Инструкция по установке и эксплуатации ПО «Arbitrage MOEX / Арбитраж MOEX»

Программа «Arbitrage MOEX / Арбитраж MOEX» (далее Программа) представляет собой дистрибутив (файл tar.gz), внутри которого находится код, написанный на языке C++. Данная программа должна быть встроена в ПО клиента для анализа биржевых данных или торговли на Московской Бирже.

1. БАЗОВОЕ СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Системные программные средства, для которых обеспечивается эффективная работа Программы:

- операционная система для компиляции: Ubuntu 20.04
- операционная система для запуска: Ubuntu 20.04

На сервере для компиляции необходимо следующее программное обеспечение:

- Компилятор с поддержкой c++23, например clang-16
- cmake v3.19.1 или новее

2. ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

Минимальные технические характеристики сервера для компиляции:

- процессор 2GHz;
- память 2GB
- свободное дисковое пространство 2GB

Минимальные технические характеристики сервера для запуска:

- процессор 2GHz;
- память 4GB;
- технические требования для ПО клиента

Рекомендуемые технические характеристики сервера для компиляции:

- 8-ядерный процессор 3GHz;
- память 32GB
- свободное дисковое пространство 2GB

Минимальные технические характеристики сервера для запуска:

- процессор 5GHz;
- память 16GB;
- технические требования для ПО клиента

Установка Программы на сервер для компиляции

Установка должна производиться на операционную систему Ubuntu 20.04
Перед установкой убедитесь, что на сервере установлены следующие пакеты:

- **clang-16**
- **cmake**

Для установки этих пакетов нужно добавить репозитории (см. инструкции ниже), а потом установить, используя стандартный интерфейс для установки программ в Ubuntu 20.04:
sudo apt update && sudo apt install cmake clang-16

Инструкции для добавления репозиториев:

- https://apt.llvm.org/
- https://apt.kitware.com/

Для использования, распакуйте полученный дистрибутив (файл tar.gz) и разместите полученную директорию arbitrage внутри вашего проекта. Для подключения Программы к своему ПО, добавьте в CMakeLists.txt вашего проекта
add_subdirectory(arbitrage)
target_link_libraries(ИМЯ_ВАШЕГО_ПРИЛОЖЕНИЯ arbitrage)

а в c++ файлах добавьте
#include "arbitrge/arbitrage.h"

и используйте класс
ArbitrageMicexRtsPricePredictor

Данный класс имеет следующие интерфейсы:
ArbitrageMicexRtsPricePredictor(std::string config_filename); // инициализирует настройки прогнозирования, включая инструмент на котором предсказываются цены, инструменты которые используется для этого прогнозирования и параметры из json-файла конфига. Примеры рекомендуемых конфигов содержатся в дистрибутиве.

void process(const PriceLimitSnapshot& snapshot);
void process(const TradingStatusSnapshot& snapshot);
void process(const BookSnapshot& snapshot);
void process(const TradesSnapshot& snapshot);
double get_price(); # Возвращает текущее предсказание цены в зависимости от настроек прогнозирования и переданных ранее данных

Подробная информация по используемым классам предоставляется клиенту в рамках работ по первоначальной настройке Программы.

После этого ПО клиента будет доступно к сборке с подключённой Программой.

Эксплуатация скомпилированного экземпляра Программы

Для использования Программы необходимы данные с биржевой информацией. Если клиентское ПО предназначается для торговли на Московской Бирже, требуется организовать получение биржевой информации на торговый сервер в режиме онлайн. Если клиентское ПО предназначается для анализа биржевых данных, требуется подготовить данные с биржевой информацией в заданном формате.

Использование программы заключается в создании внутри клиентского ПО экземпляра класса *ArbitrageMicexRtsPricePredictor* и передаче биржевых данных в его методы

Для создания класса используется конструктор *ArbitrageMicexRtsPricePredictor(std::string config_filename);* в который нужно передать путь до json-файла с параметрами. Данные параметры представляют собой настройки прогнозирования, такие как инструмент, на котором предсказываются цены, прочие инструменты, биржевая информация о которых используется для прогнозирования, и различные параметры учёта биржевой информации.

Данные с биржевой информацией могут включать в себя следующее:

- Добавление, удаление и модификация заявок на покупку или продажу биржевых инструментов. На основании данной информации на стороне клиентского ПО должна формироваться книга заявок, по соответствующему инструменту. Данная книга заявок передаётся в Программу через интерфейс *void process(const BookSnapshot& snapshot);*
- Сделки, совершаемые на бирже по заданным биржевым инструментам. Данные сделки передаются в Программу через интерфейс *void process(const TradesSnapshot& snapshot);*
- Информация о статусе торгов заданными биржевыми инструментами согласно биржевой документации, опубликованной по адресу http://ftp.moex.com/. Статусы торгов передаются в Программу через интерфейс *void process(const TradingStatusSnapshot& snapshot);*
- Лимиты допустимых изменений цен биржевых инструментов. Лимиты передаются в Программу через интерфейс *void process(const PriceLimitSnapshot& snapshot);*

Наличие информации о книге заявок для заданных биржевых инструментов является необходимым условием для работы Программы. Наличие остальной биржевой информации повышает точность прогнозирования. Для получения наилучшей точности прогнозирования требуется отдавать в Программу данные с биржевой информацией после каждого изменения.

Обращаем ваше внимание на то, что биржевая информация не является публично доступной по умолчанию! При необходимости клиенты должны самостоятельно заключать договоры с Московской Биржей на получение и использование биржевой информации. Подробнее с политикой распространения биржевой информации можно ознакомиться на сайте ПАО «Московская Биржа» по адресу https://www.moex.com/ru/datapolicy/

Внутри клиентского ПО проинициализированный экземпляр Программы должен получать биржевую информацию по заданным биржевым инструментам. Для получения текущего предсказания цены в зависимости от заданных настроек прогнозирования и переданных биржевых данных требуется осуществить вызов функции *double get_price();*. Полученное предсказание цены инструмента может в дальнейшем быть использовано в ПО клиента для анализа биржевых данных или торговли на Московской Бирже.