily statistics to a file.
- void **recordMonthlyStatistics** ()
  Function to record monthly statistics to a file.
- void **closeDailyPublication** ()
  Method to close daily publication record file.
- void **closeMonthlyPublication** (bool a_flag)
  Method to close monthly publication record file.
- void **closeTransactionPublication** ()

Method to close transaction publication record file.
- void **closeSignalPublication** ()

  Method to close signal publication record file.
- void **recordSharpeRatio** ()

  Function to record sharpe ratio to daily statistics file.
- void **recordNoActivity** (**TradingStock** &a_stock)

  Method to record necessary elements for no activity (code reuse purposes).
- double **calculateSharpeRatio** ()

  Function to calculate sharpe ratio.

Private Attributes

- const std::string **UNIVERSE_PATH_PARAM** = "UNIVERSE_DIRECTORY"
- const std::string **TICKER_PATH_PARAM** = "TICKER_DATA_DIRECTORY"
- const std::string **PORTFOLIO_CAPITAL** = "PORTFOLIO_CAPITAL"
- const std::string **MAX_CAPITAL_PER_STOCK_PARAM** = "MAX_CAPITAL_PER_STOCK"
- const std::string **PERIOD_START_DATE** = "START_DATE"
- const std::string **PERIOD_END_DATE** = "END_DATE"
- const std::string **ENTRY_SIGNAL_PARAM** = "ENTRY_THRESHOLD"
- const std::string **EXIT_SIGNAL_PARAM** = "EXIT_THRESHOLD"
- const std::string **MAX_POS_PER_STOCK** = "MAX_POSITIONS_PER_STOCK"
- const std::string **DAYS_IN_POS_PARAM** = "MAX_DAYS_IN_POSITION"
- const std::string **STOP_LOSS_PARAM** = "STOP_LOSS"
- const std::string **DAILY_PUB_PARAM** = "DAILY"
- const std::string **MONTHLY_PUB_PARAM** = "MONTHLY"
- const std::string **TRANSACTION_PUB_PARAM** = "TRANSACTION"
- const std::string **SIGNAL_PUB_PARAM** = "SIGNAL"
- std::string **m_universePath**
- std::string **m_tickerPath**
- double **m_portfolioCapital**
- double **m_capitalLimitPerStock**
- double **m_entrySignal**
- double **m_exitSignal**
- **DateTime** $*$ **m_startDate**
- **DateTime** $*$ **m_endDate**
- **int m_maxPositionsPerStock**
- **int m_maxDaysPerPosition**
- double **m_stopLoss**
- bool **m_dailyPublication**
- bool **m_monthlyPublication**
- bool **m_transactionPublication**
- bool **m_signalPublication**
- std::ofstream **m_dailyInfoFile**

  Output streams to write publications.
- std::ofstream **m_monthlyInfoFile**
- std::ofstream **m_transactionInfoFile**
- std::ofstream **m_signalInfoFile**
- double **m_ROR**

  Daily return. The profit for the day divided by the capital committed.
- double **m_totalPNL**

  Cumulative PNL from the beginning of the simulation to the current date.
- double **m_dailyPNL**

The profit or loss in dollars for the day.

- double **m_longPosCount**

  Daily long position counter variable.
- double **m_shortPosCount**

  Daily short position counter variable.
- double **m_totalPosCount**

  Daily total position counter variable.
- double **m_netMarketValue**

  Total amount of capital committed to the model at the end of the day.
- double **m_grossMarketValue**

  Long capital minus the short capital at the end of the day.
- **DateTime** $*$ **m_today**

  Date today.
- std::vector$<$ double $>$ **m_dailyPNLRecord**

  Daily PNL record.
- std::vector$<$ double $>$ **m_dailyCumulativeROR**

  Daily cumulative ROR record for all trading stocks in simulation.
- std::vector$<$ double $>$ **m_sharpeRecord**

  Sharpe ratio record.
- **int m_configFilesPassed**

  Number of config files passed.
- **int m_simulationCount**

  Count of how many config files have been simulated.
- std::vector$<$ const **char** $*>$ **m_configFile**

  Var with config file name.
- **Config** $*$ **m_config**

  Configuration object.
- **DataAccess** $*$ **m_data**

  Data access object.
- std::vector$<$ **TradingStock** $*>$ **m_tradingStocks**

  All the trading stocks in the simulation.

5.5.1 Detailed Description

Simulator.h (p. **??**) Main container class for the simulator. Uses member classes for financial simulation.

Created by Salil Maharjan on 4/29/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 18 of file Simulator.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Simulator() [1/2]

```
Simulator::Simulator (
              int argc,
             const  char * argv[] )
```

Main simulator constructor.

Simulator.cpp (p. **??**) Implementation of **Simulator.h** (p. **??**).

Created by Salil Maharjan on 4/29/20. Copyright © 2020 Salil Maharjan. All rights reserved. **Simulator::Simulator** (p. **??**) Parameterized constructor for **Simulator** (p. **??**) class. Initializes the **Simulator** (p. **??**). Can handle multiple configuration files.

Parameters

argc	int Number of command line arguments.
argv	const char∗ Array of command line arguments.

Author

Salil Maharjan

Date

5/12/20.

Definition at line 26 of file Simulator.cpp.

5.5.2.2 Simulator() [2/2]

```
Simulator::Simulator ( )  [private], [default]
```

Private default constructor.

5.5.3 Member Function Documentation

5.5.3.1 calculateSharpeRatio()

```
double Simulator::calculateSharpeRatio ( )  [private]
```

Function to calculate sharpe ratio.

Simulator::calculateSharpeRatio (p. **??**) Method to calculate sharpe ratio.

Returns

double Daily sharpe ratio of the model.

Author

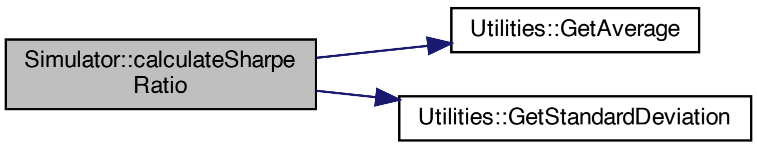
Salil Maharjan

Date

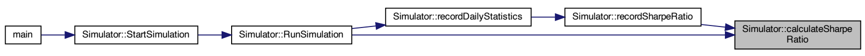
5/12/20.

Definition at line 667 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::calculateSharpe Ratio
Utilities::GetAverage
Utilities::GetStandardDeviation

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordDailyStatistics
Simulator::recordSharpeRatio
Simulator::calculateSharpe Ratio

5.5.3.2 calculateSignal()

```
double Simulator::calculateSignal (
               TickerData & a_stockData )  [private]
```

Calculate trade signal.

Simulator::calculateSignal (p. **??**) Function to calculate trade signal. Currently uses MACD indicator to get signal. If not, the simulation will run for all available data instead of stopping.

Parameters

a_stockData	**TickerData** (p. **??**)& The data of the trading stock in the specified date range.

Returns

double Trading signal.

Author

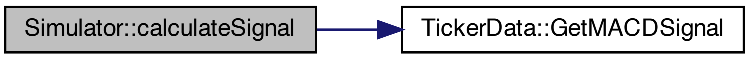
Salil Maharjan

Date

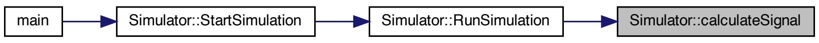
5/12/20.

Definition at line 470 of file Simulator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.3 checkParameters()

```
bool Simulator::checkParameters ( )  [private]
```

Checks if all required parameters are passed for simulator to run.

Simulator::checkParameters (p. **??**) Checks if all required parameters are passed for simulator to run.

Returns

bool If all required parameters are in configuration file.

Author

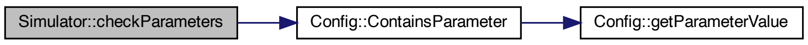
Salil Maharjan

Date

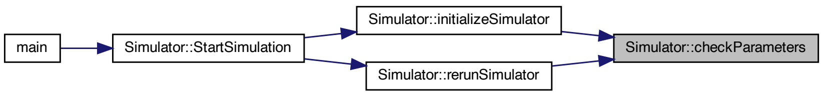
5/12/20.

Definition at line 337 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::checkParameters
Config::ContainsParameter
Config::getParameterValue

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::checkParameters

5.5.3.4 closeDailyPublication()

`void Simulator::closeDailyPublication ( )  [private]`

Method to close daily publication record file.

Simulator::closeDailyPublication (p. **??**) Method to close daily publication record file.

Author

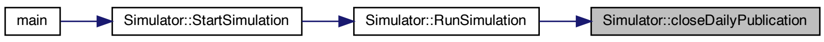
Salil Maharjan

Date

5/12/20.

Definition at line 922 of file Simulator.cpp.

Here is the caller graph for this function:



5.5.3.5 closeMonthlyPublication()

```
void Simulator::closeMonthlyPublication (
             bool a_flag )  [private]
```

Method to close monthly publication record file.

Simulator::closeMonthlyPublication (p. **??**) Method to close monthly publication record file. Monthly statistics use daily statistics, so daily must be set to write monthly publications.

Parameters

a_flag	Flag if monthly statistics can be generated.

Author

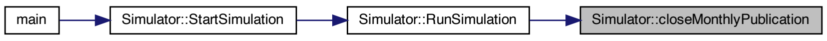
Salil Maharjan

Date

5/12/20.

Definition at line 940 of file Simulator.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::closeMonthlyPublication

5.5.3.6 closePositions()

```
void Simulator::closePositions (
              TradingStock & a_stock,
             double & a_currentPrice,
             double & a_signal )  [private]
```

Close positions for a trading stock.

Simulator::closePositions (p. **??**) Method to close positions for a trading stock.

Parameters

a_stock	**TradingStock** (p. **??**)& The stock that is being traded.
a_currentPrice	double& Current price of the stock.
a_signalToday	double& Signal calculated for the day.

Author

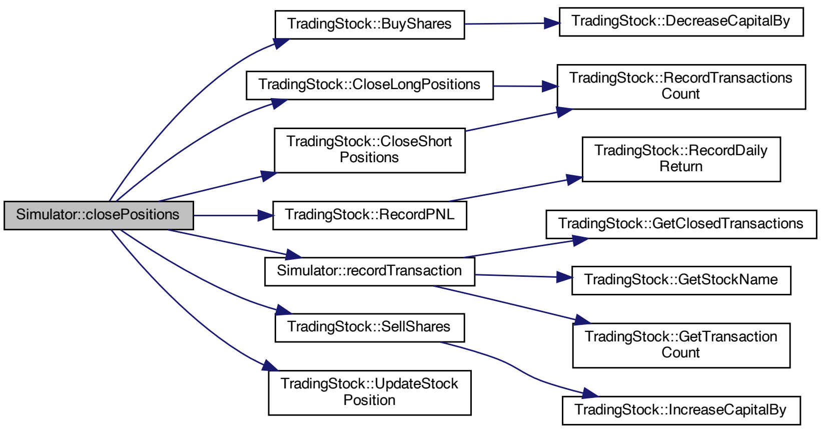
Salil Maharjan

Date

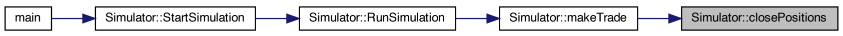
5/12/20.

Definition at line 631 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::closePositions
TradingStock::BuyShares
TradingStock::CloseLongPositions
TradingStock::CloseShort Positions
TradingStock::RecordPNL
Simulator::recordTransaction
TradingStock::SellShares
TradingStock::UpdateStock Position
TradingStock::DecreaseCapitalBy
TradingStock::RecordTransactions Count
TradingStock::RecordDaily Return
TradingStock::GetClosedTransactions
TradingStock::GetStockName
TradingStock::GetTransaction Count
TradingStock::IncreaseCapitalBy

Here is the caller graph for this function:



5.5.3.7 closeSignalPublication()

```
void Simulator::closeSignalPublication ( )  [private]
```

Method to close signal publication record file.

Simulator::closeSignalPublication (p. **??**) Method to close signal publication record file.

Author

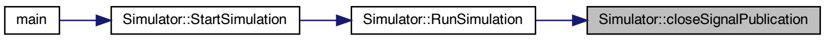
Salil Maharjan

Date

5/12/20.

Definition at line 976 of file Simulator.cpp.

Here is the caller graph for this function:



5.5.3.8 closeTransactionPublication()

```
void Simulator::closeTransactionPublication ( )  [private]
```

Method to close transaction publication record file.

Simulator::closeTransactionPublication (p. **??**) Method to close transaction publication record file.

Author

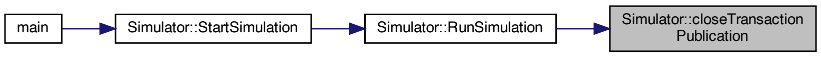
Salil Maharjan

Date

5/12/20.

Definition at line 959 of file Simulator.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::closeTransaction Publication

5.5.3.9 hasEnoughCapital()

```
bool Simulator::hasEnoughCapital (
              TradingStock & a_stock,
             double a_price )  [private]
```

Check if available capital for a stock is enough to purchase more equities at a_price.

Simulator::hasEnoughCapital (p. **??**) Check if available capital for a stock is enough to purchase more equities at a_price

Parameters

a_stock	**TradingStock** (p. **??**)& The stock that is being traded.
a_currentPrice	double& Current price of the stock.

Returns

bool If there is sufficient capital to buy positions of a_stock at a_price.

Author

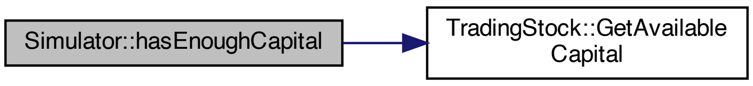
Salil Maharjan

Date

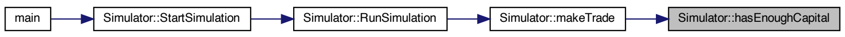
5/12/20.

Definition at line 559 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::hasEnoughCapital
TradingStock::GetAvailable Capital

Here is the caller graph for this function:



5.5.3.10 initializePublications()

```
void Simulator::initializePublications ( )  [private]
```

Initialize file write stream for the publications requested in config.

Simulator::initializePublications (p. **??**) Initializes file write streams for the publications requested in configurations.

Author

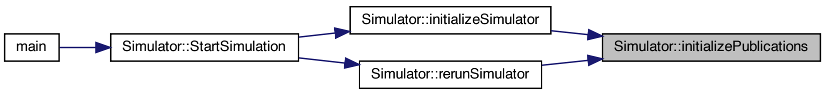
Salil Maharjan

Date

5/12/20.

Definition at line 278 of file Simulator.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::initializePublications

5.5.3.11 initializeSimulator()

```
void Simulator::initializeSimulator (
            const  char * a_configFile )  [private]
```

Initializes the simulator by loading configurations, ticker data and trading stocks.

Simulator::initializeSimulator (p. **??**) Main method to initialize simulator for run.

Parameters

a_configFile	char∗ **Config** (p. **??**) file to initialize simulator for.

Author

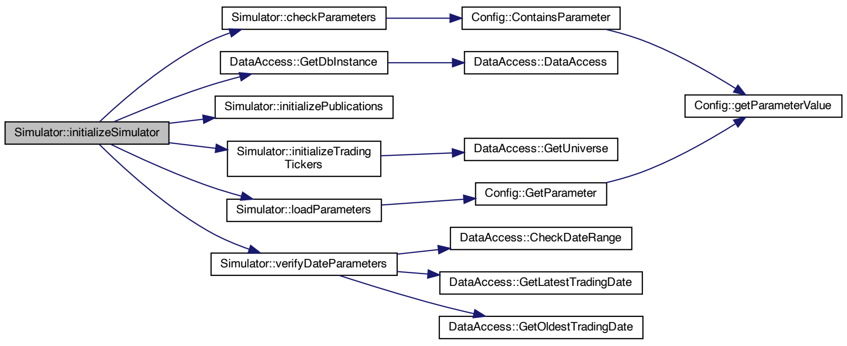
Salil Maharjan

Date

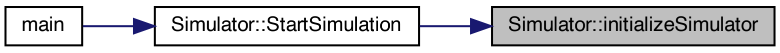
5/12/20.

Definition at line 212 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::initializeSimulator
Simulator::checkParameters
DataAccess::GetDbInstance
Simulator::initializePublications
Simulator::initializeTrading Tickers
Simulator::loadParameters
Simulator::verifyDateParameters
Config::ContainsParameter
DataAccess::DataAccess
DataAccess::GetUniverse
Config::GetParameter
DataAccess::CheckDateRange
DataAccess::GetLatestTradingDate
DataAccess::GetOldestTradingDate
Config::getParameterValue

Here is the caller graph for this function:



5.5.3.12 initializeTradingTickers()

```
void Simulator::initializeTradingTickers ( )  [private]
```

Initialize stocks that the simulator will be trading.

Simulator::initializeTradingTickers (p. **??**) Initialize stock tickers from constituents file that the simulator will be trading.

Author

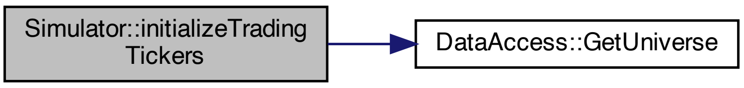
Salil Maharjan

Date

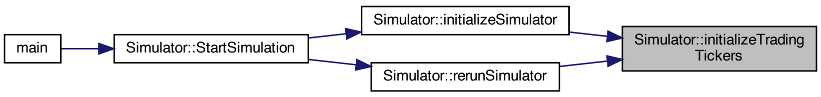
5/12/20.

Definition at line 410 of file Simulator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::initializeTrading Tickers

5.5.3.13 loadParameters()

```
void Simulator::loadParameters ( )  [private]
```

Loads parameters required for the simulator.

Simulator::loadParameters (p. **??**) Loads parameters from configuration file that are required for the simulation.

Author

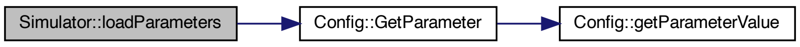
Salil Maharjan

Date

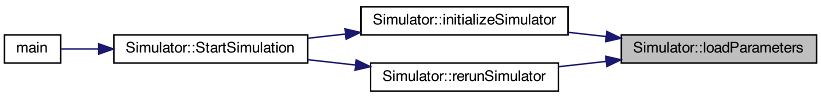
5/12/20.

Definition at line 372 of file Simulator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::loadParameters

5.5.3.14 makeTrade()

```
void Simulator::makeTrade (
              TradingStock & a_stock,
             double & a_currentPrice,
             double & a_signalToday )  [private]
```

Function to open or close positions according to the signal by checking current position.

Simulator::makeTrade (p. **??**) Function to make trade according to the signal.

Parameters

a_stock	**TradingStock** (p. **??**)& The stock that is being traded.
a_currentPrice	double& Current price of the stock.
a_signalToday	double& Signal calculated for the day.

Author

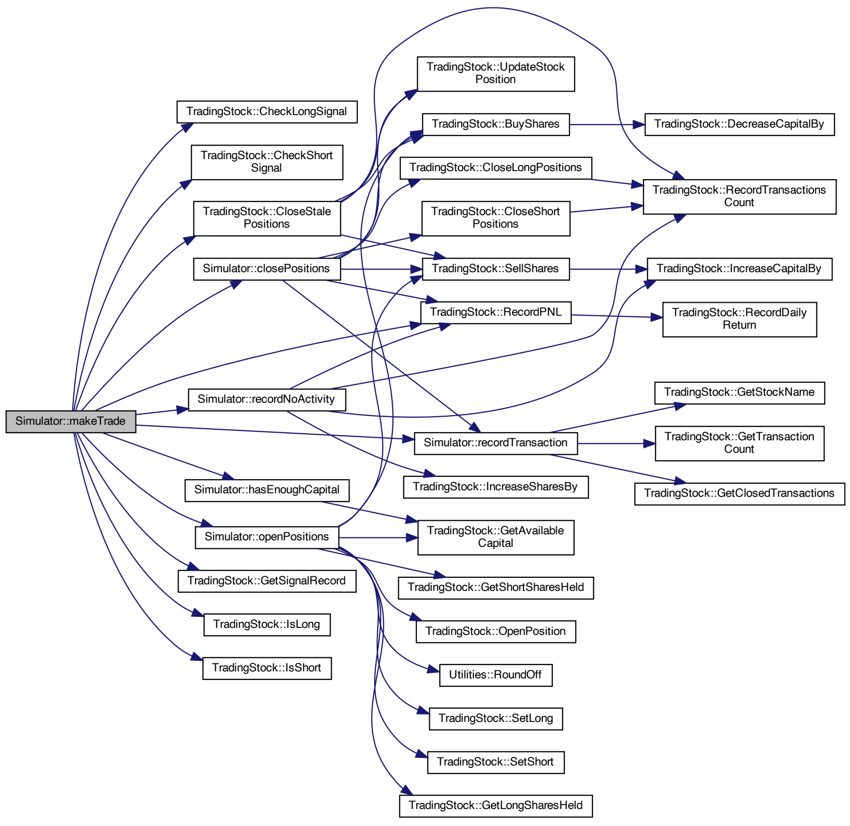
Salil Maharjan

Date

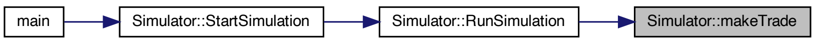
5/12/20.

Definition at line 485 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::makeTrade
TradingStock::CheckLongSignal
TradingStock::CheckShort Signal
TradingStock::CloseStale Positions
Simulator::closePositions
Simulator::recordNoActivity
Simulator::hasEnoughCapital
Simulator::openPositions
TradingStock::GetSignalRecord
TradingStock::IsLong
TradingStock::IsShort
TradingStock::UpdateStock Position
TradingStock::BuyShares
TradingStock::CloseLongPositions
TradingStock::CloseShort Positions
TradingStock::SellShares
TradingStock::RecordPNL
Simulator::recordTransaction
TradingStock::IncreaseSharesBy
TradingStock::GetAvailable Capital
TradingStock::GetShortSharesHeld
TradingStock::OpenPosition
Utilities::RoundOff
TradingStock::SetLong
TradingStock::SetShort
TradingStock::GetLongSharesHeld
TradingStock::DecreaseCapitalBy
TradingStock::RecordTransactions Count
TradingStock::IncreaseCapitalBy
TradingStock::RecordDaily Return
TradingStock::GetStockName
TradingStock::GetTransaction Count
TradingStock::GetClosedTransactions

Here is the caller graph for this function:



5.5.3.15 openPositions()

```
void Simulator::openPositions (
              TradingStock & a_stock,
             double & a_currentPrice,
             double & a_signal )  [private]
```

Open positions for a trading stock.

Simulator::openPositions (p. **??**) Method to open positions for a trading stock.

Parameters

a_stock	**TradingStock** (p. **??**)& The stock that is being traded.
a_currentPrice	double& Current price of the stock.
a_signalToday	double& Signal calculated for the day.

Author

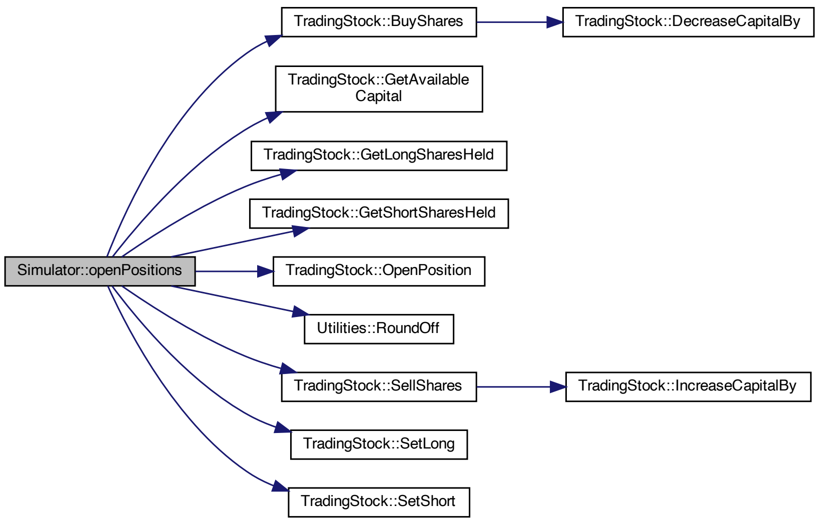
Salil Maharjan

Date

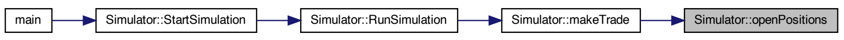
5/12/20.

Definition at line 588 of file Simulator.cpp.

Here is the call graph for this function:


TradingStock::BuyShares
TradingStock::DecreaseCapitalBy
TradingStock::GetAvailable Capital
TradingStock::GetLongSharesHeld
TradingStock::GetShortSharesHeld
Simulator::openPositions
TradingStock::OpenPosition
Utilities::RoundOff
TradingStock::SellShares
TradingStock::IncreaseCapitalBy
TradingStock::SetLong
TradingStock::SetShort

Here is the caller graph for this function:



5.5.3.16 recordDailyStatistics()

```
void Simulator::recordDailyStatistics ( )  [private]
```

Function to record daily statistics to a file.

Simulator::recordDailyStatistics (p. **??**) Function to record daily statistics to daily stats file.

Author

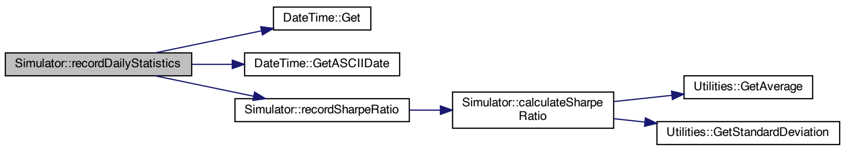
Salil Maharjan

Date

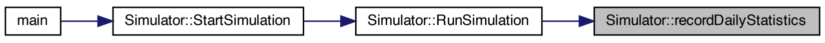
5/12/20.

Definition at line 742 of file Simulator.cpp.

Here is the call graph for this function:


DateTime::Get
Simulator::recordDailyStatistics
DateTime::GetASCIIDate
Simulator::recordSharpeRatio
Simulator::calculateSharpe Ratio
Utilities::GetAverage
Utilities::GetStandardDeviation

Here is the caller graph for this function:



5.5.3.17 recordMonthlyStatistics()

```
void Simulator::recordMonthlyStatistics ( )  [private]
```

Function to record monthly statistics to a file.

Simulator::recordMonthlyStatistics (p. **??**) Function to record monthly statistics to a file. Uses the generated daily publication to generate a monthly report.

Author

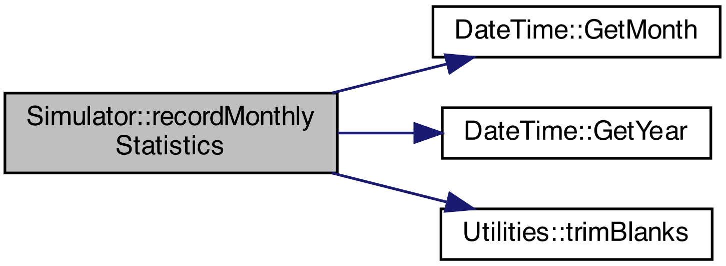
Salil Maharjan

Date

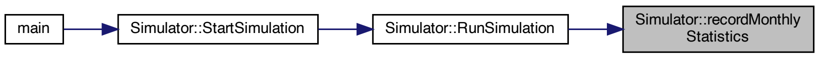
5/12/20.

Definition at line 770 of file Simulator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordMonthly Statistics

5.5.3.18 recordNoActivity()

```
void Simulator::recordNoActivity (
              TradingStock & a_stock )  [private]
```

Method to record necessary elements for no activity (code reuse purposes).

Simulator::recordNoActivity (p. **??**) Method to record necessary elements for no activity (code reuse purposes).

Parameters

a_stock	**TradingStock** (p. **??**)& The stock that is being traded.

Author

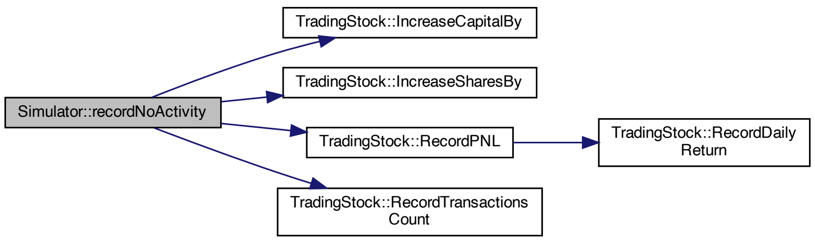
Salil Maharjan

Date

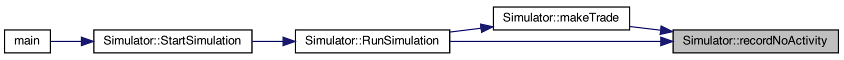
5/12/20.

Definition at line 571 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::recordNoActivity
TradingStock::IncreaseCapitalBy
TradingStock::IncreaseSharesBy
TradingStock::RecordPNL
TradingStock::RecordDaily Return
TradingStock::RecordTransactions Count

Here is the caller graph for this function:



5.5.3.19 recordSharpeRatio()

```
void Simulator::recordSharpeRatio ( )  [private]
```

Function to record sharpe ratio to daily statistics file.

Simulator::recordSharpeRatio (p. **??**) Method to record sharpe ratio to statistics file. Uses calculateSharpeRatio function to calculate sharpe ratio for the entire model.

Author

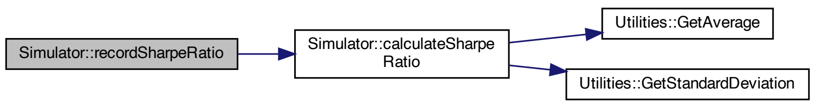
Salil Maharjan

Date

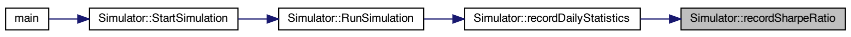
5/12/20.

Definition at line 683 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::recordSharpeRatio
Simulator::calculateSharpe Ratio
Utilities::GetAverage
Utilities::GetStandardDeviation

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordDailyStatistics
Simulator::recordSharpeRatio

5.5.3.20 recordSignalInfo()

```
void Simulator::recordSignalInfo (
            double a_signal )  [private]
```

Function to record signals to a file.

Simulator::recordSignalInfo (p. **??**) Function to record signals to a file.

Parameters

a_signal	The day's signal

Author

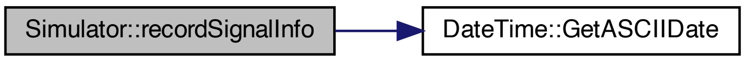
Salil Maharjan

Date

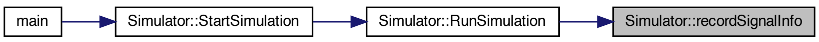
5/12/20.

Definition at line 730 of file Simulator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.21 recordTransaction()

```
void Simulator::recordTransaction (
              TradingStock & a_stock )  [private]
```

Function to record transactions to a file.

Simulator::recordTransaction (p. **??**) Function to record closed transactions to the transaction report file.

Parameters

a_stock	**TradingStock** (p. **??**)& The stock that is being traded.

Author

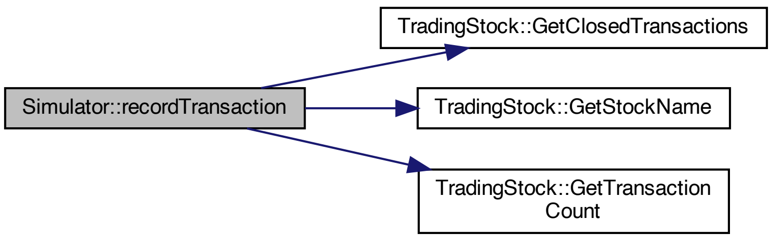
Salil Maharjan

Date

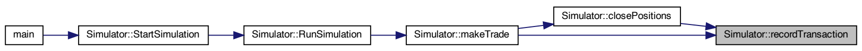
5/12/20.

Definition at line 704 of file Simulator.cpp.

Here is the call graph for this function:


TradingStock::GetClosedTransactions
Simulator::recordTransaction
TradingStock::GetStockName
TradingStock::GetTransaction Count

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
Simulator::recordTransaction

5.5.3.22 rerunSimulator()

```
void Simulator::rerunSimulator (
             const  char * a_configFile )  [private]
```

Rerun simulator for consecutive configuration files.

Simulator::rerunSimulator (p. **??**) Rerun simulator for consecutive configuration files

Parameters

a_configFile	char∗ **Config** (p. **??**) file to initialize simulator for.

Author

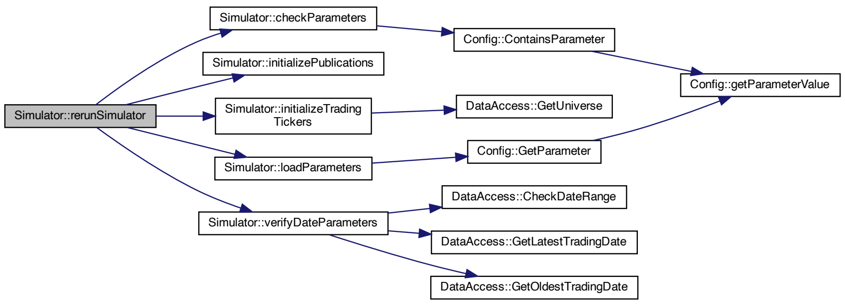
Salil Maharjan

Date

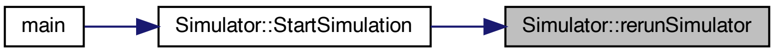
5/12/20.

Definition at line 246 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::rerunSimulator
Simulator::checkParameters
Simulator::initializePublications
Simulator::initializeTrading Tickers
Simulator::loadParameters
Simulator::verifyDateParameters
Config::ContainsParameter
Config::getParameterValue
DataAccess::GetUniverse
Config::GetParameter
DataAccess::CheckDateRange
DataAccess::GetLatestTradingDate
DataAccess::GetOldestTradingDate

Here is the caller graph for this function:



5.5.3.23 RunSimulation()

```
void Simulator::RunSimulation ( )
```

Main method to run simulation.

Simulator::RunSimulation (p. **??**) Main method to run the simulation.

Author

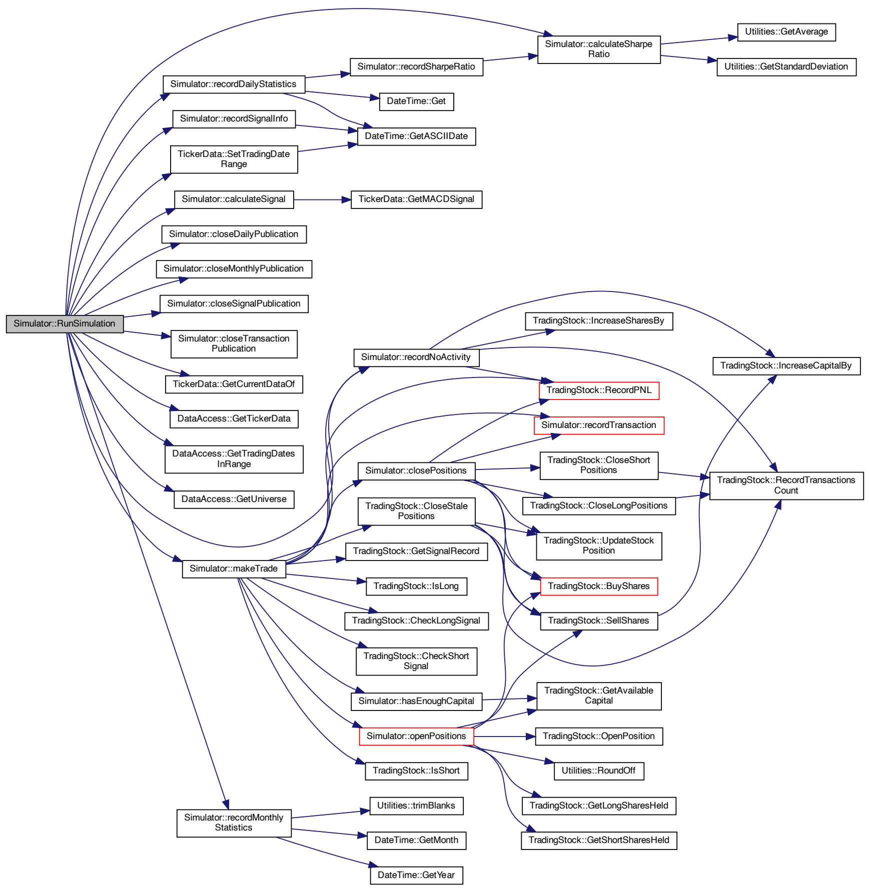
Salil Maharjan

Date

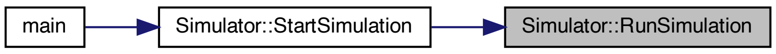
5/12/20.

Definition at line 75 of file Simulator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.24 StartSimulation()

```
void Simulator::StartSimulation ( )
```

Interface method to start the simulation.

Simulator::StartSimulation (p. **??**) Interface method to start the simulation

Author

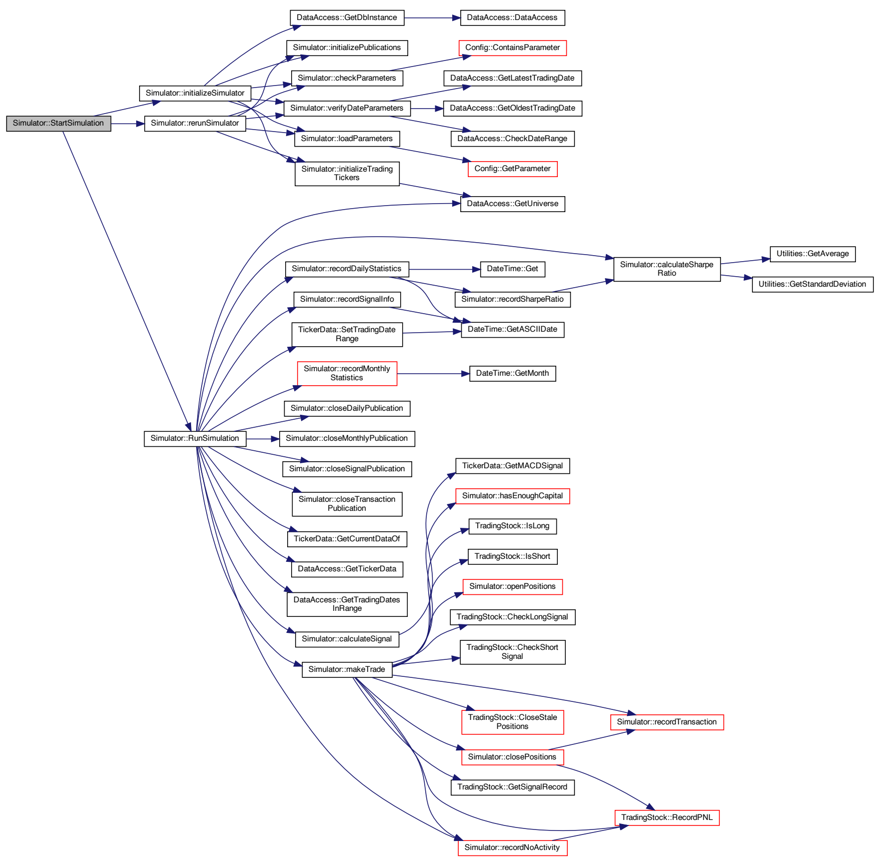
Salil Maharjan

Date

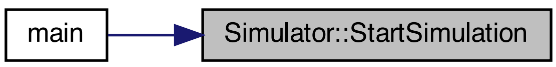
5/12/20.

Definition at line 52 of file Simulator.cpp.

Here is the call graph for this function:


Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
DataAccess::GetDbInstance
DataAccess::DataAccess
Simulator::initializePublications
Config::ContainsParameter
Simulator::checkParameters
DataAccess::GetLatestTradingDate
Simulator::verifyDateParameters
DataAccess::GetOldestTradingDate
Simulator::loadParameters
DataAccess::CheckDateRange
Simulator::initializeTrading Tickers
Config::GetParameter
DataAccess::GetUniverse
Simulator::calculateSharpe Ratio
Utilities::GetAverage
Utilities::GetStandardDeviation
Simulator::recordDailyStatistics
DateTime::Get
Simulator::recordSignalInfo
Simulator::recordSharpeRatio
TickerData::SetTradingDate Range
DateTime::GetASCIIDate
Simulator::recordMonthly Statistics
DateTime::GetMonth
Simulator::closeDailyPublication
Simulator::RunSimulation
Simulator::closeMonthlyPublication
Simulator::closeSignalPublication
TickerData::GetMACDSignal
Simulator::hasEnoughCapital
Simulator::closeTransaction Publication
TradingStock::IsLong
TickerData::GetCurrentDataOf
TradingStock::IsShort
DataAccess::GetTickerData
Simulator::openPositions
DataAccess::GetTradingDates InRange
TradingStock::CheckLongSignal
Simulator::calculateSignal
TradingStock::CheckShort Signal
Simulator::makeTrade
TradingStock::CloseStale Positions
Simulator::recordTransaction
Simulator::closePositions
TradingStock::GetSignalRecord
TradingStock::RecordPNL
Simulator::recordNoActivity

Here is the caller graph for this function:



5.5.3.25 verifyDateParameters()

```
void Simulator::verifyDateParameters ( )  [private]
```

Verify date range.

Simulator::verifyDateParameters (p. **??**) Verify date range in configuration file to assert if data is available for that range. If not, the simulation will run for all available data instead of stopping.

Author

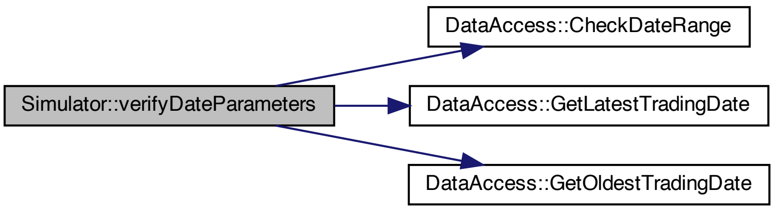
Salil Maharjan

Date

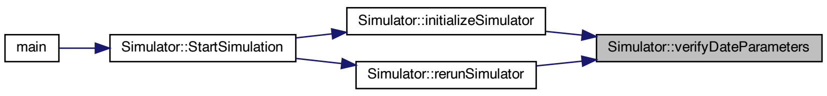
5/12/20.

Definition at line 447 of file Simulator.cpp.

Here is the call graph for this function:


DataAccess::CheckDateRange
Simulator::verifyDateParameters
DataAccess::GetLatestTradingDate
DataAccess::GetOldestTradingDate

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::initializeSimulator
Simulator::rerunSimulator
Simulator::verifyDateParameters

5.5.4 Member Data Documentation

5.5.4.1 DAILY_PUB_PARAM

```
const std::string Simulator::DAILY_PUB_PARAM = "DAILY"  [private]
```

5.5.5 [PUBLICATION PARAMETERS]

Definition at line 65 of file Simulator.h.

5.5.5.1 DAYS_IN_POS_PARAM

```
const std::string Simulator::DAYS_IN_POS_PARAM = "MAX_DAYS_IN_POSITION"  [private]
```

Definition at line 62 of file Simulator.h.

5.5.5.2 ENTRY_SIGNAL_PARAM

```
const std::string Simulator::ENTRY_SIGNAL_PARAM = "ENTRY_THRESHOLD"  [private]
```

5.5.6 [SIGNAL THRESHOLDS]

Definition at line 58 of file Simulator.h.

5.5.6.1 EXIT_SIGNAL_PARAM

```
const std::string Simulator::EXIT_SIGNAL_PARAM = "EXIT_THRESHOLD"  [private]
```

Definition at line 59 of file Simulator.h.

5.5.6.2 m_capitalLimitPerStock

```
double Simulator::m_capitalLimitPerStock  [private]
```

Definition at line 83 of file Simulator.h.

5.5.6.3 m_config

```
Config* Simulator::m_config  [private]
```

Configuration object.

Definition at line 163 of file Simulator.h.

5.5.6.4 m_configFile

```
std::vector<const char*> Simulator::m_configFile [private]
```

Var with config file name.

Definition at line 161 of file Simulator.h.

5.5.6.5 m_configFilesPassed

```
int Simulator::m_configFilesPassed [private]
```

Number of config files passed.

Definition at line 157 of file Simulator.h.

5.5.6.6 m_dailyCumulativeROR

```
std::vector<double> Simulator::m_dailyCumulativeROR [private]
```

Daily cumulative ROR record for all trading stocks in simulation.

Definition at line 147 of file Simulator.h.

5.5.6.7 m_dailyInfoFile

```
std::ofstream Simulator::m_dailyInfoFile [private]
```

Output streams to write publications.

Definition at line 108 of file Simulator.h.

5.5.6.8 m_dailyPNL

```
double Simulator::m_dailyPNL [private]
```

The profit or loss in dollars for the day.

Definition at line 125 of file Simulator.h.

5.5.6.9 m_dailyPNLRecord

```
std::vector<double> Simulator::m_dailyPNLRecord  [private]
```

Daily PNL record.

Definition at line 145 of file Simulator.h.

5.5.6.10 m_dailyPublication

```
bool Simulator::m_dailyPublication  [private]
```

5.5.7 [PUBLICATION PARAMETERS]

Definition at line 95 of file Simulator.h.

5.5.7.1 m_data

```
DataAccess* Simulator::m_data  [private]
```

Data access object.

Definition at line 165 of file Simulator.h.

5.5.7.2 m_endDate

```
DateTime* Simulator::m_endDate  [private]
```

Definition at line 89 of file Simulator.h.

5.5.7.3 m_entrySignal

```
double Simulator::m_entrySignal  [private]
```

5.5.8 [SIGNAL THRESHOLDS]

Definition at line 85 of file Simulator.h.

5.5.8.1 m_exitSignal

```
double Simulator::m_exitSignal  [private]
```

Definition at line 86 of file Simulator.h.

5.5.8.2 m_grossMarketValue

```
double Simulator::m_grossMarketValue  [private]
```

Long capital minus the short capital at the end of the day.

Definition at line 135 of file Simulator.h.

5.5.8.3 m_longPosCount

```
double Simulator::m_longPosCount  [private]
```

Daily long position counter variable.

Definition at line 127 of file Simulator.h.

5.5.8.4 m_maxDaysPerPosition

```
int Simulator::m_maxDaysPerPosition  [private]
```

Definition at line 92 of file Simulator.h.

5.5.8.5 m_maxPositionsPerStock

```
int Simulator::m_maxPositionsPerStock  [private]
```

5.5.9 [TRADING PARAMETERS]

Definition at line 91 of file Simulator.h.

5.5.9.1 m_monthlyInfoFile

```
std::ofstream Simulator::m_monthlyInfoFile  [private]
```

Definition at line 109 of file Simulator.h.

5.5.9.2 m_monthlyPublication

```
bool Simulator::m_monthlyPublication  [private]
```

Definition at line 96 of file Simulator.h.

5.5.9.3 m_netMarketValue

```
double Simulator::m_netMarketValue  [private]
```

Total amount of capital committed to the model at the end of the day.

Definition at line 133 of file Simulator.h.

5.5.9.4 m_portfolioCapital

```
double Simulator::m_portfolioCapital  [private]
```

5.5.10 [PORTFOLIO DATA]

Definition at line 82 of file Simulator.h.

5.5.10.1 m_ROR

```
double Simulator::m_ROR  [private]
```

Daily return. The profit for the day divided by the capital committed.

Definition at line 121 of file Simulator.h.

5.5.10.2 m_sharpeRecord

```
std::vector<double> Simulator::m_sharpeRecord  [private]
```

Sharpe ratio record.

Definition at line 149 of file Simulator.h.

5.5.10.3 m_shortPosCount

```
double Simulator::m_shortPosCount  [private]
```

Daily short position counter variable.

Definition at line 129 of file Simulator.h.

5.5.10.4 m_signalInfoFile

```
std::ofstream Simulator::m_signalInfoFile  [private]
```

Definition at line 111 of file Simulator.h.

5.5.10.5 m_signalPublication

```
bool Simulator::m_signalPublication  [private]
```

Definition at line 98 of file Simulator.h.

5.5.10.6 m_simulationCount

```
int Simulator::m_simulationCount  [private]
```

Count of how many config files have been simulated.

Definition at line 159 of file Simulator.h.

5.5.10.7 m_startDate

```
DateTime* Simulator::m_startDate  [private]
```

5.5.11 [SIMULATION PERIOD]

Definition at line 88 of file Simulator.h.

5.5.11.1 m_stopLoss

```
double Simulator::m_stopLoss  [private]
```

Definition at line 93 of file Simulator.h.

5.5.11.2 m_tickerPath

```
std::string Simulator::m_tickerPath  [private]
```

Definition at line 80 of file Simulator.h.

5.5.11.3 m_today

```
DateTime* Simulator::m_today  [private]
```

Date today.

Definition at line 137 of file Simulator.h.

5.5.11.4 m_totalPNL

```
double Simulator::m_totalPNL  [private]
```

Cumulative PNL from the beginning of the simulation to the current date.

Definition at line 123 of file Simulator.h.

5.5.11.5 m_totalPosCount

```
double Simulator::m_totalPosCount  [private]
```

Daily total position counter variable.

Definition at line 131 of file Simulator.h.

5.5.11.6 m_tradingStocks

```
std::vector< TradingStock*> Simulator::m_tradingStocks  [private]
```

All the trading stocks in the simulation.

Definition at line 167 of file Simulator.h.

5.5.11.7 m_transactionInfoFile

```
std::ofstream Simulator::m_transactionInfoFile  [private]
```

Definition at line 110 of file Simulator.h.

5.5.11.8 m_transactionPublication

```
bool Simulator::m_transactionPublication  [private]
```

Definition at line 97 of file Simulator.h.

5.5.11.9 m_universePath

```
std::string Simulator::m_universePath  [private]
```

Member variables from the configuration file

5.5.12 [DIRECTORY DATA]

Definition at line 79 of file Simulator.h.

5.5.12.1 MAX_CAPITAL_PER_STOCK_PARAM

```
const std::string Simulator::MAX_CAPITAL_PER_STOCK_PARAM = "MAX_CAPITAL_PER_STOCK"  [private]
```

Definition at line 53 of file Simulator.h.

5.5.12.2 MAX_POS_PER_STOCK

```
const std::string Simulator::MAX_POS_PER_STOCK = "MAX_POSITIONS_PER_STOCK"  [private]
```

5.5.13 [TRADING PARAMETERS]

Definition at line 61 of file Simulator.h.

5.5.13.1 MONTHLY_PUB_PARAM

```
const std::string Simulator::MONTHLY_PUB_PARAM = "MONTHLY"  [private]
```

Definition at line 66 of file Simulator.h.

5.5.13.2 PERIOD_END_DATE

```
const std::string Simulator::PERIOD_END_DATE = "END_DATE"  [private]
```

Definition at line 56 of file Simulator.h.

5.5.13.3 PERIOD_START_DATE

```
const std::string Simulator::PERIOD_START_DATE = "START_DATE"  [private]
```

5.5.14 [SIMULATION PERIOD]

Definition at line 55 of file Simulator.h.

5.5.14.1 PORTFOLIO_CAPITAL

```
const std::string Simulator::PORTFOLIO_CAPITAL = "PORTFOLIO_CAPITAL"  [private]
```

5.5.15 [PORTFOLIO DATA]

Definition at line 52 of file Simulator.h.

5.5.15.1 SIGNAL_PUB_PARAM

```
const std::string Simulator::SIGNAL_PUB_PARAM = "SIGNAL"  [private]
```

Definition at line 68 of file Simulator.h.

5.5.15.2 STOP_LOSS_PARAM

```
const std::string Simulator::STOP_LOSS_PARAM = "STOP_LOSS"  [private]
```

Definition at line 63 of file Simulator.h.

5.5.15.3 TICKER_PATH_PARAM

```
const std::string Simulator::TICKER_PATH_PARAM = "TICKER_DATA_DIRECTORY"  [private]
```

Definition at line 50 of file Simulator.h.

5.5.15.4 TRANSACTION_PUB_PARAM

```
const std::string Simulator::TRANSACTION_PUB_PARAM = "TRANSACTION"  [private]
```

Definition at line 67 of file Simulator.h.

5.5.15.5 UNIVERSE_PATH_PARAM

```
const std::string Simulator::UNIVERSE_PATH_PARAM = "UNIVERSE_DIRECTORY"  [private]
```

5.5.16 [DIRECTORY DATA]

Definition at line 49 of file Simulator.h.

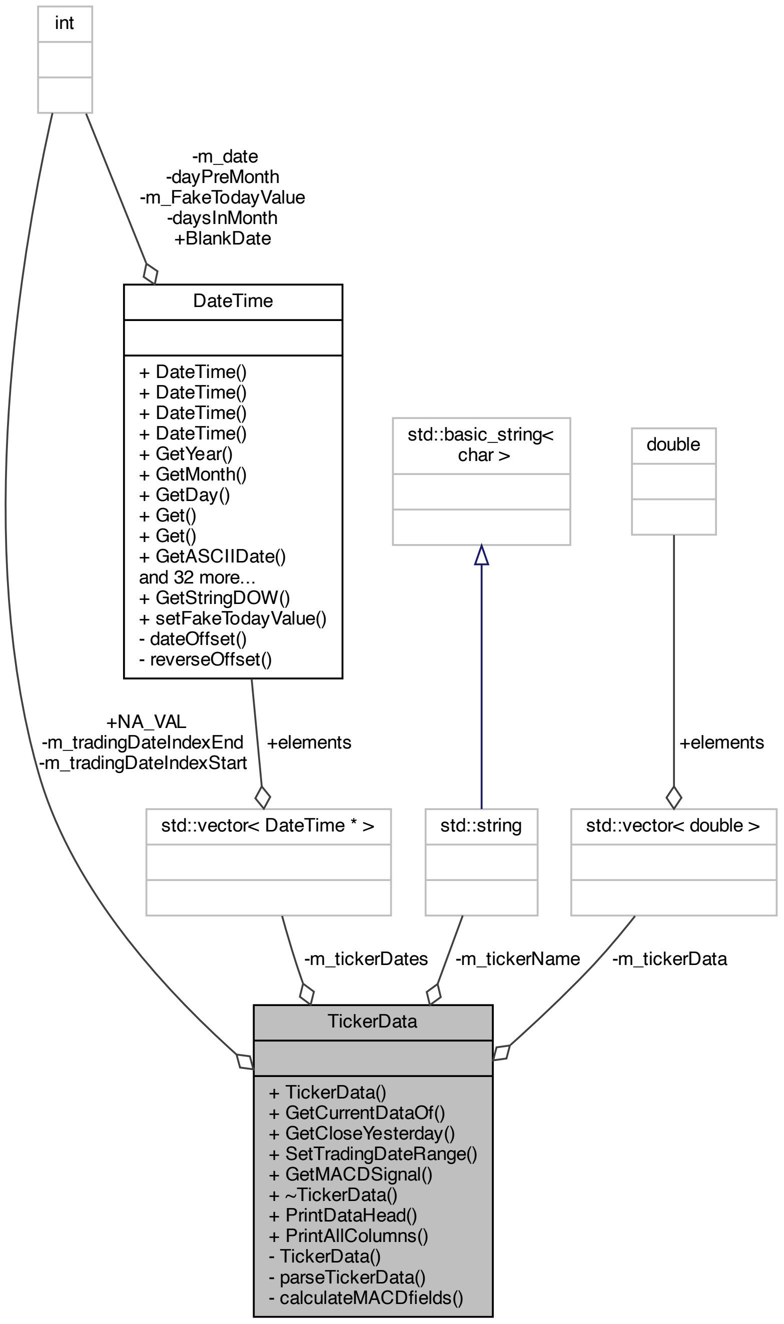
The documentation for this class was generated from the following files:

- **Simulator.h**
- **Simulator.cpp**

5.6 TickerData Class Reference

```
#include <TickerData.h>
```

Collaboration diagram for TickerData:



Public Member Functions

- **TickerData** (std::string &a_tickerName, const std::string &a_directory)

Parameterized constructor to create a ticker object.

- double **GetCurrentDataOf** (**TICKER_FIELDS** a_field)

  Get today's data of a_field.
- double **GetCloseYesterday** ()

  Get yesterday's close price.
- void **SetTradingDateRange** (**DateTime** ∗a_startDate, **DateTime** ∗a_endDate)

  Method to set trading date range.
- double **GetMACDSignal** ()

  Get current date's MACD signal.
- ∼**TickerData** ()=default

  Default deconstructor.
- void **PrintDataHead** ()

  Print head of data columns.
- void **PrintAllColumns** ()

  Print sample of all columns including data and calculated columns.

Static Public Attributes

- static const **int** **NA_VAL** = -999

  Fill constant for unavailable data.

Private Member Functions

- **TickerData** ()=default

  Default constructor.
- void **parseTickerData** (const std::string &a_directory)

  Parse ticker data from data source files.
- void **calculateMACDfields** ()

  Calculate MACD fields and append to the ticker data.

Private Attributes

- std::string **m_tickerName**

  Stock ticker name.
- std::vector< double > **m_tickerData** [**TICKER_FIELDS::END_TICKER_FIELDS**]

  ***TickerData** (p. **??**) data for a stock in memory.*
- std::vector< **DateTime** ∗ > **m_tickerDates**

  ***TickerData** (p. **??**) dates of the price data.*
- **int** **m_tradingDateIndexStart**

  Trading start date index in m_tickerDates.
- **int** **m_tradingDateIndexEnd**

  Trading end date index in m_tickerDates.

5.6.1 Detailed Description

Definition at line 55 of file TickerData.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 TickerData() [1/2]

```
TickerData::TickerData (
            std::string & a_tickerName,
            const std::string & a_directory )
```

Parameterized constructor to create a ticker object.

TickerData::TickerData (p. **??**). Parameterized constructor to create a ticker object

Parameters

a_tickerName	string Name of ticker
a_directory	string Price data directory path.

Author

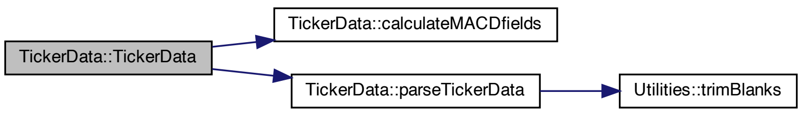
Salil Maharjan

Date

4/30/20.

Definition at line 28 of file TickerData.cpp.

Here is the call graph for this function:



5.6.2.2 ∼TickerData()

```
TickerData::∼TickerData ( )  [default]
```

Default deconstructor.

5.6.2.3 TickerData() `[2/2]`

```
TickerData::TickerData ( )  [private], [default]
```

Default constructor.

5.6.3 Member Function Documentation

5.6.3.1 calculateMACDfields()

```
void TickerData::calculateMACDfields ( )  [private]
```

Calculate MACD fields and append to the ticker data.

TickerData::calculateMACDfields (p. **??**) Calculate MACD fields and append to the ticker data.

Author

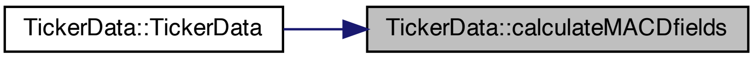
Salil Maharjan

Date

4/30/20.

Definition at line 159 of file TickerData.cpp.

Here is the caller graph for this function:



5.6.3.2 GetCloseYesterday()

```
double TickerData::GetCloseYesterday ( )  [inline]
```

Get yesterday's close price.

Definition at line 80 of file TickerData.h.

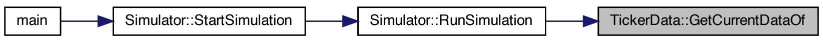
5.6.3.3 GetCurrentDataOf()

```
double TickerData::GetCurrentDataOf (
              TICKER_FIELDS a_field )  [inline]
```

Get today's data of a_field.

Definition at line 77 of file TickerData.h.

Here is the caller graph for this function:



5.6.3.4 GetMACDSignal()

```
double TickerData::GetMACDSignal ( )
```

Get current date's MACD signal.

TickerData::GetMACDSignal (p. **??**). Get current date's MACD signal.

Returns

double The current MACD signal.

Author

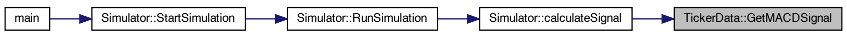
Salil Maharjan

Date

4/30/20.

Definition at line 76 of file TickerData.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::calculateSignal
TickerData::GetMACDSignal

5.6.3.5 parseTickerData()

```
void TickerData::parseTickerData (
             const std::string & a_directory )  [private]
```

Parse ticker data from data source files.

TickerData::parseTickerData (p. **??**). Parse ticker data from data source files.

Parameters

a_directory	string Path where the price data is located.

Author

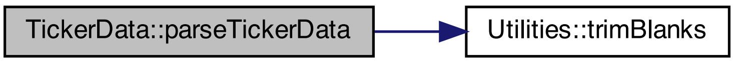
Salil Maharjan

Date

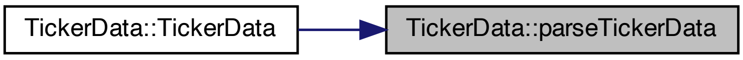
4/30/20.

Definition at line 105 of file TickerData.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.6 PrintAllColumns()

```
void TickerData::PrintAllColumns ( )
```

Print sample of all columns including data and calculated columns.

TickerData::PrintAllColumns (p. **??**) Method to print a sample of data for all data and calculated columns. Used for debugging and checking.

Author

Salil Maharjan

Date

4/30/20.

Definition at line 305 of file TickerData.cpp.

5.6.3.7 PrintDataHead()

```
void TickerData::PrintDataHead ( )
```

Print head of data columns.

TickerData::PrintDataHead (p. **??**) Method to print first 10 data entries for the ticker object. Only print data fields. Used for debugging and checking.

Author

Salil Maharjan

Date

4/30/20.

Definition at line 261 of file TickerData.cpp.

5.6.3.8 SetTradingDateRange()

```
void TickerData::SetTradingDateRange (
              DateTime * a_startDate,
              DateTime * a_endDate )
```

Method to set trading date range.

TickerData::SetTradingDateRange (p. **??**). Method to set trading date range.

Parameters

a_startDate	DateTime∗ Start date of simulation to set.
a_endDate	DateTime∗ End date of simulation to set.

Author

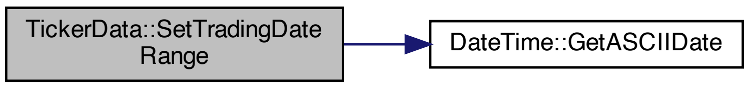
Salil Maharjan

Date

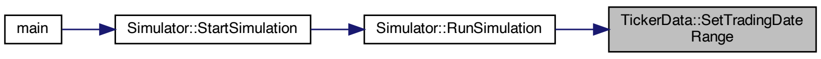
4/30/20.

Definition at line 47 of file TickerData.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.4 Member Data Documentation

5.6.4.1 m_tickerData

`std::vector<double> TickerData::m_tickerData[ TICKER_FIELDS::END_TICKER_FIELDS] [private]`

TickerData (p. **??**) data for a stock in memory.

Definition at line 117 of file TickerData.h.

5.6.4.2 m_tickerDates

`std::vector< DateTime*> TickerData::m_tickerDates [private]`

TickerData (p. **??**) dates of the price data.

Definition at line 119 of file TickerData.h.

5.6.4.3 m_tickerName

`std::string TickerData::m_tickerName [private]`

Stock ticker name.

Definition at line 115 of file TickerData.h.

5.6.4.4 m_tradingDateIndexEnd

```
int TickerData::m_tradingDateIndexEnd  [private]
```

Trading end date index in m_tickerDates.

Definition at line 123 of file TickerData.h.

5.6.4.5 m_tradingDateIndexStart

```
int TickerData::m_tradingDateIndexStart  [private]
```

Trading start date index in m_tickerDates.

Definition at line 121 of file TickerData.h.

5.6.4.6 NA_VAL

```
const  int TickerData::NA_VAL = -999  [static]
```

Fill constant for unavailable data.

TickerData.cpp (p. **??**) Implementation of **TickerData.h** (p. **??**)

Created by Salil Maharjan on 4/30/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 63 of file TickerData.h.

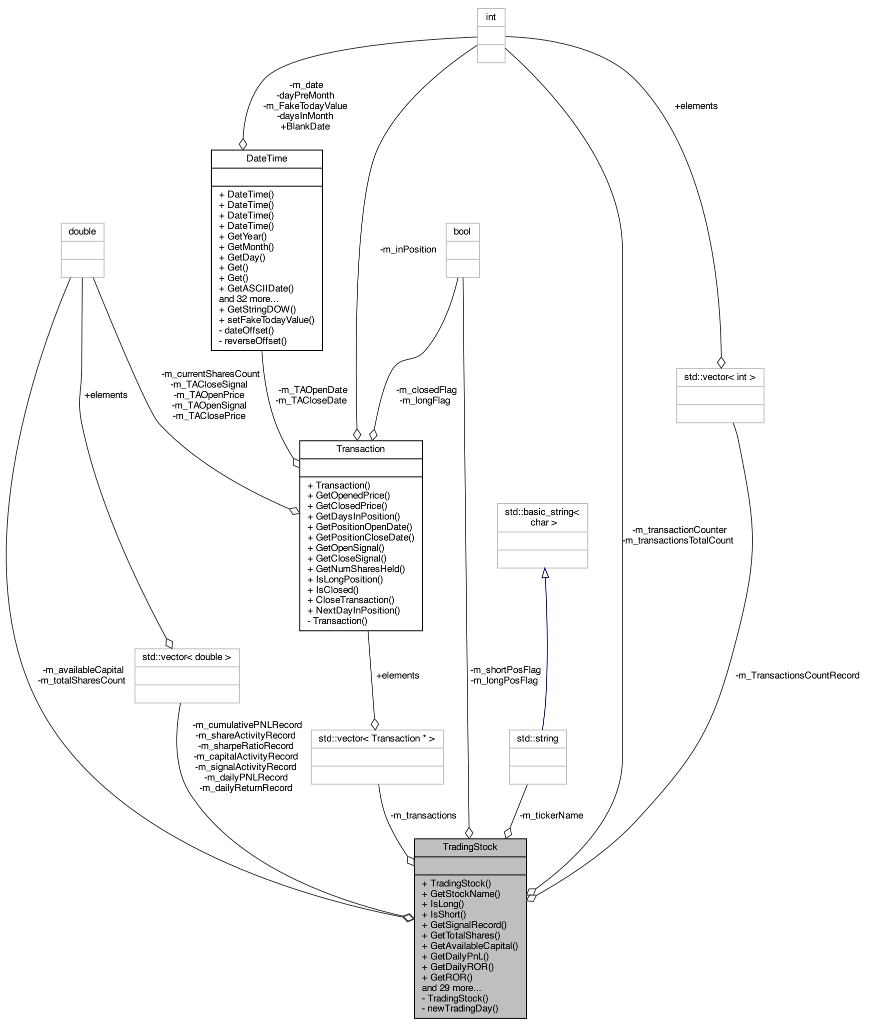
The documentation for this class was generated from the following files:

- **TickerData.h**
- **TickerData.cpp**

5.7 TradingStock Class Reference

```
#include <TradingStock.h>
```

Collaboration diagram for TradingStock:



Public Member Functions

- **TradingStock** (std::string a_tickerName, double a_startingCapital)

  Class parameterized constructor.
- const std::string **GetStockName** ()
- const bool **IsLong** ()
- const bool **IsShort** ()
- const std::vector< double > **GetSignalRecord** ()
- const double **GetTotalShares** ()
- const double **GetAvailableCapital** ()
- const double **GetDailyPnL** ()
- const double **GetDailyROR** ()
- const std::vector< double > **GetROR** ()

- const double **GetLatestShareActivity** ()
- const double **GetSharpeToday** ()
- double **GetTransactionCount** ()

  Method to get current completed transaction count (Does not consider incomplete transactions.)
- const double **GetShortSharesHeld** ()

  Method to get short shares held.
- const double **GetLongSharesHeld** ()

  Method to get long shares held.
- std::vector$<$ **Transaction** $*$ $>$ **GetClosedTransactions** ()

  Method to get closed transactions and remove them from memory.
- const double **GetShortCapital** ()

  Method to get short capital in investment.
- const double **GetLongCapital** ()

  Method to get long capital in investment.
- void **SetLong** (bool a_long)
- void **SetShort** (bool a_short)
- void **IncreaseSharesBy** (double a_share)
- void **DecreaseSharesBy** (double a_share)
- void **IncreaseCapitalBy** (double a_share)
- void **DecreaseCapitalBy** (double a_share)
- void **SellShares** (double a_shares, double a_price)

  Method to sell shares.
- void **BuyShares** (double a_shares, double a_price)

  Method to buys shares.
- void **CloseStalePositions** (**int** a_dayLimit, **DateTime** $*$a_date, double a_signal, double a_price)

  Method to close positions that exceed maximum days limit.
- void **OpenPosition** (**DateTime** $*$a_date, double a_signal, double a_share, double a_price)

  Method to open a position for the trading stock.
- double **CloseLongPositions** (**DateTime** $*$a_date, double a_signal, double a_price, double &a_invested↩ Captial, double &a_numOfShares)

  Method to close all long positions.
- double **CloseShortPositions** (**DateTime** $*$a_date, double a_signal, double a_price, double &a_invested↩ Captial, double &a_numOfShares)

  Method to close all short positions.
- bool **CheckShortSignal** (double a_price)

  Method to check if signal is profitable for short positions.
- bool **CheckLongSignal** (double a_price)

  Method to check if signal is profitable for long positions.
- void **UpdateStockPosition** ()

  Method to update long/short position status of the trading stock.
- void **RecordSignal** (double a_signal)

  Record day's signal.
- void **RecordPNL** (double a_pnl, double a_investedCapital)

  Record day's PNL.
- void **RecordTransactionsCount** (**int** a_count)

  Record day's number of transactions.
- void **RecordDailyReturn** (double a_amount)

  Recard day's daily return.
- double **GetInvestedCapital** ()

  Get latest committed capital in an investment.
- void **CalculateDailySharpeRatio** ()

  Calculate daily sharpe ratio for individual stock.

Private Member Functions

- **TradingStock** ()=default

  Private default constructor.
- void **newTradingDay** ()

  Method to update days in position of all held shares.

Private Attributes

- std::string **m_tickerName**

  Ticker name.
- double **m_availableCapital**

  Available capital for trading stock.
- **int m_transactionsTotalCount**

  Total transactions count.
- **int m_transactionCounter**

  Current complete transaction counter.
- double **m_totalSharesCount**

  Total shares held for stock.
- bool **m_longPosFlag**

  Long position flag.
- bool **m_shortPosFlag**

  Short position flag.
- std::vector< **Transaction** ∗ > **m_transactions**

  Records all transactions.
- std::vector< double > **m_signalActivityRecord**

  Records daily signals.
- std::vector< double > **m_capitalActivityRecord**

  Records daily capital activities.
- std::vector< double > **m_shareActivityRecord**

  Records daily position changes.
- std::vector< double > **m_dailyPNLRecord**

  Records daily PNL.
- std::vector< double > **m_dailyReturnRecord**

  Records daily rate of return (ROR)
- std::vector< double > **m_cumulativePNLRecord**

  Records daily cumulative PNL.
- std::vector< **int** > **m_TransactionsCountRecord**

  Records daily transaction counts.
- std::vector< double > **m_sharpeRatioRecord**

  Records daily sharpe ratio.

5.7.1 Detailed Description

TradingStock.h (p. **??**) Class that handles the investment portfolio of a trading stock in the simulator. Handles trading details for each ticker used in the simulator.

Created by Salil Maharjan on 05/03/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 18 of file TradingStock.h.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 TradingStock() [1/2]

```
TradingStock::TradingStock (
            std::string a_tickerName,
            double a_startingCapital )
```

Class parameterized constructor.

TradingStock.cpp (p. **??**) Implementation of **TradingStock.h** (p. **??**)

Created by Salil Maharjan on 05/03/20. Copyright © 2020 Salil Maharjan. All rights reserved. **TradingStock::↩ TradingStock** (p. **??**) Class parameterized constructor to create a trading stock object.

Parameters

a_tickerName	string Trading stock's ticker name.
a_startingCapital	double Initial capital to invest for the stock.

Author

Salil Maharjan

Date

5/14/20.

Definition at line 26 of file TradingStock.cpp.

5.7.2.2 TradingStock() [2/2]

```
TradingStock::TradingStock ( )  [private], [default]
```

Private default constructor.

5.7.3 Member Function Documentation

5.7.3.1 BuyShares()

```
void TradingStock::BuyShares (
            double a_shares,
            double a_price )
```

Method to buys shares.

TradingStock::BuyShares (p. **??**) Method to buy shares. Updates capital and record variables.

Parameters

a_shares	double Number of shares to buy.
a_price	double Price at which to buy shares.

Author

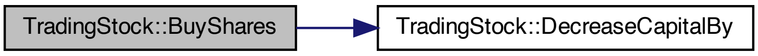
Salil Maharjan

Date

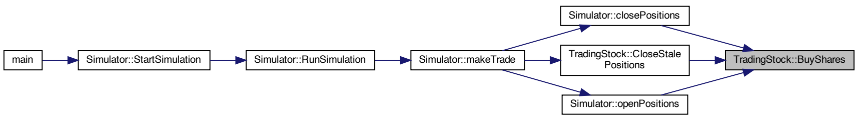
5/14/20.

Definition at line 63 of file TradingStock.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.3.2 CalculateDailySharpeRatio()

```
void TradingStock::CalculateDailySharpeRatio ( )
```

Calculate daily sharpe ratio for individual stock.

TradingStock::CalculateDailySharpeRatio (p. **??**) Calculate daily sharpe ratio for individual stock. Can be used to get individual stock's Sharpe ratio instead of the complete model's Sharpe.

Author

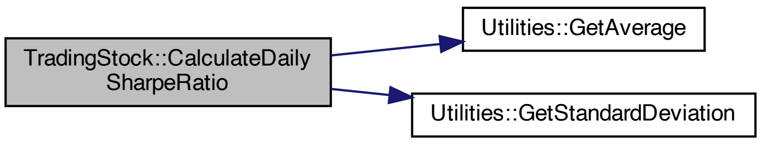
Salil Maharjan

Date

5/14/20.

Definition at line 456 of file TradingStock.cpp.

Here is the call graph for this function:


TradingStock::CalculateDaily SharpeRatio
Utilities::GetAverage
Utilities::GetStandardDeviation

5.7.3.3 CheckLongSignal()

```
bool TradingStock::CheckLongSignal (
             double a_price )
```

Method to check if signal is profitable for long positions.

TradingStock::CheckLongSignal (p. **??**) Method to check if signal is profitable for long positions.

Parameters

a_price	double Current price of shares.

Returns

bool If signal will be profitable.

Author

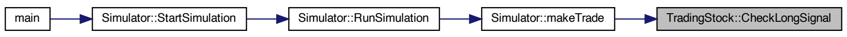
Salil Maharjan

Date

5/14/20.

Definition at line 324 of file TradingStock.cpp.

Here is the caller graph for this function:



5.7.3.4 CheckShortSignal()

```
bool TradingStock::CheckShortSignal (
             double a_price )
```

Method to check if signal is profitable for short positions.

TradingStock::CheckShortSignal (p. **??**) Method to check if signal is profitable for short positions.

Parameters

a_price	double Current price of shares.

Returns

bool If signal will be profitable.

Author

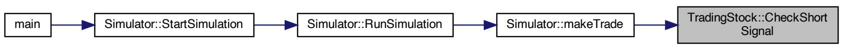
Salil Maharjan

Date

5/14/20.

Definition at line 304 of file TradingStock.cpp.

Here is the caller graph for this function:



5.7.3.5 CloseLongPositions()

```
double TradingStock::CloseLongPositions (
             DateTime * a_date,
             double a_signal,
             double a_price,
             double & a_investedCaptial,
             double & a_numOfShares )
```

Method to close all long positions.

TradingStock::CloseLongPositions (p. **??**) Method to close profiting long positions. Returns profit/loss generated from closing long positions. Updates invested capital in closing long positions and number of long positions that are variables passed by reference.

Parameters

a_date	DateTime∗ Today's date.
a_signal	double Current day's signal.
a_price	double Price of shares at the time of closing.
a_investedCaptial	double& Invested capital placeholder to return the invested capital in long positions.
a_numOfShares	double& Placeholder to return number of shares in long position that were closed.

Returns

double Profit or loss generated from closing long positions.

Author

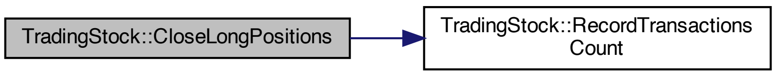
Salil Maharjan

Date

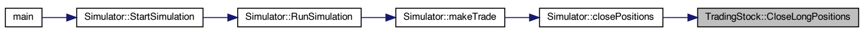
5/14/20.

Definition at line 215 of file TradingStock.cpp.

Here is the call graph for this function:


TradingStock::CloseLongPositions
TradingStock::RecordTransactions Count

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
TradingStock::CloseLongPositions

5.7.3.6 CloseShortPositions()

```
double TradingStock::CloseShortPositions (
              DateTime * a_date,
             double a_signal,
             double a_price,
             double & a_investedCaptial,
             double & a_numOfShares )
```

Method to close all short positions.

TradingStock::CloseShortPositions (p. **??**) Method to close profiting short positions. Returns profit/loss generated from closing short positions. Updates invested capital in closing short positions and number of short positions that are variables passed by reference.

Parameters

a_date	DateTime∗ Today's date.
a_signal	double Current day's signal.
a_price	double Price of shares at the time of closing.
a_investedCaptial	double& Invested capital placeholder to return the invested capital in long positions.
a_numOfShares	double& Placeholder to return number of shares in long position that were closed.

Returns

double Profit or loss generated from closing long positions.

Author

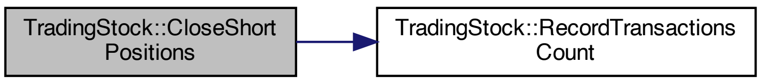
Salil Maharjan

Date

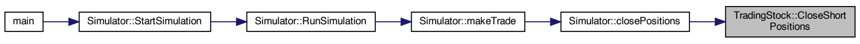
5/14/20.

Definition at line 262 of file TradingStock.cpp.

Here is the call graph for this function:


TradingStock::CloseShort Positions
TradingStock::RecordTransactions Count

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
TradingStock::CloseShort Positions

5.7.3.7 CloseStalePositions()

```
void TradingStock::CloseStalePositions (
              int a_dayLimit,
              DateTime * a_date,
              double a_signal,
              double a_price )
```

Method to close positions that exceed maximum days limit.

TradingStock::CloseStalePositions (p. **??**) Method to close positions that exceed maximum days limit.

Parameters

a_dayLimit	int Maximum days in position limit.
a_date	DateTime∗ Today's date.
a_signal	double Current day's signal.
a_price	double Current price of shares.

Author

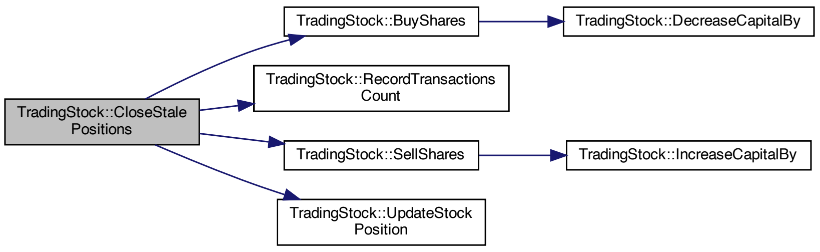
Salil Maharjan

Date

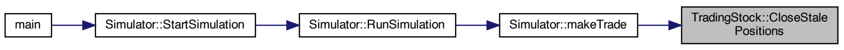
5/14/20.

Definition at line 371 of file TradingStock.cpp.

Here is the call graph for this function:


TradingStock::BuyShares
TradingStock::DecreaseCapitalBy
TradingStock::RecordTransactions Count
TradingStock::CloseStale Positions
TradingStock::SellShares
TradingStock::IncreaseCapitalBy
TradingStock::UpdateStock Position

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
TradingStock::CloseStale Positions

5.7.3.8 DecreaseCapitalBy()

```
void TradingStock::DecreaseCapitalBy (
             double a_share )
```

TradingStock::DecreaseCapitalBy (p. **??**) Function to decrease available capital in the stock and record it.

Parameters

a_capital	double Capital to decrease.

Author

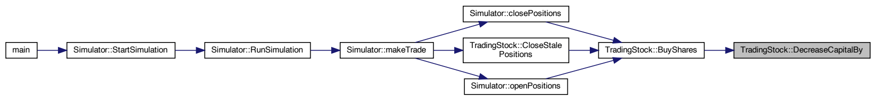
Salil Maharjan

Date

5/14/20.

Definition at line 131 of file TradingStock.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
TradingStock::CloseStale Positions
Simulator::openPositions
TradingStock::BuyShares
TradingStock::DecreaseCapitalBy

5.7.3.9 DecreaseSharesBy()

```
void TradingStock::DecreaseSharesBy (
             double a_share )
```

TradingStock::DecreaseSharesBy (p. **??**) Function to decrease shares owned in the stock.

Parameters

a_share	double Number of shares to remove.

Author

Salil Maharjan

Date

5/14/20.

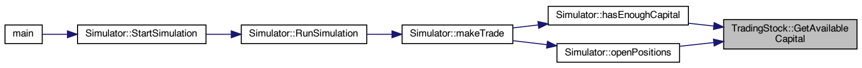
Definition at line 105 of file TradingStock.cpp.

5.7.3.10 GetAvailableCapital()

```
const double TradingStock::GetAvailableCapital ( )  [inline]
```

Definition at line 38 of file TradingStock.h.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::hasEnoughCapital
Simulator::openPositions
TradingStock::GetAvailable Capital

5.7.3.11 GetClosedTransactions()

```
std::vector< Transaction * > TradingStock::GetClosedTransactions ( )
```

Method to get closed transactions and remove them from memory.

TradingStock::GetClosedTransactions (p. **??**) Method to get closed transactions. Once returned, they are removed from the record.

Returns

double Total capital in long positions.

Author

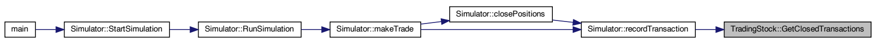
Salil Maharjan

Date

5/14/20.

Definition at line 570 of file TradingStock.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
Simulator::recordTransaction
TradingStock::GetClosedTransactions

5.7.3.12 GetDailyPnL()

```
const double TradingStock::GetDailyPnL ( )  [inline]
```

Definition at line 39 of file TradingStock.h.

5.7.3.13 GetDailyROR()

```
const double TradingStock::GetDailyROR ( )  [inline]
```

Definition at line 40 of file TradingStock.h.

5.7.3.14 GetInvestedCapital()

```
double TradingStock::GetInvestedCapital ( )
```

Get latest committed capital in an investment.

TradingStock::GetInvestedCapital (p. **??**) Get latest committed capital in an investment

Returns

double Latest committed capital activity.

Author

Salil Maharjan

Date

5/14/20.

Definition at line 436 of file TradingStock.cpp.

5.7.3.15 GetLatestShareActivity()

```
const double TradingStock::GetLatestShareActivity ( )  [inline]
```

Definition at line 43 of file TradingStock.h.

5.7.3.16 GetLongCapital()

```
const double TradingStock::GetLongCapital ( )
```

Method to get long capital in investment.

TradingStock::GetLongCapital (p. **??**) Accessor method to get capital invested in long positions in trading stock.

Returns

double Total capital in long positions.

Author

Salil Maharjan

Date

5/14/20.

Definition at line 549 of file TradingStock.cpp.

5.7.3.17 GetLongSharesHeld()

```
const double TradingStock::GetLongSharesHeld ( )
```

Method to get long shares held.

TradingStock::GetLongSharesHeld (p. **??**) Accessor method to get long shares held in the trading stock.

Returns

double Number of short positions held.

Author

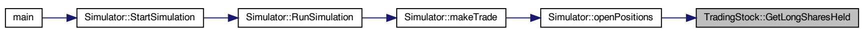
Salil Maharjan

Date

5/14/20.

Definition at line 507 of file TradingStock.cpp.

Here is the caller graph for this function:



5.7.3.18 GetROR()

```
const std::vector<double> TradingStock::GetROR ( )  [inline]
```

Definition at line 42 of file TradingStock.h.

5.7.3.19 GetSharpeToday()

```
const double TradingStock::GetSharpeToday ( )
```

TradingStock::GetSharpeToday (p. **??**) Accessor method to get the Sharpe ratio of the current day of an individual stock.

Author

Salil Maharjan

Date

5/14/20.

Definition at line 471 of file TradingStock.cpp.

5.7.3.20 GetShortCapital()

```
const double TradingStock::GetShortCapital ( )
```

Method to get short capital in investment.

TradingStock::GetShortCapital (p. **??**) Accessor method to get capital invested in short positions in trading stock.

Returns

double Total capital in short positions.

Author

Salil Maharjan

Date

5/14/20.

Definition at line 528 of file TradingStock.cpp.

5.7.3.21 GetShortSharesHeld()

```
const double TradingStock::GetShortSharesHeld ( )
```

Method to get short shares held.

TradingStock::GetShortSharesHeld (p. **??**) Accessor method to get short shares held in the trading stock.

Returns

double Number of short positions held.

Author

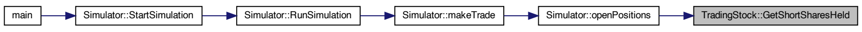
Salil Maharjan

Date

5/14/20.

Definition at line 486 of file TradingStock.cpp.

Here is the caller graph for this function:

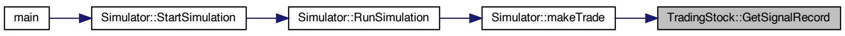


5.7.3.22 GetSignalRecord()

```
const std::vector<double> TradingStock::GetSignalRecord ( )  [inline]
```

Definition at line 36 of file TradingStock.h.

Here is the caller graph for this function:

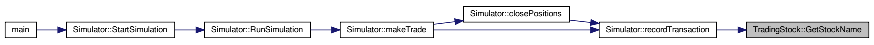


5.7.3.23 GetStockName()

```
const std::string TradingStock::GetStockName ( )  [inline]
```

Definition at line 33 of file TradingStock.h.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
Simulator::recordTransaction
TradingStock::GetStockName

5.7.3.24 GetTotalShares()

```
const double TradingStock::GetTotalShares ( )  [inline]
```

Definition at line 37 of file TradingStock.h.

5.7.3.25 GetTransactionCount()

```
double TradingStock::GetTransactionCount ( )
```

Method to get current completed transaction count (Does not consider incomplete transactions.)

TradingStock::GetTransactionCount (p. **??**) Function to get current closed transaction counter.

Returns

double Number of current completed transactions.

Author

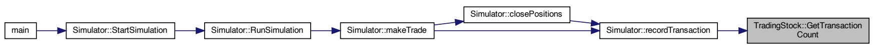
Salil Maharjan

Date

5/14/20.

Definition at line 594 of file TradingStock.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
Simulator::recordTransaction
TradingStock::GetTransaction Count

5.7.3.26 IncreaseCapitalBy()

```
void TradingStock::IncreaseCapitalBy (
            double a_capital )
```

TradingStock::IncreaseCapitalBy (p. **??**) Function to increase available capital in the stock and record it.

Parameters

a_capital	double Capital to increase.

Author

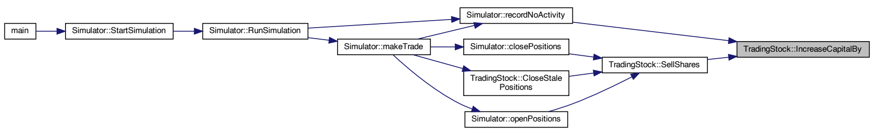
Salil Maharjan

Date

5/14/20.

Definition at line 118 of file TradingStock.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::recordNoActivity
Simulator::closePositions
TradingStock::CloseStale Positions
Simulator::openPositions
TradingStock::SellShares
TradingStock::IncreaseCapitalBy

5.7.3.27 IncreaseSharesBy()

```
void TradingStock::IncreaseSharesBy (
            double a_share )
```

TradingStock::IncreaseSharesBy (p. **??**) Function to increase shares owned in the stock.

Parameters

a_share	double Number of shares to add.

Author

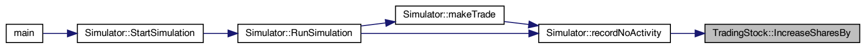
Salil Maharjan

Date

5/14/20.

Definition at line 92 of file TradingStock.cpp.

Here is the caller graph for this function:

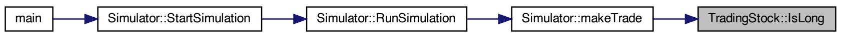

main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::recordNoActivity
TradingStock::IncreaseSharesBy

5.7.3.28 IsLong()

```
const bool TradingStock::IsLong ( ) [inline]
```

Definition at line 34 of file TradingStock.h.

Here is the caller graph for this function:

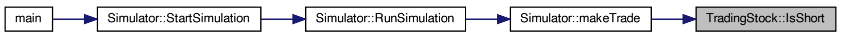


5.7.3.29 IsShort()

```
const bool TradingStock::IsShort ( ) [inline]
```

Definition at line 35 of file TradingStock.h.

Here is the caller graph for this function:



5.7.3.30 newTradingDay()

```
void TradingStock::newTradingDay ( )  [private]
```

Method to update days in position of all held shares.

TradingStock::newTradingDay (p. **??**) Method to update days in position of all held shares

Author

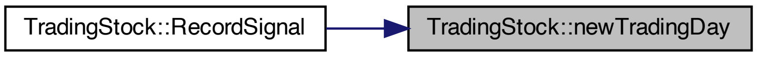
Salil Maharjan

Date

5/14/20.

Definition at line 610 of file TradingStock.cpp.

Here is the caller graph for this function:



5.7.3.31 OpenPosition()

```
void TradingStock::OpenPosition (
              DateTime * a_date,
             double a_signal,
             double a_share,
             double a_price )
```

Method to open a position for the trading stock.

TradingStock::OpenPosition (p. **??**) Open a new position for the trading stock.

Parameters

a_date	DateTime∗ Today's date.
a_signal	double Current day's signal.
a_share	double Number of shares to open.
a_price	double Price of shares at the time of opening.

Author

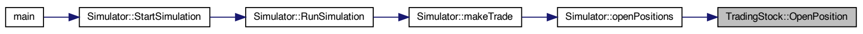
Salil Maharjan

Date

5/14/20.

Definition at line 197 of file TradingStock.cpp.

Here is the caller graph for this function:



5.7.3.32 RecordDailyReturn()

```
void TradingStock::RecordDailyReturn (
             double a_amount )
```

Recard day's daily return.

TradingStock::RecordDailyReturn (p. **??**) Record daily rate of return (ROR).

Parameters

a_amount	double Daily ROR.

Author

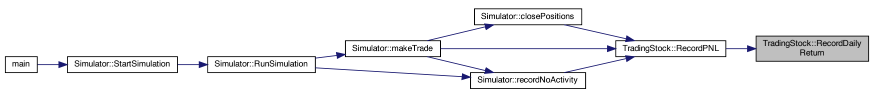
Salil Maharjan

Date

5/14/20.

Definition at line 182 of file TradingStock.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
Simulator::recordNoActivity
TradingStock::RecordPNL
TradingStock::RecordDaily Return

5.7.3.33 RecordPNL()

```
void TradingStock::RecordPNL (
            double a_pnl,
            double a_investedCapital )
```

Record day's PNL.

TradingStock::RecordPNL (p. **??**) Records PNL and calls RecordDailyReturn to record ROR. (coupled functions)

Parameters

a_pnl	double Profit or loss for the day.
a_investedCapital	double Invested capital in the position generating profit or loss.

Author

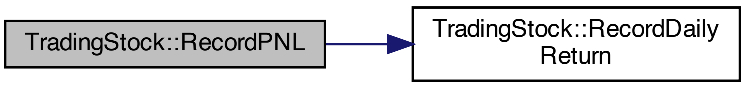
Salil Maharjan

Date

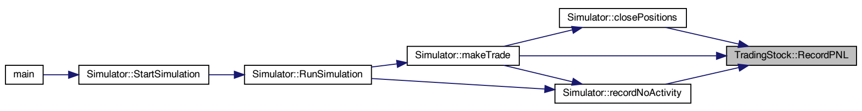
5/14/20.

Definition at line 145 of file TradingStock.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
Simulator::recordNoActivity
TradingStock::RecordPNL

5.7.3.34 RecordSignal()

```
void TradingStock::RecordSignal (
            double a_signal )
```

Record day's signal.

TradingStock::RecordSignal (p. **??**) Function to record the trading day signal and update the days in position of the stocks held.

Parameters

a_signal	double Day's trading signal.

Author

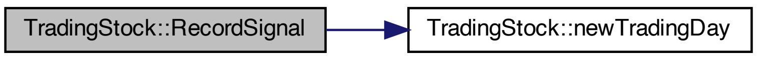
Salil Maharjan

Date

5/14/20.

Definition at line 79 of file TradingStock.cpp.

Here is the call graph for this function:



5.7.3.35 RecordTransactionsCount()

```
void TradingStock::RecordTransactionsCount (
              int a_count )
```

Record day's number of transactions.

TradingStock::RecordTransactionsCount (p. **??**) Record daily transaction count.

Parameters

a_count	int Daily transactions count.

Author

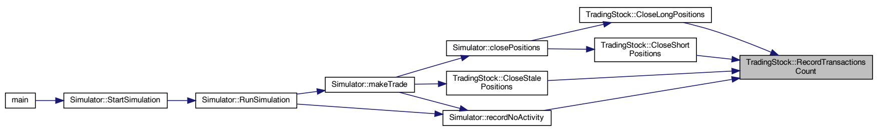
Salil Maharjan

Date

5/14/20.

Definition at line 169 of file TradingStock.cpp.

Here is the caller graph for this function:


TradingStock::CloseLongPositions
Simulator::closePositions
TradingStock::CloseShort Positions
TradingStock::RecordTransactions Count
Simulator::makeTrade
TradingStock::CloseStale Positions
main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::recordNoActivity

5.7.3.36 SellShares()

```
void TradingStock::SellShares (
             double a_shares,
             double a_price )
```

Method to sell shares.

TradingStock::SellShares (p. **??**) Method to sell shares. Updates capital and record variables.

Parameters

a_shares	double Number of shares to sell.
a_price	double Price at which to sell shares.

Author

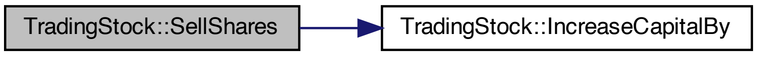
Salil Maharjan

Date

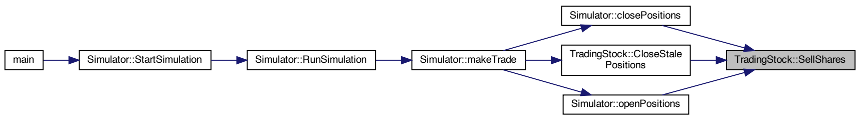
5/14/20.

Definition at line 46 of file TradingStock.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

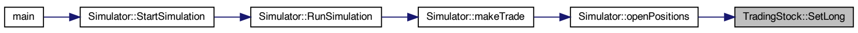


5.7.3.37 SetLong()

```
void TradingStock::SetLong (
             bool a_long )  [inline]
```

Definition at line 63 of file TradingStock.h.

Here is the caller graph for this function:

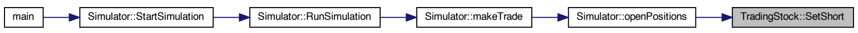


5.7.3.38 SetShort()

```
void TradingStock::SetShort (
             bool a_short )  [inline]
```

Definition at line 64 of file TradingStock.h.

Here is the caller graph for this function:



5.7.3.39 UpdateStockPosition()

```
void TradingStock::UpdateStockPosition ( )
```

Method to update long/short position status of the trading stock.

TradingStock::UpdateStockPosition (p. **??**) Method to update long/short position flags of the trading stock.

Author

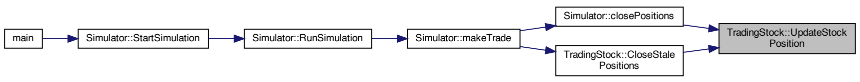
Salil Maharjan

Date

5/14/20.

Definition at line 342 of file TradingStock.cpp.

Here is the caller graph for this function:


main
Simulator::StartSimulation
Simulator::RunSimulation
Simulator::makeTrade
Simulator::closePositions
TradingStock::CloseStale Positions
TradingStock::UpdateStock Position

5.7.4 Member Data Documentation

5.7.4.1 m_availableCapital

```
double TradingStock::m_availableCapital [private]
```

Available capital for trading stock.

Definition at line 129 of file TradingStock.h.

5.7.4.2 m_capitalActivityRecord

```
std::vector<double> TradingStock::m_capitalActivityRecord [private]
```

Records daily capital activities.

Definition at line 151 of file TradingStock.h.

5.7.4.3 m_cumulativePNLRecord

```
std::vector<double> TradingStock::m_cumulativePNLRecord [private]
```

Records daily cumulative PNL.

Definition at line 159 of file TradingStock.h.

5.7.4.4 m_dailyPNLRecord

```
std::vector<double> TradingStock::m_dailyPNLRecord [private]
```

Records daily PNL.

Definition at line 155 of file TradingStock.h.

5.7.4.5 m_dailyReturnRecord

```
std::vector<double> TradingStock::m_dailyReturnRecord [private]
```

Records daily rate of return (ROR)

Definition at line 157 of file TradingStock.h.

5.7.4.6 m_longPosFlag

```
bool TradingStock::m_longPosFlag [private]
```

Long position flag.

Definition at line 137 of file TradingStock.h.

5.7.4.7 m_shareActivityRecord

```
std::vector<double> TradingStock::m_shareActivityRecord [private]
```

Records daily position changes.

Definition at line 153 of file TradingStock.h.

5.7.4.8 m_sharpeRatioRecord

```
std::vector<double> TradingStock::m_sharpeRatioRecord  [private]
```

Records daily sharpe ratio.

Definition at line 163 of file TradingStock.h.

5.7.4.9 m_shortPosFlag

```
bool TradingStock::m_shortPosFlag  [private]
```

Short position flag.

Definition at line 139 of file TradingStock.h.

5.7.4.10 m_signalActivityRecord

```
std::vector<double> TradingStock::m_signalActivityRecord  [private]
```

Records daily signals.

Definition at line 149 of file TradingStock.h.

5.7.4.11 m_tickerName

```
std::string TradingStock::m_tickerName  [private]
```

Ticker name.

Definition at line 127 of file TradingStock.h.

5.7.4.12 m_totalSharesCount

```
double TradingStock::m_totalSharesCount  [private]
```

Total shares held for stock.

Definition at line 135 of file TradingStock.h.

5.7.4.13 m_transactionCounter

```
int TradingStock::m_transactionCounter  [private]
```

Current complete transaction counter.

Definition at line 133 of file TradingStock.h.

5.7.4.14 m_transactions

```
std::vector< Transaction*> TradingStock::m_transactions  [private]
```

Records all transactions.

Definition at line 147 of file TradingStock.h.

5.7.4.15 m_TransactionsCountRecord

```
std::vector< int> TradingStock::m_TransactionsCountRecord  [private]
```

Records daily transaction counts.

Definition at line 161 of file TradingStock.h.

5.7.4.16 m_transactionsTotalCount

```
int TradingStock::m_transactionsTotalCount  [private]
```

Total transactions count.

Definition at line 131 of file TradingStock.h.

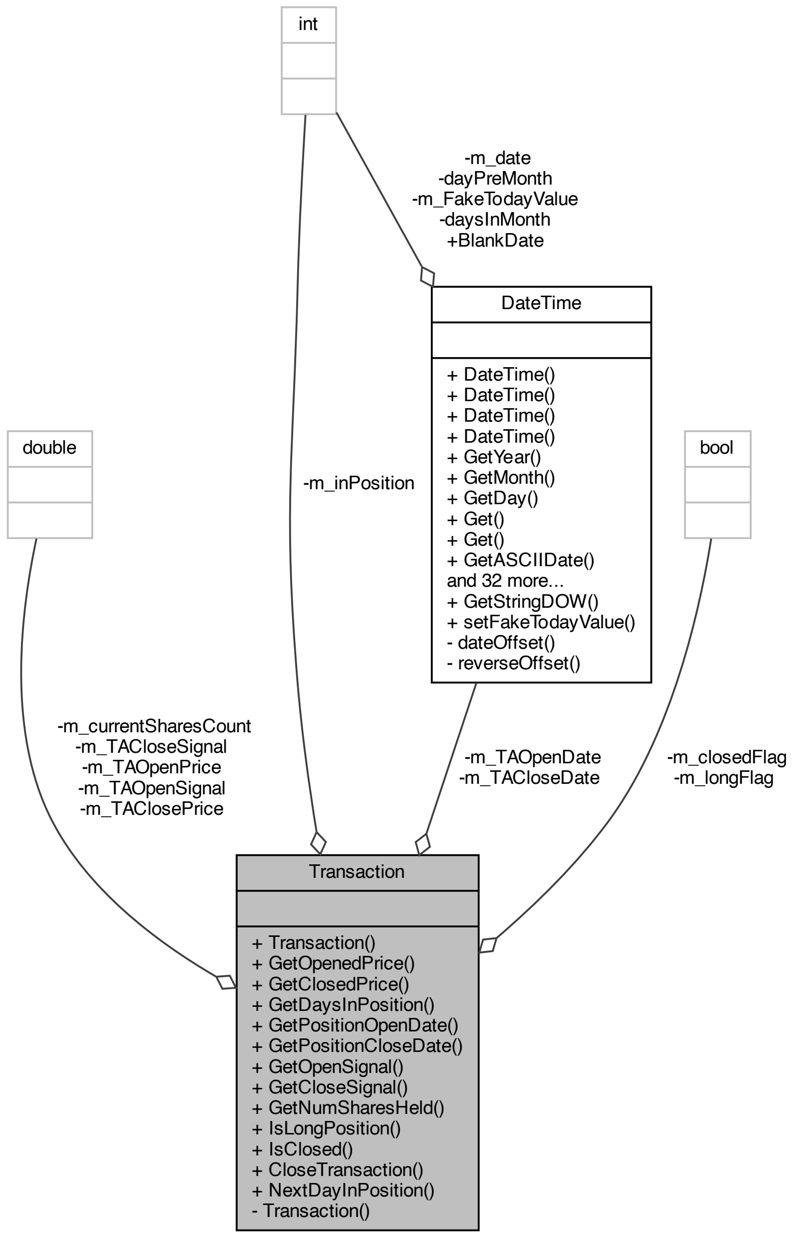
The documentation for this class was generated from the following files:

- **TradingStock.h**
- **TradingStock.cpp**

5.8 Transaction Class Reference

`#include <Transaction.h>`

Collaboration diagram for Transaction:



Public Member Functions

- **Transaction** (**DateTime** ∗a_date, double a_signal, double a_share, double a_price)

  *Main parameterized **Transaction** (*p. **??***) class constructor.*
- const double **GetOpenedPrice** ()

- const double **GetClosedPrice** ()
- const **int** **GetDaysInPosition** ()
- **DateTime** ∗ **GetPositionOpenDate** ()
- **DateTime** ∗ **GetPositionCloseDate** ()
- const double **GetOpenSignal** ()
- const double **GetCloseSignal** ()
- const double **GetNumSharesHeld** ()
- const bool **IsLongPosition** ()
- const bool **IsClosed** ()
- double **CloseTransaction** (**DateTime** ∗a_date, double a_signal, double a_price)

  Method to close constructed transaction objects.
- void **NextDayInPosition** ()

  Transaction *(*p. **??***) method to update the number of days in position of a stock on each trading day.*

Private Member Functions

- **Transaction** ()

  Empty default constructor.

Private Attributes

- **DateTime** ∗ **m_TAOpenDate**

  Transaction *(*p. **??***) open date.*
- **DateTime** ∗ **m_TACloseDate**

  Transaction *(*p. **??***) close date.*
- double **m_TAOpenSignal**

  Transaction *(*p. **??***) open signal.*
- double **m_TACloseSignal**

  Transaction *(*p. **??***) close signal.*
- double **m_currentSharesCount**

  Shares in transaction.
- double **m_TAOpenPrice**

  Transaction *(*p. **??***) open price.*
- double **m_TAClosePrice**

  Transaction *(*p. **??***) close price.*
- **int** **m_inPosition**
- bool **m_longFlag**

  Flag if the transaction is for a long/short position.
- bool **m_closedFlag**

  Flag if transaction is closed.

5.8.1 Detailed Description

Transaction.h (p. **??**) Class that handles the record details of transactions for trading stocks.

Created by Salil Maharjan on 05/22/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 16 of file Transaction.h.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Transaction() [1/2]

```
Transaction::Transaction (
              DateTime * a_date,
             double a_signal,
             double a_share,
             double a_price )
```

Main parameterized **Transaction** (p. **??**) class constructor.

Transaction.cpp (p. **??**) Implementation of **Transaction.h** (p. **??**).

Created by Salil Maharjan on 05/22/20. Copyright © 2020 Salil Maharjan. All rights reserved. **Transaction::**↩ **Transaction** (p. **??**) Main parameterized **Transaction** (p. **??**) class constructor. Opens a transaction.

Parameters

a_date	DateTime∗ The transaction open date.
a_signal	double The signal while opening transaction.
a_share	double The number of shares to open position with.
a_price	double The price at which the transaction is opened.

Author

Salil Maharjan

Date

5/22/20.

Definition at line 26 of file Transaction.cpp.

5.8.2.2 Transaction() [2/2]

```
Transaction::Transaction ( )  [inline], [private]
```

Empty default constructor.

Definition at line 55 of file Transaction.h.

5.8.3 Member Function Documentation

5.8.3.1 CloseTransaction()

```
double Transaction::CloseTransaction (
              DateTime * a_date,
             double a_signal,
             double a_price )
```

Method to close constructed transaction objects.

Transaction::CloseTransaction (p. **??**) Function to close transaction and return the PNL from the completed transaction.

Parameters

a_date	DateTime∗ The transaction close date.
a_signal	double The signal while closing transaction.
a_price	double The price at which the transaction is closed.

Returns

double The profit/loss from completing the transaction.

Author

Salil Maharjan

Date

5/22/20.

Definition at line 70 of file Transaction.cpp.

5.8.3.2 GetClosedPrice()

```
const double Transaction::GetClosedPrice ( )  [inline]
```

Definition at line 32 of file Transaction.h.

5.8.3.3 GetCloseSignal()

```
const double Transaction::GetCloseSignal ( )  [inline]
```

Definition at line 37 of file Transaction.h.

5.8.3.4 GetDaysInPosition()

```
const int Transaction::GetDaysInPosition ( ) [inline]
```

Definition at line 33 of file Transaction.h.

5.8.3.5 GetNumSharesHeld()

```
const double Transaction::GetNumSharesHeld ( ) [inline]
```

Definition at line 38 of file Transaction.h.

5.8.3.6 GetOpenedPrice()

```
const double Transaction::GetOpenedPrice ( ) [inline]
```

Definition at line 31 of file Transaction.h.

5.8.3.7 GetOpenSignal()

```
const double Transaction::GetOpenSignal ( ) [inline]
```

Definition at line 36 of file Transaction.h.

5.8.3.8 GetPositionCloseDate()

```
DateTime* Transaction::GetPositionCloseDate ( ) [inline]
```

Definition at line 35 of file Transaction.h.

5.8.3.9 GetPositionOpenDate()

```
DateTime* Transaction::GetPositionOpenDate ( ) [inline]
```

Definition at line 34 of file Transaction.h.

5.8.3.10 IsClosed()

```
const bool Transaction::IsClosed ( ) [inline]
```

Definition at line 40 of file Transaction.h.

5.8.3.11 IsLongPosition()

```
const bool Transaction::IsLongPosition ( ) [inline]
```

Definition at line 39 of file Transaction.h.

5.8.3.12 NextDayInPosition()

```
void Transaction::NextDayInPosition ( )
```

Transaction (p. **??**) method to update the number of days in position of a stock on each trading day.

Transaction::NextDayInPosition (p. **??**) Method to update the number of days in position in each trading day.

Author

Salil Maharjan

Date

5/22/20.

Definition at line 53 of file Transaction.cpp.

5.8.4 Member Data Documentation

5.8.4.1 m_closedFlag

```
bool Transaction::m_closedFlag [private]
```

Flag if transaction is closed.

Definition at line 80 of file Transaction.h.

5.8.4.2 m_currentSharesCount

```
double Transaction::m_currentSharesCount [private]
```

Shares in transaction.

Definition at line 70 of file Transaction.h.

5.8.4.3 m_inPosition

```
int Transaction::m_inPosition [private]
```

5.8.5 of days in position

Definition at line 76 of file Transaction.h.

5.8.5.1 m_longFlag

```
bool Transaction::m_longFlag [private]
```

Flag if the transaction is for a long/short position.

Definition at line 78 of file Transaction.h.

5.8.5.2 m_TACloseDate

```
DateTime* Transaction::m_TACloseDate [private]
```

Transaction (p. **??**) close date.

Definition at line 64 of file Transaction.h.

5.8.5.3 m_TAClosePrice

```
double Transaction::m_TAClosePrice [private]
```

Transaction (p. **??**) close price.

Definition at line 74 of file Transaction.h.

5.8.5.4 m_TACloseSignal

`double Transaction::m_TACloseSignal [private]`

Transaction (p. **??**) close signal.

Definition at line 68 of file Transaction.h.

5.8.5.5 m_TAOpenDate

`DateTime* Transaction::m_TAOpenDate [private]`

Transaction (p. **??**) open date.

Definition at line 62 of file Transaction.h.

5.8.5.6 m_TAOpenPrice

`double Transaction::m_TAOpenPrice [private]`

Transaction (p. **??**) open price.

Definition at line 72 of file Transaction.h.

5.8.5.7 m_TAOpenSignal

`double Transaction::m_TAOpenSignal [private]`

Transaction (p. **??**) open signal.

Definition at line 66 of file Transaction.h.

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

- **Transaction.h**
- **Transaction.cpp**

Chapter 6

File Documentation

6.1 cmake-build-debug/CMakeCache.txt File Reference

Functions

- Compatible Apple **LLVM** (**clang**-1103.0.32.59)"
- **__attribute__** ((__blocks__(byref)))
- **__attribute__** ((objc_gc(weak)))

Variables

- CMAKE_ADDR2LINE **__pad0__**
- **__clang__**
- **__clang_major__**
- **__clang_minor__**
- **__clang_patchlevel__**
- **__clang_version__**
- **clang**
- **__GNUC_MINOR__**
- **__GNUC_PATCHLEVEL__**
- **__GNUC__**
- **__GXX_ABI_VERSION**
- **__ATOMIC_RELAXED**
- **__ATOMIC_CONSUME**
- **__ATOMIC_ACQUIRE**
- **__ATOMIC_RELEASE**
- **__ATOMIC_ACQ_REL**
- **__ATOMIC_SEQ_CST**
- **__OPENCL_MEMORY_SCOPE_WORK_ITEM**
- **__OPENCL_MEMORY_SCOPE_WORK_GROUP**
- **__OPENCL_MEMORY_SCOPE_DEVICE**
- **__OPENCL_MEMORY_SCOPE_ALL_SVM_DEVICES**
- **__OPENCL_MEMORY_SCOPE_SUB_GROUP**
- **__PRAGMA_REDEFINE_EXTNAME**
- **__VERSION__**
- **__OBJC_BOOL_IS_BOOL**
- **__CONSTANT_CFSTRINGS__**

- **__block**
- **__BLOCKS__**
- **__ORDER_LITTLE_ENDIAN__**
- **__ORDER_BIG_ENDIAN__**
- **__ORDER_PDP_ENDIAN__**
- **__BYTE_ORDER__**
- **__LITTLE_ENDIAN__**
- **_LP64**
- **__LP64__**
- **__CHAR_BIT__**
- **__SCHAR_MAX__**
- **__SHRT_MAX__**
- **__INT_MAX__**
- **__LONG_MAX__**
- **__LONG_LONG_MAX__**
- **__WCHAR_MAX__**
- **__WINT_MAX__**
- **__INTMAX_MAX__**
- **__SIZE_MAX__**
- **__UINTMAX_MAX__**
- **__PTRDIFF_MAX__**
- **__INTPTR_MAX__**
- **__UINTPTR_MAX__**
- **__SIZEOF_DOUBLE__**
- **__SIZEOF_FLOAT__**
- **__SIZEOF_INT__**
- **__SIZEOF_LONG__**
- **__SIZEOF_LONG_DOUBLE__**
- **__SIZEOF_LONG_LONG__**
- **__SIZEOF_POINTER__**
- **__SIZEOF_SHORT__**
- **__SIZEOF_PTRDIFF_T__**
- **__SIZEOF_SIZE_T__**
- **__SIZEOF_WCHAR_T__**
- **__SIZEOF_WINT_T__**
- **__SIZEOF_INT128__**
- **__INTMAX_TYPE__**
- long **int**
- **__INTMAX_FMTd__**
- **ld**
- **__INTMAX_FMTi__**
- **li**
- **__INTMAX_C_SUFFIX__**
- **L**
- **__UINTMAX_TYPE__**
- **__UINTMAX_FMTo__**
- **lo**
- **__UINTMAX_FMTu__**
- **lu**
- **__UINTMAX_FMTx__**
- **lx**
- **__UINTMAX_FMTX__**
- **lX**
- **__UINTMAX_C_SUFFIX__**
- **UL**

- **__INTMAX_WIDTH__**
- **__PTRDIFF_TYPE__**
- **__PTRDIFF_FMTd__**
- **__PTRDIFF_FMTi__**
- **__PTRDIFF_WIDTH__**
- **__INTPTR_TYPE__**
- **__INTPTR_FMTd__**
- **__INTPTR_FMTi__**
- **__INTPTR_WIDTH__**
- **__SIZE_TYPE__**
- **__SIZE_FMTo__**
- **__SIZE_FMTu__**
- **__SIZE_FMTx__**
- **__SIZE_FMTX__**
- **__SIZE_WIDTH__**
- **__WCHAR_TYPE__**
- **__WCHAR_WIDTH__**
- **__WINT_TYPE__**
- **__WINT_WIDTH__**
- **__SIG_ATOMIC_WIDTH__**
- **__SIG_ATOMIC_MAX__**
- **__CHAR16_TYPE__**
- unsigned **short**
- **__CHAR32_TYPE__**
- **__UINTMAX_WIDTH__**
- **__UINTPTR_TYPE__**
- **__UINTPTR_FMTo__**
- **__UINTPTR_FMTu__**
- **__UINTPTR_FMTx__**
- **__UINTPTR_FMTX__**
- **__UINTPTR_WIDTH__**
- **__FLT16_DENORM_MIN__**
- **__FLT16_HAS_DENORM__**
- **__FLT16_DIG__**
- **__FLT16_DECIMAL_DIG__**
- **__FLT16_EPSILON__**
- **__FLT16_HAS_INFINITY__**
- **__FLT16_HAS_QUIET_NAN__**
- **__FLT16_MANT_DIG__**
- **__FLT16_MAX_10_EXP__**
- **__FLT16_MAX_EXP__**
- **__FLT16_MAX__**
- **__FLT16_MIN_10_EXP__**
- **__FLT16_MIN_EXP__**
- **__FLT16_MIN__**
- **__FLT_DENORM_MIN__**
- **__FLT_HAS_DENORM__**
- **__FLT_DIG__**
- **__FLT_DECIMAL_DIG__**
- **__FLT_EPSILON__**
- **__FLT_HAS_INFINITY__**
- **__FLT_HAS_QUIET_NAN__**
- **__FLT_MANT_DIG__**
- **__FLT_MAX_10_EXP__**
- **__FLT_MAX_EXP__**

- **__FLT_MAX__**
- **__FLT_MIN_10_EXP__**
- **__FLT_MIN_EXP__**
- **__FLT_MIN__**
- **__DBL_DENORM_MIN__**
- **__DBL_HAS_DENORM__**
- **__DBL_DIG__**
- **__DBL_DECIMAL_DIG__**
- **__DBL_EPSILON__**
- **__DBL_HAS_INFINITY__**
- **__DBL_HAS_QUIET_NAN__**
- **__DBL_MANT_DIG__**
- **__DBL_MAX_10_EXP__**
- **__DBL_MAX_EXP__**
- **__DBL_MAX__**
- **__DBL_MIN_10_EXP__**
- **__DBL_MIN_EXP__**
- **__DBL_MIN__**
- **__LDBL_DENORM_MIN__**
- **__LDBL_HAS_DENORM__**
- **__LDBL_DIG__**
- **__LDBL_DECIMAL_DIG__**
- **__LDBL_EPSILON__**
- **__LDBL_HAS_INFINITY__**
- **__LDBL_HAS_QUIET_NAN__**
- **__LDBL_MANT_DIG__**
- **__LDBL_MAX_10_EXP__**
- **__LDBL_MAX_EXP__**
- **__LDBL_MAX__**
- **__LDBL_MIN_10_EXP__**
- **__LDBL_MIN_EXP__**
- **__LDBL_MIN__**
- **__POINTER_WIDTH__**
- **__BIGGEST_ALIGNMENT__**
- **__INT8_TYPE__**
- signed **char**
- **__INT8_FMTd__**
- **hhd**
- **__INT8_FMTi__**
- **hhi**
- **__INT8_C_SUFFIX__**
- **__INT16_TYPE__**
- **__INT16_FMTd__**
- **hd**
- **__INT16_FMTi__**
- **hi**
- **__INT16_C_SUFFIX__**
- **__INT32_TYPE__**
- **__INT32_FMTd__**
- **d**
- **__INT32_FMTi__**
- **i**
- **__INT32_C_SUFFIX__**
- **__INT64_TYPE__**
- **__INT64_FMTd__**

- **lld**
- **__INT64_FMTi__**
- **lli**
- **__INT64_C_SUFFIX__**
- **LL**
- **__UINT8_TYPE__**
- **__UINT8_FMTo__**
- **hho**
- **__UINT8_FMTu__**
- **hhu**
- **__UINT8_FMTx__**
- **hhx**
- **__UINT8_FMTX__**
- **hhX**
- **__UINT8_C_SUFFIX__**
- **__UINT8_MAX__**
- **__INT8_MAX__**
- **__UINT16_TYPE__**
- **__UINT16_FMTo__**
- **ho**
- **__UINT16_FMTu__**
- **hu**
- **__UINT16_FMTx__**
- **hx**
- **__UINT16_FMTX__**
- **hX**
- **__UINT16_C_SUFFIX__**
- **__UINT16_MAX__**
- **__INT16_MAX__**
- **__UINT32_TYPE__**
- **__UINT32_FMTo__**
- **o**
- **__UINT32_FMTu__**
- **u**
- **__UINT32_FMTx__**
- **x**
- **__UINT32_FMTX__**
- **X**
- **__UINT32_C_SUFFIX__**
- **U**
- **__UINT32_MAX__**
- **__INT32_MAX__**
- **__UINT64_TYPE__**
- **__UINT64_FMTo__**
- **llo**
- **__UINT64_FMTu__**
- **llu**
- **__UINT64_FMTx__**
- **llx**
- **__UINT64_FMTX__**
- **llX**
- **__UINT64_C_SUFFIX__**
- **ULL**
- **__UINT64_MAX__**
- **__INT64_MAX__**

- **__INT_LEAST8_TYPE__**
- **__INT_LEAST8_MAX__**
- **__INT_LEAST8_FMTd__**
- **__INT_LEAST8_FMTi__**
- **__UINT_LEAST8_TYPE__**
- **__UINT_LEAST8_MAX__**
- **__UINT_LEAST8_FMTo__**
- **__UINT_LEAST8_FMTu__**
- **__UINT_LEAST8_FMTx__**
- **__UINT_LEAST8_FMTX__**
- **__INT_LEAST16_TYPE__**
- **__INT_LEAST16_MAX__**
- **__INT_LEAST16_FMTd__**
- **__INT_LEAST16_FMTi__**
- **__UINT_LEAST16_TYPE__**
- **__UINT_LEAST16_MAX__**
- **__UINT_LEAST16_FMTo__**
- **__UINT_LEAST16_FMTu__**
- **__UINT_LEAST16_FMTx__**
- **__UINT_LEAST16_FMTX__**
- **__INT_LEAST32_TYPE__**
- **__INT_LEAST32_MAX__**
- **__INT_LEAST32_FMTd__**
- **__INT_LEAST32_FMTi__**
- **__UINT_LEAST32_TYPE__**
- **__UINT_LEAST32_MAX__**
- **__UINT_LEAST32_FMTo__**
- **__UINT_LEAST32_FMTu__**
- **__UINT_LEAST32_FMTx__**
- **__UINT_LEAST32_FMTX__**
- **__INT_LEAST64_TYPE__**
- **__INT_LEAST64_MAX__**
- **__INT_LEAST64_FMTd__**
- **__INT_LEAST64_FMTi__**
- **__UINT_LEAST64_TYPE__**
- **__UINT_LEAST64_MAX__**
- **__UINT_LEAST64_FMTo__**
- **__UINT_LEAST64_FMTu__**
- **__UINT_LEAST64_FMTx__**
- **__UINT_LEAST64_FMTX__**
- **__INT_FAST8_TYPE__**
- **__INT_FAST8_MAX__**
- **__INT_FAST8_FMTd__**
- **__INT_FAST8_FMTi__**
- **__UINT_FAST8_TYPE__**
- **__UINT_FAST8_MAX__**
- **__UINT_FAST8_FMTo__**
- **__UINT_FAST8_FMTu__**
- **__UINT_FAST8_FMTx__**
- **__UINT_FAST8_FMTX__**
- **__INT_FAST16_TYPE__**
- **__INT_FAST16_MAX__**
- **__INT_FAST16_FMTd__**
- **__INT_FAST16_FMTi__**
- **__UINT_FAST16_TYPE__**

- **__UINT_FAST16_MAX__**
- **__UINT_FAST16_FMTo__**
- **__UINT_FAST16_FMTu__**
- **__UINT_FAST16_FMTx__**
- **__UINT_FAST16_FMTX__**
- **__INT_FAST32_TYPE__**
- **__INT_FAST32_MAX__**
- **__INT_FAST32_FMTd__**
- **__INT_FAST32_FMTi__**
- **__UINT_FAST32_TYPE__**
- **__UINT_FAST32_MAX__**
- **__UINT_FAST32_FMTo__**
- **__UINT_FAST32_FMTu__**
- **__UINT_FAST32_FMTx__**
- **__UINT_FAST32_FMTX__**
- **__INT_FAST64_TYPE__**
- **__INT_FAST64_MAX__**
- **__INT_FAST64_FMTd__**
- **__INT_FAST64_FMTi__**
- **__UINT_FAST64_TYPE__**
- **__UINT_FAST64_MAX__**
- **__UINT_FAST64_FMTo__**
- **__UINT_FAST64_FMTu__**
- **__UINT_FAST64_FMTx__**
- **__UINT_FAST64_FMTX__**
- **__USER_LABEL_PREFIX__**
- **_**
- **__FINITE_MATH_ONLY__**
- **__GNUC_STDC_INLINE__**
- **__GCC_ATOMIC_TEST_AND_SET_TRUEVAL**
- **__CLANG_ATOMIC_BOOL_LOCK_FREE**
- **__CLANG_ATOMIC_CHAR_LOCK_FREE**
- **__CLANG_ATOMIC_CHAR16_T_LOCK_FREE**
- **__CLANG_ATOMIC_CHAR32_T_LOCK_FREE**
- **__CLANG_ATOMIC_WCHAR_T_LOCK_FREE**
- **__CLANG_ATOMIC_SHORT_LOCK_FREE**
- **__CLANG_ATOMIC_INT_LOCK_FREE**
- **__CLANG_ATOMIC_LONG_LOCK_FREE**
- **__CLANG_ATOMIC_LLONG_LOCK_FREE**
- **__CLANG_ATOMIC_POINTER_LOCK_FREE**
- **__GCC_ATOMIC_BOOL_LOCK_FREE**
- **__GCC_ATOMIC_CHAR_LOCK_FREE**
- **__GCC_ATOMIC_CHAR16_T_LOCK_FREE**
- **__GCC_ATOMIC_CHAR32_T_LOCK_FREE**
- **__GCC_ATOMIC_WCHAR_T_LOCK_FREE**
- **__GCC_ATOMIC_SHORT_LOCK_FREE**
- **__GCC_ATOMIC_INT_LOCK_FREE**
- **__GCC_ATOMIC_LONG_LOCK_FREE**
- **__GCC_ATOMIC_LLONG_LOCK_FREE**
- **__GCC_ATOMIC_POINTER_LOCK_FREE**
- **__NO_INLINE__**
- **__PIC__**
- **__pic__**
- **__FLT_EVAL_METHOD__**
- **__FLT_RADIX__**

- **__DECIMAL_DIG__**
- **__SSP__**
- **__nonnull**
- **_Nonnull**
- **__null_unspecified**
- **_Null_unspecified**
- **__nullable**
- **_Nullable**
- **__GCC_ASM_FLAG_OUTPUTS__**
- **__code_model_small_**
- **__amd64__**
- **__amd64**
- **__x86_64**
- **__x86_64__**
- **__core2**
- **__core2__**
- **__tune_core2__**
- **__REGISTER_PREFIX__**
- **__NO_MATH_INLINES**
- **__FXSR__**
- **__SSE4_1__**
- **__SSSE3__**
- **__SSE3__**
- **__SSE2__**
- **__SSE2_MATH__**
- **__SSE__**
- **__SSE_MATH__**
- **__MMX__**
- **__GCC_HAVE_SYNC_COMPARE_AND_SWAP_1**
- **__GCC_HAVE_SYNC_COMPARE_AND_SWAP_2**
- **__GCC_HAVE_SYNC_COMPARE_AND_SWAP_4**
- **__GCC_HAVE_SYNC_COMPARE_AND_SWAP_8**
- **__GCC_HAVE_SYNC_COMPARE_AND_SWAP_16**
- **__APPLE_CC__**
- **__APPLE__**
- **__STDC_NO_THREADS__**
- **OBJC_NEW_PROPERTIES**
- **__apple_build_version__**
- **__weak**
- **__strong**
- **__unsafe_unretained**
- **__DYNAMIC__**
- **__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__**
- **__MACH__**
- **__STDC__**
- **__STDC_HOSTED__**
- **__STDC_VERSION__**
- **__STDC_UTF_16__**
- **__STDC_UTF_32__**
- **__llvm__**
- **__cpp_rtti**
- **__cpp_exceptions**
- **__cpp_threadsafe_static_init**
- **__cpp_impl_destroying_delete**
- **__EXCEPTIONS**

- **__GXX_RTTI**
- **__DEPRECATED**
- **__GNUG__**
- **__GXX_WEAK__**
- **__private_extern__**
- **extern**
- **__GNUC_GNU_INLINE__**
- **__GLIBCXX_TYPE_INT_N_0**
- **__int128**
- **__GLIBCXX_BITSIZE_INT_N_0**
- **__cplusplus**
- **__STDCPP_DEFAULT_NEW_ALIGNMENT__**
- CMAKE_EXTRA_GENERATOR_CXX_SYSTEM_INCLUDE_DIRS **__pad1__**
- Library Developer CommandLineTools usr lib **clang** **include**
- System Library **Frameworks**
- Library **Frameworks** **CMAKE_EXTRA_GENERATOR_C_SYSTEM_DEFINED_MACROS**
- CMAKE_EXTRA_GENERATOR_C_SYSTEM_INCLUDE_DIRS **__pad2__**

6.1.1 Function Documentation

6.1.1.1 __attribute__() [1/2]

```
__attribute__ (
             (__blocks__(byref))  )
```

6.1.1.2 __attribute__() [2/2]

```
__attribute__ (
             (objc_gc(weak))  )
```

6.1.1.3 LLVM()

```
Compatible Apple LLVM (
              clang-1103.0.32.  59 )
```

6.1.2 Variable Documentation

6.1.2.1 _

```
_
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.2 __amd64

```
__amd64
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.3 __amd64__

```
__amd64__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.4 __APPLE__

```
__APPLE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.5 __apple_build_version__

```
__apple_build_version__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.6 __APPLE_CC__

```
__APPLE_CC__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.7 __ATOMIC_ACQ_REL

```
__ATOMIC_ACQ_REL
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.8 __ATOMIC_ACQUIRE

```
__ATOMIC_ACQUIRE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.9 __ATOMIC_CONSUME

```
__ATOMIC_CONSUME
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.10 __ATOMIC_RELAXED

```
__ATOMIC_RELAXED
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.11 __ATOMIC_RELEASE

```
__ATOMIC_RELEASE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.12 __ATOMIC_SEQ_CST

```
__ATOMIC_SEQ_CST
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.13 __BIGGEST_ALIGNMENT__

```
__BIGGEST_ALIGNMENT__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.14 __block

```
__block
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.15 __BLOCKS__

```
__BLOCKS__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.16 __BYTE_ORDER__

```
__BYTE_ORDER__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.17 __CHAR16_TYPE__

```
__CHAR16_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.18 __CHAR32_TYPE__

```
__CHAR32_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.19 __CHAR_BIT__

```
__CHAR_BIT__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.20 __clang__

```
__clang__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.21 __CLANG_ATOMIC_BOOL_LOCK_FREE

```
__CLANG_ATOMIC_BOOL_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.22 __CLANG_ATOMIC_CHAR16_T_LOCK_FREE

```
__CLANG_ATOMIC_CHAR16_T_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.23 __CLANG_ATOMIC_CHAR32_T_LOCK_FREE

```
__CLANG_ATOMIC_CHAR32_T_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.24 __CLANG_ATOMIC_CHAR_LOCK_FREE

```
__CLANG_ATOMIC_CHAR_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.25 __CLANG_ATOMIC_INT_LOCK_FREE

```
__CLANG_ATOMIC_INT_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.26 __CLANG_ATOMIC_LLONG_LOCK_FREE

```
__CLANG_ATOMIC_LLONG_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.27 __CLANG_ATOMIC_LONG_LOCK_FREE

```
__CLANG_ATOMIC_LONG_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.28 __CLANG_ATOMIC_POINTER_LOCK_FREE

```
__CLANG_ATOMIC_POINTER_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.29 __CLANG_ATOMIC_SHORT_LOCK_FREE

```
__CLANG_ATOMIC_SHORT_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.30 __CLANG_ATOMIC_WCHAR_T_LOCK_FREE

```
__CLANG_ATOMIC_WCHAR_T_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.31 __clang_major__

```
__clang_major__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.32 __clang_minor__

```
__clang_minor__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.33 __clang_patchlevel__

```
__clang_patchlevel__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.34 __clang_version__

```
__clang_version__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.35 __code_model_small_

```
__code_model_small_
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.36 __CONSTANT_CFSTRINGS__

```
__CONSTANT_CFSTRINGS__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.37 __core2

```
__core2
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.38 __core2__

```
__core2__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.39 __cplusplus

```
__cplusplus
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.40 __cpp_exceptions

```
__cpp_exceptions
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.41 __cpp_impl_destroying_delete

```
__cpp_impl_destroying_delete
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.42 __cpp_rtti

```
__cpp_rtti
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.43 __cpp_threadsafe_static_init

```
__cpp_threadsafe_static_init
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.44 __DBL_DECIMAL_DIG__

```
__DBL_DECIMAL_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.45 __DBL_DENORM_MIN__

```
__DBL_DENORM_MIN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.46 __DBL_DIG__

```
__DBL_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.47 __DBL_EPSILON__

```
__DBL_EPSILON__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.48 __DBL_HAS_DENORM__

```
__DBL_HAS_DENORM__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.49 __DBL_HAS_INFINITY__

```
__DBL_HAS_INFINITY__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.50 __DBL_HAS_QUIET_NAN__

```
__DBL_HAS_QUIET_NAN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.51 __DBL_MANT_DIG__

```
__DBL_MANT_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.52 __DBL_MAX_10_EXP__

```
__DBL_MAX_10_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.53 __DBL_MAX__

```
__DBL_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.54 __DBL_MAX_EXP__

```
__DBL_MAX_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.55 __DBL_MIN_10_EXP__

`__DBL_MIN_10_EXP__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.56 __DBL_MIN__

`__DBL_MIN__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.57 __DBL_MIN_EXP__

`__DBL_MIN_EXP__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.58 __DECIMAL_DIG__

`__DECIMAL_DIG__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.59 __DEPRECATED

`__DEPRECATED`

Definition at line 299 of file CMakeCache.txt.

6.1.2.60 __DYNAMIC__

`__DYNAMIC__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.61 __ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__

```
__ENVIRONMENT_MAC_OS_X_VERSION_MIN_REQUIRED__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.62 __EXCEPTIONS

```
__EXCEPTIONS
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.63 __FINITE_MATH_ONLY__

```
__FINITE_MATH_ONLY__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.64 __FLT16_DECIMAL_DIG__

```
__FLT16_DECIMAL_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.65 __FLT16_DENORM_MIN__

```
__FLT16_DENORM_MIN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.66 __FLT16_DIG__

```
__FLT16_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.67 __FLT16_EPSILON__

```
__FLT16_EPSILON__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.68 __FLT16_HAS_DENORM__

```
__FLT16_HAS_DENORM__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.69 __FLT16_HAS_INFINITY__

```
__FLT16_HAS_INFINITY__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.70 __FLT16_HAS_QUIET_NAN__

```
__FLT16_HAS_QUIET_NAN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.71 __FLT16_MANT_DIG__

```
__FLT16_MANT_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.72 __FLT16_MAX_10_EXP__

```
__FLT16_MAX_10_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.73 __FLT16_MAX__

```
__FLT16_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.74 __FLT16_MAX_EXP__

```
__FLT16_MAX_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.75 __FLT16_MIN_10_EXP__

```
__FLT16_MIN_10_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.76 __FLT16_MIN__

```
__FLT16_MIN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.77 __FLT16_MIN_EXP__

```
__FLT16_MIN_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.78 __FLT_DECIMAL_DIG__

```
__FLT_DECIMAL_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.79 __FLT_DENORM_MIN__

```
__FLT_DENORM_MIN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.80 __FLT_DIG__

```
__FLT_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.81 __FLT_EPSILON__

```
__FLT_EPSILON__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.82 __FLT_EVAL_METHOD__

```
__FLT_EVAL_METHOD__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.83 __FLT_HAS_DENORM__

```
__FLT_HAS_DENORM__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.84 __FLT_HAS_INFINITY__

```
__FLT_HAS_INFINITY__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.85 __FLT_HAS_QUIET_NAN__

```
__FLT_HAS_QUIET_NAN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.86 __FLT_MANT_DIG__

```
__FLT_MANT_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.87 __FLT_MAX_10_EXP__

```
__FLT_MAX_10_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.88 __FLT_MAX__

```
__FLT_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.89 __FLT_MAX_EXP__

```
__FLT_MAX_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.90 __FLT_MIN_10_EXP__

```
__FLT_MIN_10_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.91 __FLT_MIN__

```
__FLT_MIN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.92 __FLT_MIN_EXP__

```
__FLT_MIN_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.93 __FLT_RADIX__

```
__FLT_RADIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.94 __FXSR__

```
__FXSR__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.95 __GCC_ASM_FLAG_OUTPUTS__

```
__GCC_ASM_FLAG_OUTPUTS__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.96 __GCC_ATOMIC_BOOL_LOCK_FREE

```
__GCC_ATOMIC_BOOL_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.97 __GCC_ATOMIC_CHAR16_T_LOCK_FREE

```
__GCC_ATOMIC_CHAR16_T_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.98 __GCC_ATOMIC_CHAR32_T_LOCK_FREE

```
__GCC_ATOMIC_CHAR32_T_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.99 __GCC_ATOMIC_CHAR_LOCK_FREE

```
__GCC_ATOMIC_CHAR_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.100 __GCC_ATOMIC_INT_LOCK_FREE

```
__GCC_ATOMIC_INT_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.101 __GCC_ATOMIC_LLONG_LOCK_FREE

```
__GCC_ATOMIC_LLONG_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.102 __GCC_ATOMIC_LONG_LOCK_FREE

```
__GCC_ATOMIC_LONG_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.103 __GCC_ATOMIC_POINTER_LOCK_FREE

```
__GCC_ATOMIC_POINTER_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.104 __GCC_ATOMIC_SHORT_LOCK_FREE

```
__GCC_ATOMIC_SHORT_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.105 __GCC_ATOMIC_TEST_AND_SET_TRUEVAL

```
__GCC_ATOMIC_TEST_AND_SET_TRUEVAL
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.106 __GCC_ATOMIC_WCHAR_T_LOCK_FREE

```
__GCC_ATOMIC_WCHAR_T_LOCK_FREE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.107 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_1

```
__GCC_HAVE_SYNC_COMPARE_AND_SWAP_1
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.108 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_16

```
__GCC_HAVE_SYNC_COMPARE_AND_SWAP_16
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.109 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_2

```
__GCC_HAVE_SYNC_COMPARE_AND_SWAP_2
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.110 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_4

```
__GCC_HAVE_SYNC_COMPARE_AND_SWAP_4
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.111 __GCC_HAVE_SYNC_COMPARE_AND_SWAP_8

```
__GCC_HAVE_SYNC_COMPARE_AND_SWAP_8
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.112 __GLIBCXX_BITSIZE_INT_N_0

```
__GLIBCXX_BITSIZE_INT_N_0
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.113 __GLIBCXX_TYPE_INT_N_0

```
__GLIBCXX_TYPE_INT_N_0
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.114 __GNUC__

```
__GNUC__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.115 __GNUC_GNU_INLINE__

`__GNUC_GNU_INLINE__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.116 __GNUC_MINOR__

`__GNUC_MINOR__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.117 __GNUC_PATCHLEVEL__

`__GNUC_PATCHLEVEL__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.118 __GNUC_STDC_INLINE__

`__GNUC_STDC_INLINE__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.119 __GNUG__

`__GNUG__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.120 __GXX_ABI_VERSION

`__GXX_ABI_VERSION`

Definition at line 299 of file CMakeCache.txt.

6.1.2.121 __GXX_RTTI

```
__GXX_RTTI
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.122 __GXX_WEAK__

```
__GXX_WEAK__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.123 __int128

```
__int128
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.124 __INT16_C_SUFFIX__

```
__INT16_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.125 __INT16_FMTd__

```
__INT16_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.126 __INT16_FMTi__

```
__INT16_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.127 __INT16_MAX__

```
__INT16_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.128 __INT16_TYPE__

```
__INT16_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.129 __INT32_C_SUFFIX__

```
__INT32_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.130 __INT32_FMTd__

```
__INT32_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.131 __INT32_FMTi__

```
__INT32_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.132 __INT32_MAX__

```
__INT32_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.133 __INT32_TYPE__

```
__INT32_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.134 __INT64_C_SUFFIX__

```
__INT64_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.135 __INT64_FMTd__

```
__INT64_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.136 __INT64_FMTi__

```
__INT64_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.137 __INT64_MAX__

```
__INT64_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.138 __INT64_TYPE__

```
__INT64_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.139 __INT8_C_SUFFIX__

```
__INT8_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.140 __INT8_FMTd__

```
__INT8_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.141 __INT8_FMTi__

```
__INT8_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.142 __INT8_MAX__

```
__INT8_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.143 __INT8_TYPE__

```
__INT8_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.144 __INT_FAST16_FMTd__

```
__INT_FAST16_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.145 __INT_FAST16_FMTi__

```
__INT_FAST16_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.146 __INT_FAST16_MAX__

```
__INT_FAST16_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.147 __INT_FAST16_TYPE__

```
__INT_FAST16_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.148 __INT_FAST32_FMTd__

```
__INT_FAST32_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.149 __INT_FAST32_FMTi__

```
__INT_FAST32_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.150 __INT_FAST32_MAX__

```
__INT_FAST32_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.151 __INT_FAST32_TYPE__

```
__INT_FAST32_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.152 __INT_FAST64_FMTd__

```
__INT_FAST64_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.153 __INT_FAST64_FMTi__

```
__INT_FAST64_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.154 __INT_FAST64_MAX__

```
__INT_FAST64_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.155 __INT_FAST64_TYPE__

```
__INT_FAST64_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.156 __INT_FAST8_FMTd__

```
__INT_FAST8_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.157 __INT_FAST8_FMTi__

```
__INT_FAST8_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.158 __INT_FAST8_MAX__

```
__INT_FAST8_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.159 __INT_FAST8_TYPE__

```
__INT_FAST8_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.160 __INT_LEAST16_FMTd__

```
__INT_LEAST16_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.161 __INT_LEAST16_FMTi__

```
__INT_LEAST16_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.162 __INT_LEAST16_MAX__

```
__INT_LEAST16_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.163 __INT_LEAST16_TYPE__

```
__INT_LEAST16_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.164 __INT_LEAST32_FMTd__

```
__INT_LEAST32_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.165 __INT_LEAST32_FMTi__

```
__INT_LEAST32_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.166 __INT_LEAST32_MAX__

```
__INT_LEAST32_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.167 __INT_LEAST32_TYPE__

```
__INT_LEAST32_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.168 __INT_LEAST64_FMTd__

```
__INT_LEAST64_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.169 __INT_LEAST64_FMTi__

```
__INT_LEAST64_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.170 __INT_LEAST64_MAX__

```
__INT_LEAST64_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.171 __INT_LEAST64_TYPE__

```
__INT_LEAST64_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.172 __INT_LEAST8_FMTd__

```
__INT_LEAST8_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.173 __INT_LEAST8_FMTi__

```
__INT_LEAST8_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.174 __INT_LEAST8_MAX__

```
__INT_LEAST8_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.175 __INT_LEAST8_TYPE__

```
__INT_LEAST8_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.176 __INT_MAX__

```
__INT_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.177 __INTMAX_C_SUFFIX__

```
__INTMAX_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.178 __INTMAX_FMTd__

```
__INTMAX_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.179 __INTMAX_FMTi__

```
__INTMAX_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.180 __INTMAX_MAX__

```
__INTMAX_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.181 __INTMAX_TYPE__

```
__INTMAX_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.182 __INTMAX_WIDTH__

```
__INTMAX_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.183 __INTPTR_FMTd__

```
__INTPTR_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.184 __INTPTR_FMTi__

```
__INTPTR_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.185 __INTPTR_MAX__

```
__INTPTR_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.186 __INTPTR_TYPE__

```
__INTPTR_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.187 __INTPTR_WIDTH__

```
__INTPTR_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.188 __LDBL_DECIMAL_DIG__

```
__LDBL_DECIMAL_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.189 __LDBL_DENORM_MIN__

```
__LDBL_DENORM_MIN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.190 __LDBL_DIG__

```
__LDBL_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.191 __LDBL_EPSILON__

```
__LDBL_EPSILON__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.192 __LDBL_HAS_DENORM__

```
__LDBL_HAS_DENORM__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.193 __LDBL_HAS_INFINITY__

```
__LDBL_HAS_INFINITY__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.194 __LDBL_HAS_QUIET_NAN__

```
__LDBL_HAS_QUIET_NAN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.195 __LDBL_MANT_DIG__

```
__LDBL_MANT_DIG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.196 __LDBL_MAX_10_EXP__

```
__LDBL_MAX_10_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.197 __LDBL_MAX__

```
__LDBL_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.198 __LDBL_MAX_EXP__

```
__LDBL_MAX_EXP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.199 __LDBL_MIN_10_EXP__

`__LDBL_MIN_10_EXP__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.200 __LDBL_MIN__

`__LDBL_MIN__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.201 __LDBL_MIN_EXP__

`__LDBL_MIN_EXP__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.202 __LITTLE_ENDIAN__

`__LITTLE_ENDIAN__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.203 __llvm__

`__llvm__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.204 __LONG_LONG_MAX__

`__LONG_LONG_MAX__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.205 __LONG_MAX__

```
__LONG_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.206 __LP64__

```
__LP64__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.207 __MACH__

```
__MACH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.208 __MMX__

```
__MMX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.209 __NO_INLINE__

```
__NO_INLINE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.210 __NO_MATH_INLINES

```
__NO_MATH_INLINES
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.211 __nonnull

```
__nonnull
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.212 __null_unspecified

```
__null_unspecified
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.213 __nullable

```
__nullable
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.214 __OBJC_BOOL_IS_BOOL

```
__OBJC_BOOL_IS_BOOL
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.215 __OPENCL_MEMORY_SCOPE_ALL_SVM_DEVICES

```
__OPENCL_MEMORY_SCOPE_ALL_SVM_DEVICES
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.216 __OPENCL_MEMORY_SCOPE_DEVICE

```
__OPENCL_MEMORY_SCOPE_DEVICE
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.217 __OPENCL_MEMORY_SCOPE_SUB_GROUP

```
__OPENCL_MEMORY_SCOPE_SUB_GROUP
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.218 __OPENCL_MEMORY_SCOPE_WORK_GROUP

```
__OPENCL_MEMORY_SCOPE_WORK_GROUP
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.219 __OPENCL_MEMORY_SCOPE_WORK_ITEM

```
__OPENCL_MEMORY_SCOPE_WORK_ITEM
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.220 __ORDER_BIG_ENDIAN__

```
__ORDER_BIG_ENDIAN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.221 __ORDER_LITTLE_ENDIAN__

```
__ORDER_LITTLE_ENDIAN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.222 __ORDER_PDP_ENDIAN__

```
__ORDER_PDP_ENDIAN__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.223 __pad0__

```
CMAKE_ADDR2LINE __pad0__
```

Definition at line 18 of file CMakeCache.txt.

6.1.2.224 __pad1__

```
CMAKE_EXTRA_GENERATOR_CXX_SYSTEM_INCLUDE_DIRS __pad1__
```

Definition at line 301 of file CMakeCache.txt.

6.1.2.225 __pad2__

```
CMAKE_EXTRA_GENERATOR_C_SYSTEM_INCLUDE_DIRS __pad2__
```

Definition at line 305 of file CMakeCache.txt.

6.1.2.226 __PIC__

```
__PIC__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.227 __pic__

```
__pic__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.228 __POINTER_WIDTH__

```
__POINTER_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.229 __PRAGMA_REDEFINE_EXTNAME

```
__PRAGMA_REDEFINE_EXTNAME
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.230 __private_extern__

```
__private_extern__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.231 __PTRDIFF_FMTd__

```
__PTRDIFF_FMTd__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.232 __PTRDIFF_FMTi__

```
__PTRDIFF_FMTi__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.233 __PTRDIFF_MAX__

```
__PTRDIFF_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.234 __PTRDIFF_TYPE__

```
__PTRDIFF_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.235 __PTRDIFF_WIDTH__

```
__PTRDIFF_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.236 __REGISTER_PREFIX__

```
__REGISTER_PREFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.237 __SCHAR_MAX__

```
__SCHAR_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.238 __SHRT_MAX__

```
__SHRT_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.239 __SIG_ATOMIC_MAX__

```
__SIG_ATOMIC_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.240 __SIG_ATOMIC_WIDTH__

```
__SIG_ATOMIC_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.241 __SIZE_FMTo__

`__SIZE_FMTo__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.242 __SIZE_FMTu__

`__SIZE_FMTu__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.243 __SIZE_FMTx__

`__SIZE_FMTx__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.244 __SIZE_FMTX__

`__SIZE_FMTX__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.245 __SIZE_MAX__

`__SIZE_MAX__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.246 __SIZE_TYPE__

`__SIZE_TYPE__`

Definition at line 299 of file CMakeCache.txt.

6.1.2.247 __SIZE_WIDTH__

```
__SIZE_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.248 __SIZEOF_DOUBLE__

```
__SIZEOF_DOUBLE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.249 __SIZEOF_FLOAT__

```
__SIZEOF_FLOAT__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.250 __SIZEOF_INT128__

```
__SIZEOF_INT128__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.251 __SIZEOF_INT__

```
__SIZEOF_INT__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.252 __SIZEOF_LONG__

```
__SIZEOF_LONG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.253 __SIZEOF_LONG_DOUBLE__

```
__SIZEOF_LONG_DOUBLE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.254 __SIZEOF_LONG_LONG__

```
__SIZEOF_LONG_LONG__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.255 __SIZEOF_POINTER__

```
__SIZEOF_POINTER__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.256 __SIZEOF_PTRDIFF_T__

```
__SIZEOF_PTRDIFF_T__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.257 __SIZEOF_SHORT__

```
__SIZEOF_SHORT__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.258 __SIZEOF_SIZE_T__

```
__SIZEOF_SIZE_T__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.259 __SIZEOF_WCHAR_T__

```
__SIZEOF_WCHAR_T__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.260 __SIZEOF_WINT_T__

```
__SIZEOF_WINT_T__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.261 __SSE2__

```
__SSE2__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.262 __SSE2_MATH__

```
__SSE2_MATH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.263 __SSE3__

```
__SSE3__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.264 __SSE4_1__

```
__SSE4_1__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.265 __SSE__

```
__SSE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.266 __SSE_MATH__

```
__SSE_MATH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.267 __SSP__

```
__SSP__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.268 __SSSE3__

```
__SSSE3__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.269 __STDC__

```
__STDC__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.270 __STDC_HOSTED__

```
__STDC_HOSTED__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.271 __STDC_NO_THREADS__

```
__STDC_NO_THREADS__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.272 __STDC_UTF_16__

```
__STDC_UTF_16__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.273 __STDC_UTF_32__

```
__STDC_UTF_32__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.274 __STDC_VERSION__

```
__STDC_VERSION__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.275 __STDCPP_DEFAULT_NEW_ALIGNMENT__

```
__STDCPP_DEFAULT_NEW_ALIGNMENT__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.276 __strong

```
__strong
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.277 __tune_core2__

```
__tune_core2__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.278 __UINT16_C_SUFFIX__

```
__UINT16_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.279 __UINT16_FMTo__

```
__UINT16_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.280 __UINT16_FMTu__

```
__UINT16_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.281 __UINT16_FMTx__

```
__UINT16_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.282 __UINT16_FMTX__

```
__UINT16_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.283 __UINT16_MAX__

```
__UINT16_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.284 __UINT16_TYPE__

```
__UINT16_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.285 __UINT32_C_SUFFIX__

```
__UINT32_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.286 __UINT32_FMTo__

```
__UINT32_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.287 __UINT32_FMTu__

```
__UINT32_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.288 __UINT32_FMTx__

```
__UINT32_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.289 __UINT32_FMTX__

```
__UINT32_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.290 __UINT32_MAX__

```
__UINT32_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.291 __UINT32_TYPE__

```
__UINT32_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.292 __UINT64_C_SUFFIX__

```
__UINT64_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.293 __UINT64_FMTo__

```
__UINT64_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.294 __UINT64_FMTu__

```
__UINT64_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.295 __UINT64_FMTX__

```
__UINT64_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.296 __UINT64_FMTx__

```
__UINT64_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.297 __UINT64_MAX__

```
__UINT64_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.298 __UINT64_TYPE__

```
__UINT64_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.299 __UINT8_C_SUFFIX__

```
__UINT8_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.300 __UINT8_FMTo__

```
__UINT8_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.301 __UINT8_FMTu__

```
__UINT8_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.302 __UINT8_FMTx__

```
__UINT8_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.303 __UINT8_FMTX__

```
__UINT8_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.304 __UINT8_MAX__

```
__UINT8_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.305 __UINT8_TYPE__

```
__UINT8_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.306 __UINT_FAST16_FMTo__

```
__UINT_FAST16_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.307 __UINT_FAST16_FMTu__

```
__UINT_FAST16_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.308 __UINT_FAST16_FMTX__

```
__UINT_FAST16_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.309 __UINT_FAST16_FMTx__

```
__UINT_FAST16_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.310 __UINT_FAST16_MAX__

```
__UINT_FAST16_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.311 __UINT_FAST16_TYPE__

```
__UINT_FAST16_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.312 __UINT_FAST32_FMTo__

```
__UINT_FAST32_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.313 __UINT_FAST32_FMTu__

```
__UINT_FAST32_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.314 __UINT_FAST32_FMTx__

```
__UINT_FAST32_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.315 __UINT_FAST32_FMTX__

```
__UINT_FAST32_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.316 __UINT_FAST32_MAX__

```
__UINT_FAST32_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.317 __UINT_FAST32_TYPE__

```
__UINT_FAST32_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.318 __UINT_FAST64_FMTo__

```
__UINT_FAST64_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.319 __UINT_FAST64_FMTu__

```
__UINT_FAST64_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.320 __UINT_FAST64_FMTx__

```
__UINT_FAST64_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.321 __UINT_FAST64_FMTX__

```
__UINT_FAST64_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.322 __UINT_FAST64_MAX__

```
__UINT_FAST64_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.323 __UINT_FAST64_TYPE__

```
__UINT_FAST64_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.324 __UINT_FAST8_FMTo__

```
__UINT_FAST8_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.325 __UINT_FAST8_FMTu__

```
__UINT_FAST8_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.326 __UINT_FAST8_FMTX__

```
__UINT_FAST8_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.327 __UINT_FAST8_FMTx__

```
__UINT_FAST8_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.328 __UINT_FAST8_MAX__

```
__UINT_FAST8_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.329 __UINT_FAST8_TYPE__

```
__UINT_FAST8_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.330 __UINT_LEAST16_FMTo__

```
__UINT_LEAST16_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.331 __UINT_LEAST16_FMTu__

```
__UINT_LEAST16_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.332 __UINT_LEAST16_FMTx__

```
__UINT_LEAST16_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.333 __UINT_LEAST16_FMTX__

```
__UINT_LEAST16_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.334 __UINT_LEAST16_MAX__

```
__UINT_LEAST16_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.335 __UINT_LEAST16_TYPE__

```
__UINT_LEAST16_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.336 __UINT_LEAST32_FMTo__

```
__UINT_LEAST32_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.337 __UINT_LEAST32_FMTu__

```
__UINT_LEAST32_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.338 __UINT_LEAST32_FMTX__

```
__UINT_LEAST32_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.339 __UINT_LEAST32_FMTx__

```
__UINT_LEAST32_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.340 __UINT_LEAST32_MAX__

```
__UINT_LEAST32_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.341 __UINT_LEAST32_TYPE__

```
__UINT_LEAST32_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.342 __UINT_LEAST64_FMTo__

```
__UINT_LEAST64_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.343 __UINT_LEAST64_FMTu__

```
__UINT_LEAST64_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.344 __UINT_LEAST64_FMTX__

```
__UINT_LEAST64_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.345 __UINT_LEAST64_FMTx__

```
__UINT_LEAST64_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.346 __UINT_LEAST64_MAX__

```
__UINT_LEAST64_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.347 __UINT_LEAST64_TYPE__

```
__UINT_LEAST64_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.348 __UINT_LEAST8_FMTo__

```
__UINT_LEAST8_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.349 __UINT_LEAST8_FMTu__

```
__UINT_LEAST8_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.350 __UINT_LEAST8_FMTX__

```
__UINT_LEAST8_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.351 __UINT_LEAST8_FMTx__

```
__UINT_LEAST8_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.352 __UINT_LEAST8_MAX__

```
__UINT_LEAST8_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.353 __UINT_LEAST8_TYPE__

```
__UINT_LEAST8_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.354 __UINTMAX_C_SUFFIX__

```
__UINTMAX_C_SUFFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.355 __UINTMAX_FMTo__

```
__UINTMAX_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.356 __UINTMAX_FMTu__

```
__UINTMAX_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.357 __UINTMAX_FMTX__

```
__UINTMAX_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.358 __UINTMAX_FMTx__

```
__UINTMAX_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.359 __UINTMAX_MAX__

```
__UINTMAX_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.360 __UINTMAX_TYPE__

```
__UINTMAX_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.361 __UINTMAX_WIDTH__

```
__UINTMAX_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.362 __UINTPTR_FMTo__

```
__UINTPTR_FMTo__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.363 __UINTPTR_FMTu__

```
__UINTPTR_FMTu__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.364 __UINTPTR_FMTx__

```
__UINTPTR_FMTx__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.365 __UINTPTR_FMTX__

```
__UINTPTR_FMTX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.366 __UINTPTR_MAX__

```
__UINTPTR_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.367 __UINTPTR_TYPE__

```
__UINTPTR_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.368 __UINTPTR_WIDTH__

```
__UINTPTR_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.369 __unsafe_unretained

```
__unsafe_unretained
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.370 __USER_LABEL_PREFIX__

```
__USER_LABEL_PREFIX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.371 __VERSION__

```
__VERSION__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.372 __WCHAR_MAX__

```
__WCHAR_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.373 __WCHAR_TYPE__

```
__WCHAR_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.374 __WCHAR_WIDTH__

```
__WCHAR_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.375 __weak

```
__weak
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.376 __WINT_MAX__

```
__WINT_MAX__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.377 __WINT_TYPE__

```
__WINT_TYPE__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.378 __WINT_WIDTH__

```
__WINT_WIDTH__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.379 __x86_64

```
__x86_64
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.380 __x86_64__

```
__x86_64__
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.381 _LP64

```
_LP64
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.382 _Nonnull

```
_Nonnull
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.383 _Null_unspecified

```
_Null_unspecified
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.384 _Nullable

```
_Nullable
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.385 char

```
unsigned char
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.386 clang

```
clang
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.387 CMAKE_EXTRA_GENERATOR_C_SYSTEM_DEFINED_MACROS

```
Library  Frameworks CMAKE_EXTRA_GENERATOR_C_SYSTEM_DEFINED_MACROS
```

Definition at line 303 of file CMakeCache.txt.

6.1.2.388 d

```
d
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.389 extern

```
extern
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.390 Frameworks

```
System Library Frameworks
```

Definition at line 301 of file CMakeCache.txt.

6.1.2.391 hd

```
hd
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.392 hhd

```
hhd
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.393 hhi

```
hhi
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.394 hho

```
hho
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.395 hhu

```
hhu
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.396 hhX

```
hhX
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.397 hhx

```
hhx
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.398 hi

```
hi
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.399 ho

```
ho
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.400 hu

```
hu
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.401 hx

```
hx
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.402 hX

```
hX
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.403 i

```
i
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.404 include

```
Library Developer CommandLineTools usr include
```

Definition at line 301 of file CMakeCache.txt.

6.1.2.405 int

```
long long unsigned int
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.406 L

```
L
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.407 ld

```
ld
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.408 li

```
li
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.409 LL

```
LL
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.410 lld

```
lld
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.411 lli

```
lli
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.412 llo

```
llo
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.413 llu

```
llu
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.414 llx

```
llx
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.415 llX

`llX`

Definition at line 299 of file CMakeCache.txt.

6.1.2.416 lo

`lo`

Definition at line 299 of file CMakeCache.txt.

6.1.2.417 lu

`lu`

Definition at line 299 of file CMakeCache.txt.

6.1.2.418 lx

`lx`

Definition at line 299 of file CMakeCache.txt.

6.1.2.419 lX

`lX`

Definition at line 299 of file CMakeCache.txt.

6.1.2.420 o

`o`

Definition at line 299 of file CMakeCache.txt.

6.1.2.421 OBJC_NEW_PROPERTIES

```
OBJC_NEW_PROPERTIES
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.422 short

```
unsigned short
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.423 U

```
U
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.424 u

```
u
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.425 UL

```
UL
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.426 ULL

```
ULL
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.427 X

```
X
```

Definition at line 299 of file CMakeCache.txt.

6.1.2.428 x

```
x
```

Definition at line 299 of file CMakeCache.txt.

6.2 cmake-build-debug/CMakeFiles/3.16.5/CompilerIdC/CMakeC↩ CompilerId.c File Reference

Macros

- #define **COMPILER_ID** ""
- #define **STRINGIFY_HELPER**(**X**) # **X**
- #define **STRINGIFY**(**X**) **STRINGIFY_HELPER**(**X**)
- #define **PLATFORM_ID**
- #define **ARCHITECTURE_ID**
- #define **DEC**(n)
- #define **HEX**(n)
- #define **C_DIALECT**

Functions

- **int main** (**int** argc, **char** ∗argv[ ])

Variables

- **char** const ∗ **info_compiler** = "INFO" ":" "compiler[" COMPILER_ID "]"
- **char** const ∗ **info_platform** = "INFO" ":" "platform[" PLATFORM_ID "]"
- **char** const ∗ **info_arch** = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const **char** ∗ **info_language_dialect_default**

6.2.1 Macro Definition Documentation

6.2.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 541 of file CMakeCCompilerId.c.

6.2.1.2 C_DIALECT

```
#define C_DIALECT
```

Definition at line 626 of file CMakeCCompilerId.c.

6.2.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 315 of file CMakeCCompilerId.c.

6.2.1.4 DEC

```
#define DEC(
               n )
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 545 of file CMakeCCompilerId.c.

6.2.1.5 HEX

```
#define HEX(
               n )
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 556 of file CMakeCCompilerId.c.

6.2.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 437 of file CMakeCCompilerId.c.

6.2.1.7 STRINGIFY

```
#define STRINGIFY(
              X )  STRINGIFY_HELPER( X)
```

Definition at line 336 of file CMakeCCompilerId.c.

6.2.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) # X
```

Definition at line 335 of file CMakeCCompilerId.c.

6.2.2 Function Documentation

6.2.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 645 of file CMakeCCompilerId.c.

6.2.3 Variable Documentation

6.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 615 of file CMakeCCompilerId.c.

6.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 321 of file CMakeCCompilerId.c.

6.2.3.3 info_language_dialect_default

```
const  char* info_language_dialect_default
```

Initial value:

```
=
  "INFO" ":" "dialect_default[" C_DIALECT "]"
```

Definition at line 634 of file CMakeCCompilerId.c.

6.2.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 614 of file CMakeCCompilerId.c.

6.3 cmake-build-debug/CMakeFiles/3.16.5/CompilerIdCXX/CMakeCXX↩ CompilerId.cpp File Reference

Macros

- #define **COMPILER_ID** ""
- #define **STRINGIFY_HELPER**(**X**) # **X**
- #define **STRINGIFY**(**X**) **STRINGIFY_HELPER**(**X**)
- #define **PLATFORM_ID**
- #define **ARCHITECTURE_ID**
- #define **DEC**(n)
- #define **HEX**(n)
- #define **CXX_STD** **__cplusplus**

Functions

- **int** **main** (**int** argc, **char** ∗argv[ ])

Variables

- **char** const ∗ **info_compiler** = "INFO" ":" "compiler[" COMPILER_ID "]"
- **char** const ∗ **info_platform** = "INFO" ":" "platform[" PLATFORM_ID "]"
- **char** const ∗ **info_arch** = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const **char** ∗ **info_language_dialect_default**

6.3.1 Macro Definition Documentation

6.3.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 526 of file CMakeCXXCompilerId.cpp.

6.3.1.2 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 300 of file CMakeCXXCompilerId.cpp.

6.3.1.3 CXX_STD

```
#define CXX_STD  __cplusplus
```

Definition at line 619 of file CMakeCXXCompilerId.cpp.

6.3.1.4 DEC

```
#define DEC(
              n )
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 530 of file CMakeCXXCompilerId.cpp.

6.3.1.5 HEX

```
#define HEX(
              n )
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 541 of file CMakeCXXCompilerId.cpp.

6.3.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 422 of file CMakeCXXCompilerId.cpp.

6.3.1.7 STRINGIFY

```
#define STRINGIFY(
               X )  STRINGIFY_HELPER( X)
```

Definition at line 321 of file CMakeCXXCompilerId.cpp.

6.3.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
               X ) # X
```

Definition at line 320 of file CMakeCXXCompilerId.cpp.

6.3.2 Function Documentation

6.3.2.1 main()

```
 int main (
              int argc,
              char * argv[] )
```

Definition at line 637 of file CMakeCXXCompilerId.cpp.

6.3.3 Variable Documentation

6.3.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 600 of file CMakeCXXCompilerId.cpp.

6.3.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 306 of file CMakeCXXCompilerId.cpp.

6.3.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
= "INFO" ":" "dialect_default["
  "98"
"]"
```

Definition at line 621 of file CMakeCXXCompilerId.cpp.

6.3.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 599 of file CMakeCXXCompilerId.cpp.

6.4 cmake-build-debug/CMakeFiles/clion-environment.txt File Reference

6.5 cmake-build-debug/CMakeFiles/clion-log.txt File Reference

6.6 cmake-build-debug/CMakeFiles/dorothy.dir/link.txt File Reference

Variables

- Library Developer CommandLineTools usr bin c g isysroot Library Developer CommandLineTools SDKs MacOSX10 sdk **Wl**

6.6.1 Variable Documentation

6.6.1.1 Wl

```
Library Developer CommandLineTools usr bin c g isysroot Library Developer CommandLineTools S↩
DKs MacOSX10 sdk search_paths_first Wl
```

Definition at line 1 of file link.txt.

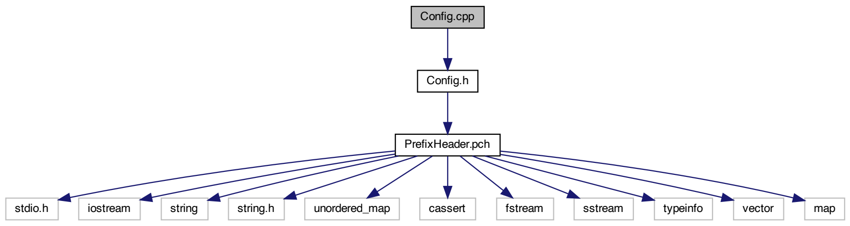
6.7 cmake-build-debug/CMakeFiles/TargetDirectories.txt File Reference

6.8 CMakeLists.txt File Reference

6.9 Config.cpp File Reference

```
#include "Config.h"
```

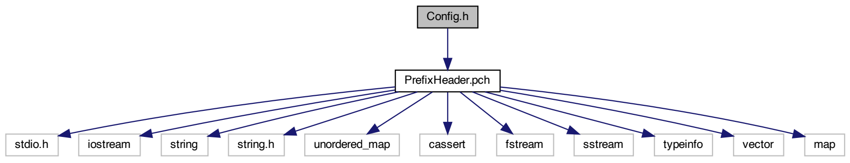
Include dependency graph for Config.cpp:



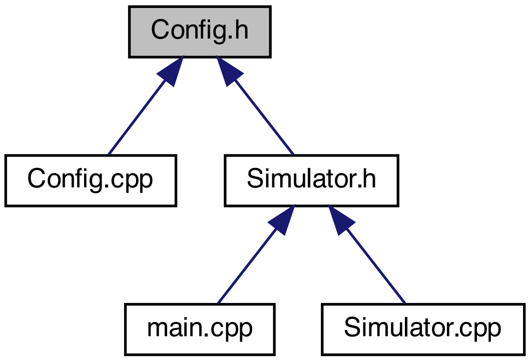
6.10 Config.h File Reference

```
#include "PrefixHeader.pch"
```

Include dependency graph for Config.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **Config**
- struct **Config::ConfigValue**

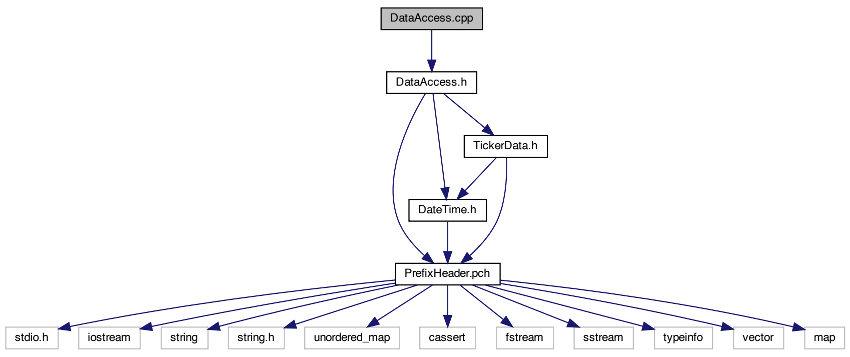
    Struct to hold configuration values and access flag.

6.11 data/constituents/universe.txt File Reference

6.12 DataAccess.cpp File Reference

```
#include "DataAccess.h"
```

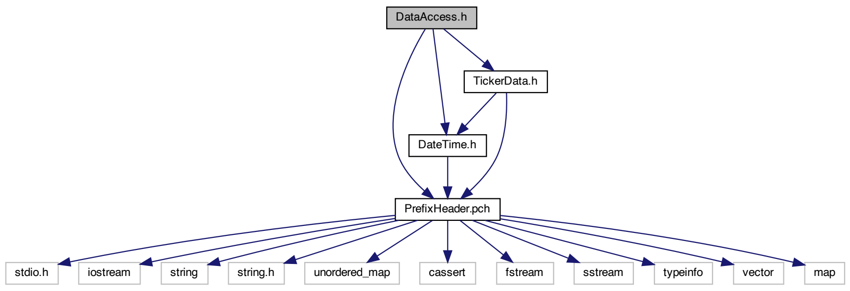
Include dependency graph for DataAccess.cpp:



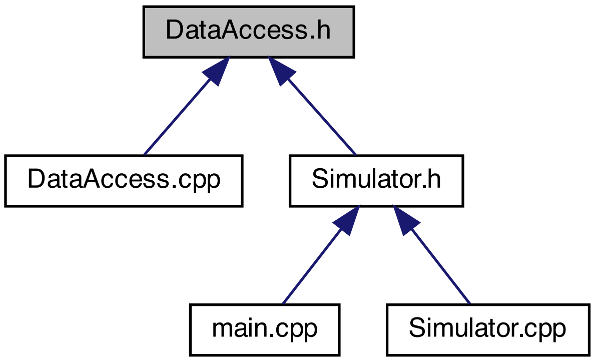
6.13 DataAccess.h File Reference

```
#include "PrefixHeader.pch"
#include "DateTime.h"
#include "TickerData.h"
```

Include dependency graph for DataAccess.h:



This graph shows which files directly or indirectly include this file:



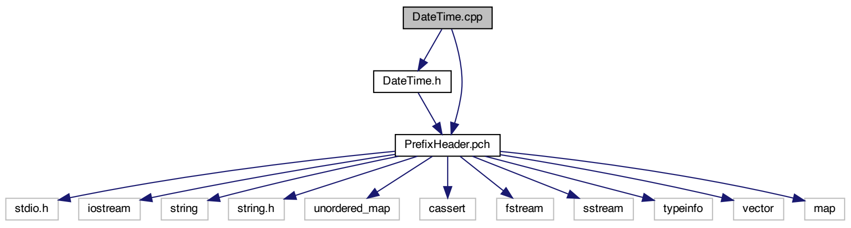
Classes

- class **DataAccess**

6.14 DateTime.cpp File Reference

```
#include "DateTime.h"
#include "PrefixHeader.pch"
```

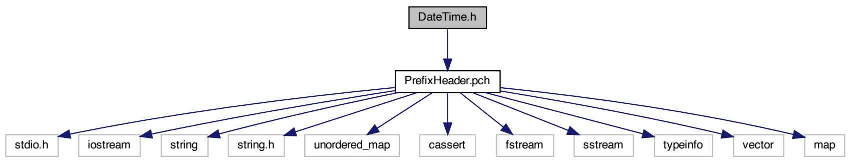
Include dependency graph for DateTime.cpp:



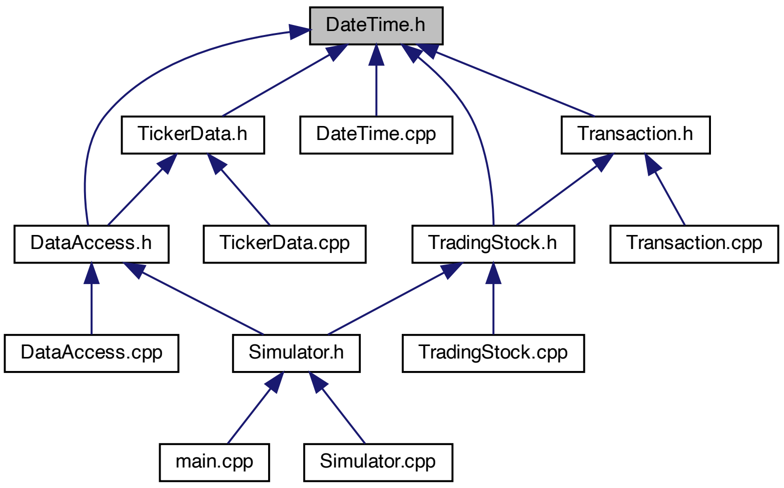
6.15 DateTime.h File Reference

```
#include "PrefixHeader.pch"
```

Include dependency graph for DateTime.h:



This graph shows which files directly or indirectly include this file:



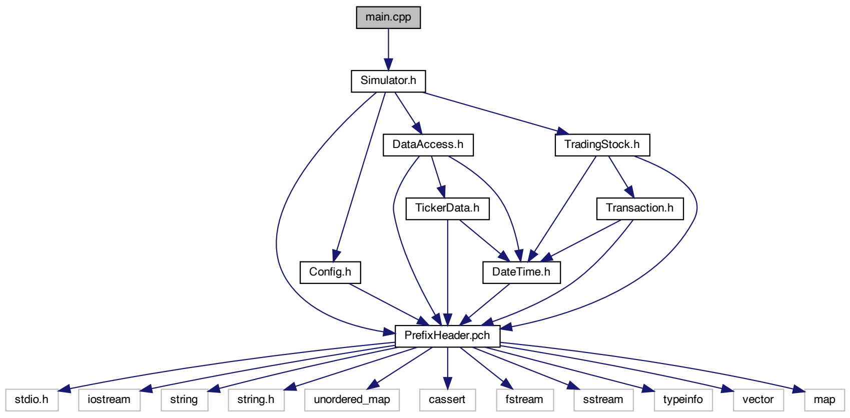
Classes

- class **DateTime**

6.16 main.cpp File Reference

```
#include "Simulator.h"
```

Include dependency graph for main.cpp:



Functions

- **int main** (**int** argc, const **char** ∗argv[ ])

6.16.1 Function Documentation

6.16.1.1 main()

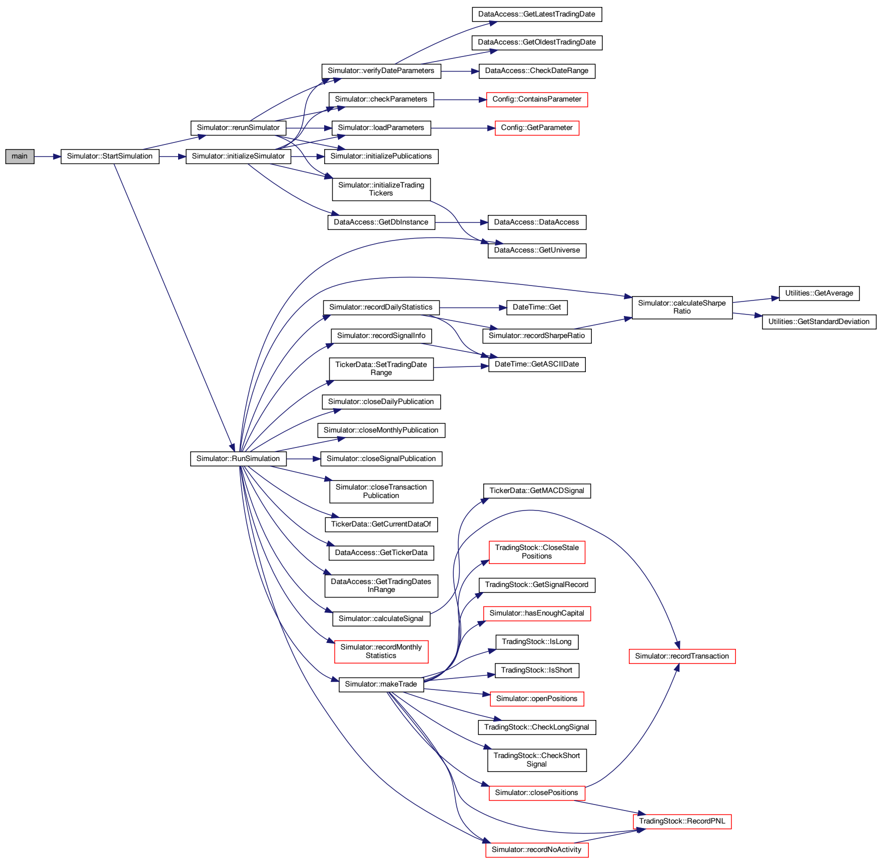
```
int main (
            int argc,
            const char * argv[] )
```

main.cpp (p. **??**) Main entry point for Dorothy financial simulator.

Created by Salil Maharjan on 3/22/20. Copyright © 2020 Salil Maharjan. All rights reserved.

Definition at line 15 of file main.cpp.

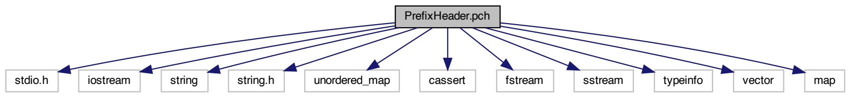
Here is the call graph for this function:


main
Simulator::StartSimulation
Simulator::rerunSimulator
Simulator::initializeSimulator
DataAccess::GetLatestTradingDate
DataAccess::GetOldestTradingDate
Simulator::verifyDateParameters
DataAccess::CheckDateRange
Simulator::checkParameters
Config::ContainsParameter
Simulator::loadParameters
Config::GetParameter
Simulator::initializePublications
Simulator::initializeTrading Tickers
DataAccess::GetDbInstance
DataAccess::DataAccess
DataAccess::GetUniverse
Utilities::GetAverage
Simulator::calculateSharpe Ratio
Utilities::GetStandardDeviation
Simulator::recordDailyStatistics
DateTime::Get
Simulator::recordSignalInfo
Simulator::recordSharpeRatio
TickerData::SetTradingDate Range
DateTime::GetASCIIDate
Simulator::closeDailyPublication
Simulator::closeMonthlyPublication
Simulator::RunSimulation
Simulator::closeSignalPublication
Simulator::closeTransaction Publication
TickerData::GetMACDSignal
TickerData::GetCurrentDataOf
TradingStock::CloseStale Positions
DataAccess::GetTickerData
DataAccess::GetTradingDates InRange
TradingStock::GetSignalRecord
Simulator::calculateSignal
Simulator::hasEnoughCapital
TradingStock::IsLong
Simulator::recordMonthly Statistics
Simulator::recordTransaction
TradingStock::IsShort
Simulator::makeTrade
Simulator::openPositions
TradingStock::CheckLongSignal
TradingStock::CheckShort Signal
Simulator::closePositions
TradingStock::RecordPNL
Simulator::recordNoActivity

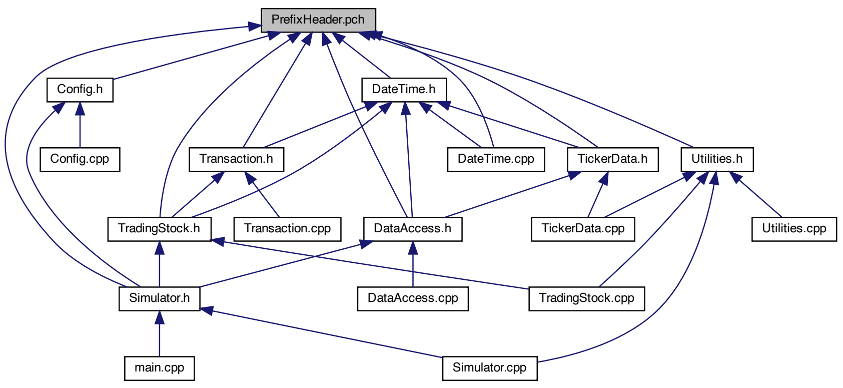
6.17 PrefixHeader.pch File Reference

```
#include <stdio.h>
#include <iostream>
#include <string>
#include <string.h>
#include <unordered_map>
#include <cassert>
#include <fstream>
#include <sstream>
#include <typeinfo>
#include <vector>
#include <map>
```

Include dependency graph for PrefixHeader.pch:



This graph shows which files directly or indirectly include this file:



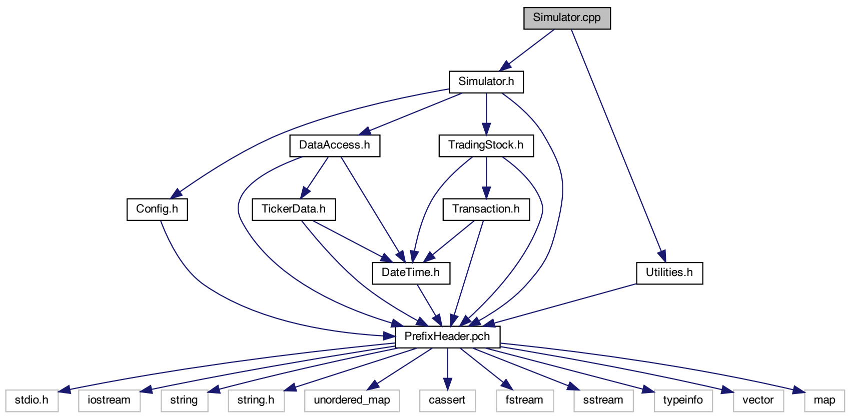
6.18 publications/TransactionReport0.txt File Reference

6.19 publications/TransactionReport1.txt File Reference

6.20 Simulator.cpp File Reference

```
#include "Simulator.h"
#include "Utilities.h"
```

Include dependency graph for Simulator.cpp:

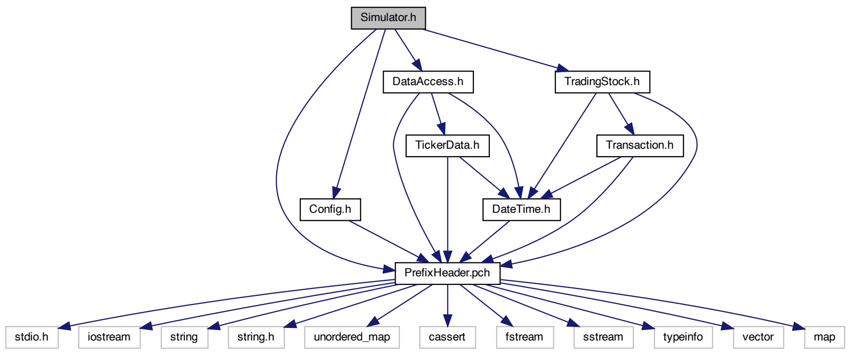


6.21 Simulator.h File Reference

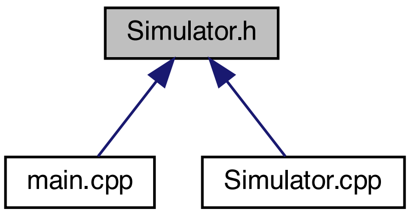
```
#include "PrefixHeader.pch"
#include "Config.h"
```

```
#include "DataAccess.h"
#include "TradingStock.h"
```

Include dependency graph for Simulator.h:



This graph shows which files directly or indirectly include this file:



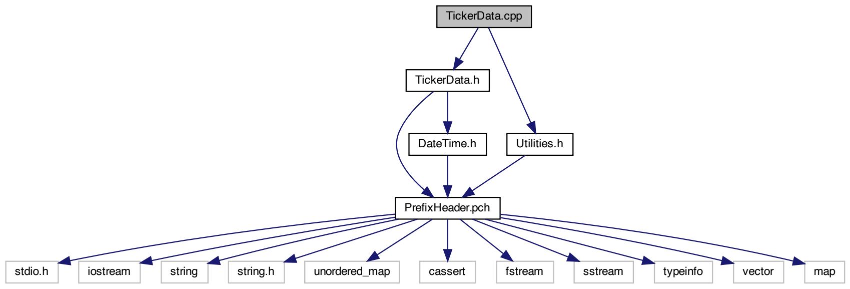
Classes

- class **Simulator**

6.22 TickerData.cpp File Reference

```
#include "TickerData.h"
#include "Utilities.h"
```

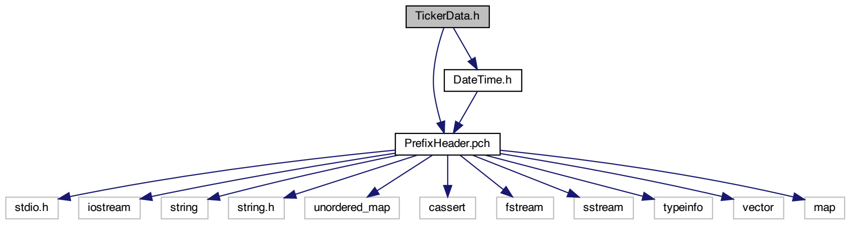
Include dependency graph for TickerData.cpp:



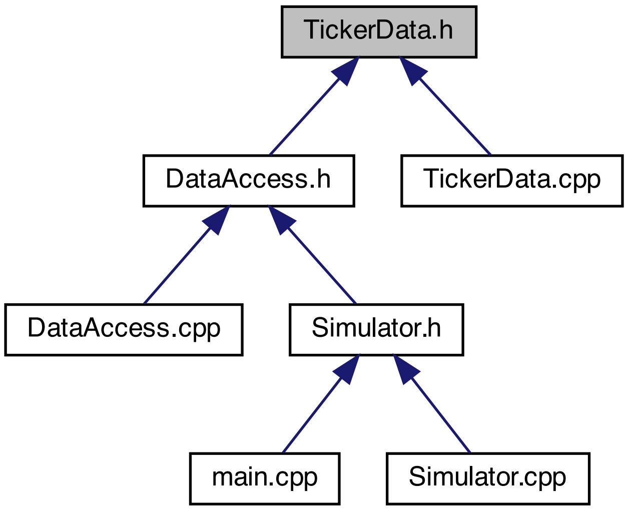
6.23 TickerData.h File Reference

```
#include "PrefixHeader.pch"
#include "DateTime.h"
```

Include dependency graph for TickerData.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **TickerData**

Enumerations

- enum **TICKER_FIELDS** {
  OPEN, **START_TICKER_FIELDS** = OPEN, **DATA_FIELD_START** = OPEN, **HIGH**,
  LOW, **CLOSE**, **VOLUME**, **ADJ_CLOSE**,
  DIVIDEND, **SPLIT**, **VWAP**, **SHARE_OUTSTANDING**,
  DATA_FIELD_END = SHARE_OUTSTANDING, **FAST_EMA**, **SLOW_EMA**, **MACD_LINE**,
  SIG_LINE, **MACD_HIST**, **END_TICKER_FIELDS** }

6.23.1 Enumeration Type Documentation

6.23.1.1 TICKER_FIELDS

```
enum TICKER_FIELDS
```

TickerData.h (p. **??**) Handles ticker data for the trading stocks.

Created by Salil Maharjan on 4/30/20. Copyright © 2020 Salil Maharjan. All rights reserved. ENUM of Ticker fields as found in the Ticker data source file. Includes computed fields used in the simulation

Enumerator

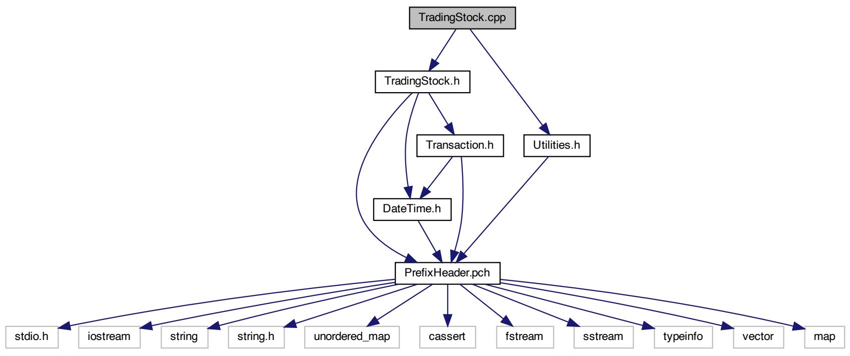
OPEN	
START_TICKER_FIELDS	
DATA_FIELD_START	
HIGH	
LOW	
CLOSE	
VOLUME	
ADJ_CLOSE	
DIVIDEND	
SPLIT	
VWAP	
SHARE_OUTSTANDING	
DATA_FIELD_END	
FAST_EMA	
SLOW_EMA	
MACD_LINE	
SIG_LINE	
MACD_HIST	
END_TICKER_FIELDS	

Definition at line 22 of file TickerData.h.

6.24 TradingStock.cpp File Reference

```
#include "TradingStock.h"
#include "Utilities.h"
```

Include dependency graph for TradingStock.cpp:

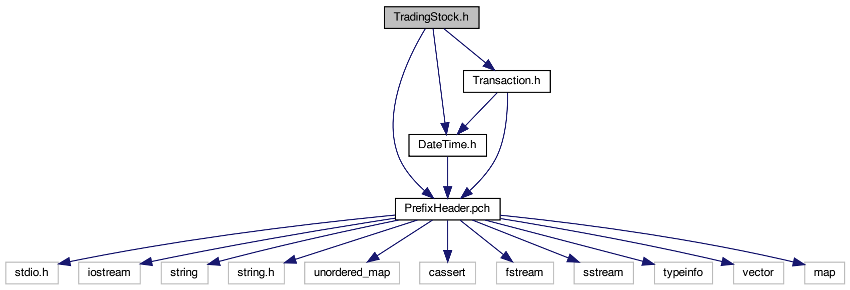


6.25 TradingStock.h File Reference

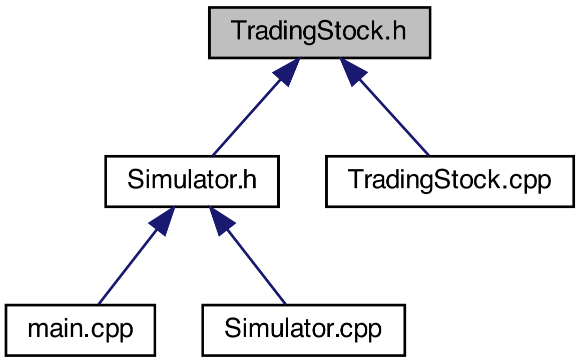
```
#include "PrefixHeader.pch"
#include "DateTime.h"
```

`#include "Transaction.h"`

Include dependency graph for TradingStock.h:



This graph shows which files directly or indirectly include this file:



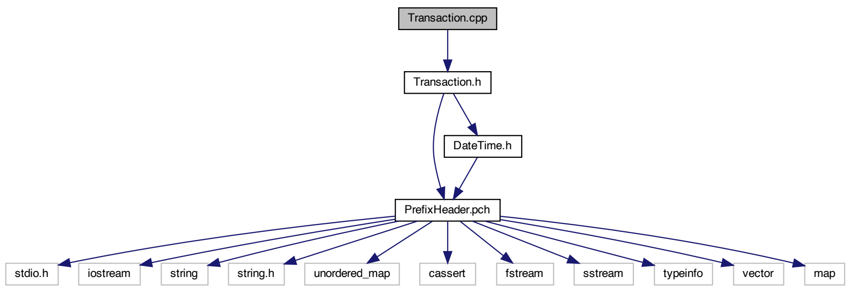
Classes

- class **TradingStock**

6.26 Transaction.cpp File Reference

`#include "Transaction.h"`

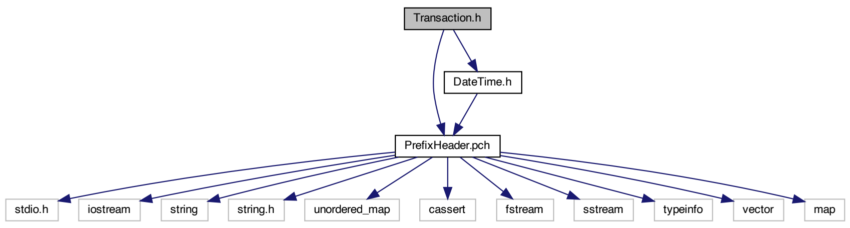
Include dependency graph for Transaction.cpp:



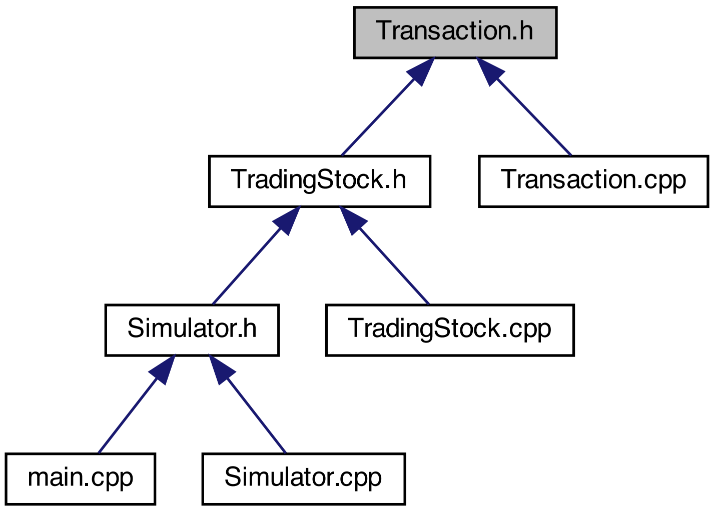
6.27 Transaction.h File Reference

```
#include "PrefixHeader.pch"
#include "DateTime.h"
```

Include dependency graph for Transaction.h:



This graph shows which files directly or indirectly include this file:



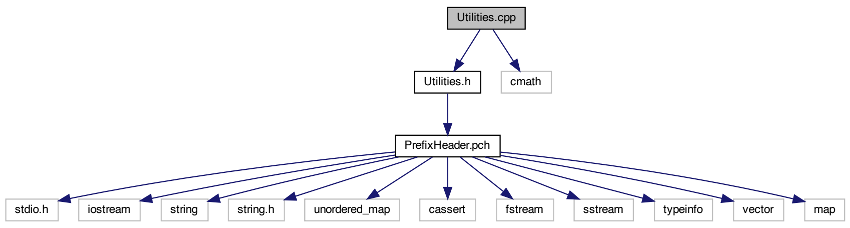
Classes

- class **Transaction**

6.28 Utilities.cpp File Reference

```
#include "Utilities.h"
#include <cmath>
```

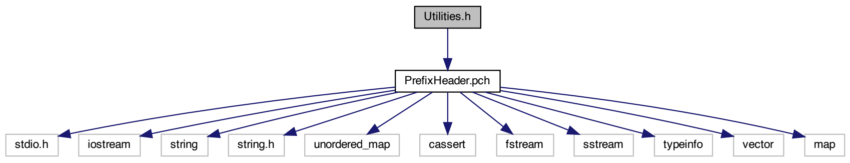
Include dependency graph for Utilities.cpp:



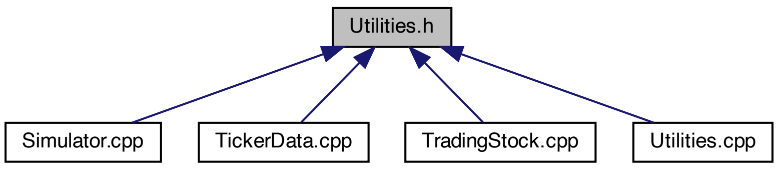
6.29 Utilities.h File Reference

`#include "PrefixHeader.pch"`

Include dependency graph for Utilities.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- **Utilities**

Functions

- **int Utilities::RoundOff** (double a_value)

  Function to round off value to the lower integral value.
- double **Utilities::GetAverage** (std::vector< double > a_list)

  Function to get average of a list with doubles.
- double **Utilities::GetStandardDeviation** (std::vector< double > a_list, double a_average)

  Function to get the standard deviation of a list with doubles.
- void **Utilities::trimBlanks** (std::string &a_str)

  Method to trim leading and trailing blanks while reading data.