1FX первый финансовый

# Руководство пользователя cAlgo

Содержание:

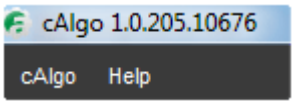

Пользовательский интерфейс....3
Главное меню....3
Панели инструментов....3
Account Bar (Панель торгового счета)....4
Панель статуса....5
Вкладки....5
Список роботов и индикаторов....6
Окно торгового терминала....7
Positions (Ордера)....7
Orders (Отложенные ордера)....8
History (История)....9
Интерфейс программирования приложений: индикаторы....11
Создать индикатор....11
Использование вложенного индикатора....13
NaN-арифметика....14
Уроки работы с cAlgo....16
Создание торгового робота....16
Торговля....19
Поддержка....22
Интерфейс программирования приложений: роботы (cAlgo API)....23
Начало....23
Торговля....24
Рабочие параметры....26
Использование индикаторов....27
Свойства символов....28
Информация по счету....28
Скопировать в log робота....29
FAQ....30
Начало работы....32
Настройки подключения....32
cAlgo: автоматическое обновление....33
cAlgo: установка и запуск....33
Демо-счета....34
Торговые счета....36

# Пользовательский интерфейс

Данный раздел содержит информацию о разных областях пользовательского интерфейса cAlgo.

## Главное меню

Из главного меню можно открыть следующие вкладки:




### cAlgo

**Login (Вход)**
Вход в торговый или демо-счет, создание нового демо-счета или изменение настроек подключения.

**Open Demo Account (Открыть демо-счет)**
Заполните данные формы для открытия демо-счета.

**Exit CAlgo (Выйти из cAlgo)**
Закрыть торговый терминал. Все запущенные роботы будут остановлены, но открытые торговые позиции не будут закрыты.

### Помощь

**Help Guide (Раздел помоши) (F1)**
Открыть раздел помощи cAlgo и учебное пособие (данный документ). Здесь объяснены в деталях функции и настройки и задачи торговой платформы cAlgo. Также вы можете нажать **F1** для того, чтобы открыть этот документ.

**About cAlgo (О cAlgo)**
Здесь вы сможете найти контактную информацию разработчиков и информацию о текущей версии платформы cAlgo.

## Панели инструментов

В платформу встроены 3 панели инструментов:

**Account Bar (Панель торгового счета)**
Доступ к вашему торговому счету.

**Quick Links (Вкладки)**
Запустить терминал cTrader одним кликом мышки, связь с разработчиками cTrader и опция полного экрана.

**Status Bar (Панель статуса)**
Информация о Сервере и соединении.

## Account Bar (Панель торгового счета)

Панель торгового счета (**Account Bar)** – это очень простой способ для пользователей для переключения между множеством торговых счетов. Содержит раскрывающийся список счетов **(Dropdown List)** и кнопку подключения к счету **(Account Button)**.

Панель торгового счета отображает:


- Номер торгового счета
- Демо (Demo) или реальный (Live) счет
- Базовую валюту депозита

### Кнопка подключения к счету 

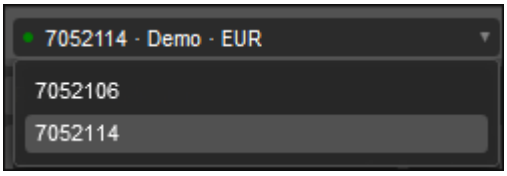
Нажмите эту кнопку, чтобы войти в окно счетов, со следующими опциями:

**Sign In (Подключение к счету)**
Подключиться к вашему демо-счету или реальному счету.

**Demo Account (Демо-счет)**
Создать демо-счет и начать обучающую торговлю виртуальными средствами.

**Connection Settings (Настройки подключения)**
Вручную настройте прокси, если необходимо соединение от конкретной корпоративной сети или интернет-провайдера.

### Раскрывающийся список счетов)




**Выберите торговый счет, из списка доступных счетов.**

**Важно**: чтобы счет появился в данном списке, вам необходимо войти под данным счетом в cAlgo **(cAlgo > Login)** и проверить поле сохранения пароля **(Remember Password)**.

**Подключитесь к другому счету,** выбрав его из данного списка.

**Чтобы удалить счет** из списка счетов cTrader, нажмите мышкой на номер данного счета и затем нажмите на кнопку **X**.

Счет будет удален из списка после подтверждения удаления..

**Внимание:** *Удаление счета из списка не приводит к его удалению у вашего брокера. Удалив торговый счет, Вы вновь сможете к нему подключиться, введя его данные. Если вам необходимо полностью закрыть ваш счет, пожалуйста, свяжитесь с вашим брокером.*

## Панель статуса

**Панель статуса** находится внизу платформы cAlgo.


**Trading Sessions (Торговые сессии)**
Здесь отображаются открытые на данный момент торговые сессии. (например, London, Frankfurt, New York, Singapore, и т.д.).

**Server Time (Серверное время)**
Время сервера cAlgo. Изменение цен на графике происходит в соответствии с серверным временем. (UTC или GMT+0).

**In/Out Traffic (Входящий/Исходящий трафик)**
Входящий и исходящий трафик торговой платформы, в килобайтах (kB).

## Вкладки

### cTrader

Нажмите данную кнопку для доступа к платформе **cTrader**.

> **Если cTrader уже установлен**, приложение будет запущено.
>
> **Если cTrader установлен и запущен**, cTrader будет активирован.
>
> **Если cTrader не установлен**, нажатие на cTrader запустит процесс установки. После завершения установки, cTrader будет запущен.

### cTDN

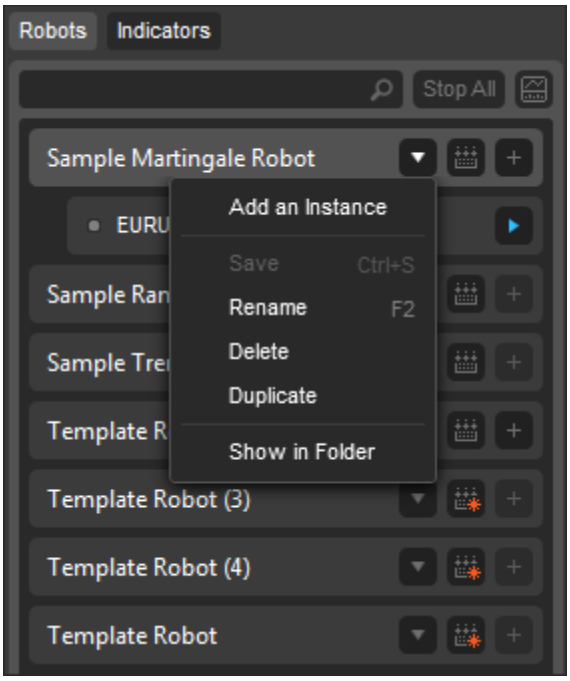
Нажмите данную кнопку для доступа к **Ресурсу Разработчиков cTrader**, где вы найдете **форумы**, **блоги** и **специальные материалы** для программирования. cTDN создан, чтобы помочь трейдерам делиться опытом, ресурсами и знаниями об использовании роботов и пользовательских индикаторов в дружеской профессиональной среде.

### Windowed Mode / Full Screen Mode (В окне/Полный экран)

Нажмите для перемещения между режимами «В окне» и «Полный экран».

## Список роботов и индикаторов

Вы можете видеть полный список **Роботов** или **Пользовательских индикаторов** в панели платформы, слева.




**Robots / Indicators (Роботы/Индикаторы)**
Переключение между двумя списками - роботов и индикаторов.

**Search (Поиск)**
Позволяет найти названия роботов и индикаторов. Список фильтруется в соответствии с вашими запросами.

**Add (Добавить)**
Создать нового робота и открыть редактор исходного кода для написания вашего алгоритма.

**Stop All (Остановить все)**
Остановить всех запущенных роботов.

**Play (Запуск)**
Запуск робота, который начнет работу со следующего рыночного тика.

**Robot / Indicator Menu (Меню робота/индикатора)**
(смотри ниже)

**Build Button (Кнопка «Создать»)**
Нажмите, чтобы перевести коды в редакторе в функционирующего робта.

**Add an Instance (Robots) (Добавить на график Робота)**
Прикрепит созданного робота к графику и создаст список настраиваемых параметров, обозначенных в коде, которые можно изменить перед тем, как запустить робота для торговли.

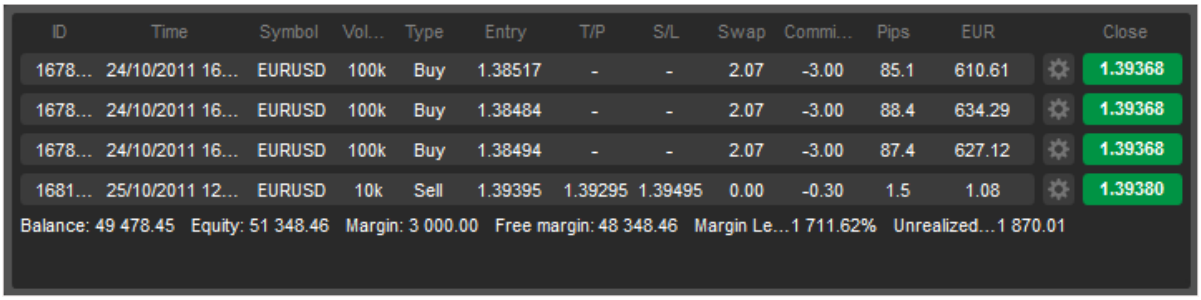
**Add an Instance (Indicators) (Добавить на график Индикатор)**
Прикрепить ваш пользовательский индикатор к графику и изменяя, при необходимости, его параметры. В таблице индикатора будут доступны для настройки те параметры, которые были указаны в коде.

# Окно торгового терминала

Здесь содержится информация о всех ваших открытых, отложенных и закрытых ордерах.

## Positions (Ордера)

Здесь отображаются все открытые ордера.



| ID | Time | Symbol | Vol... | Type | Entry | T/P | S/L | Swap | Commi... | Pips | EUR | Close |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1678... | 24/10/2011 16... | EURUSD | 100k | Buy | 1.38517 | - | - | 2.07 | -3.00 | 85.1 | 610.61 | 1.39368 |
| 1678... | 24/10/2011 16... | EURUSD | 100k | Buy | 1.38484 | - | - | 2.07 | -3.00 | 88.4 | 634.29 | 1.39368 |
| 1678... | 24/10/2011 16... | EURUSD | 100k | Buy | 1.38494 | - | - | 2.07 | -3.00 | 87.4 | 627.12 | 1.39368 |
| 1681... | 25/10/2011 12... | EURUSD | 10k | Sell | 1.39395 | 1.39295 | 1.39495 | 0.00 | -0.30 | 1.5 | 1.08 | 1.39380 |

Balance: 49 478.45 Equity: 51 348.46 Margin: 3 000.00 Free margin: 48 348.46 Margin Le...1 711.62% Unrealized...1 870.01

### Данные окна ордеров

- **ID –** Уникальный ticket-номер ордера.
- **Time** – Дата и время размещения ордера. Отображается как DD/MM/YYYY HH:MM (День/Месяц/Год Час:Минута).
- **Symbol** – Символ валютной пары.
- **Volume** – Общий объем сделки.
- **Type** – Тип ордера: **Buy (Покупка)** или **Sell (Продажа)**.
- **Entry** – Цена, по которой ордер был установлен.
- **T/P** - Уровень **Take Profit**. Если он не был выставлен, в этом поле будет отображаться знак ( - ).
- **S/L** - Уровень **Stop Loss**. Если он не был выставлен, в этом поле будет отображаться знак ( - ).
- **Swap** – Общий размер **swap**.
- **Commission** – Общая комиссия, взимаемая брокером за даннуб торговую операцию.
- **Pips** – Прибыль или убыток, в пунктах.
- **Base Currency P&L** – Прибыль или убыток, в базовой валюте.
- **Close** – Отображает текущую рыночную котировку. Нажмите **close**, чтобы закрыть сделку.

### Modify Stop Loss or Take Profit

**Take profit** and **Stop Loss** levels can be changed during trading unless prevented by the code. To modify, **right click** on the position and click **Modify Position**.

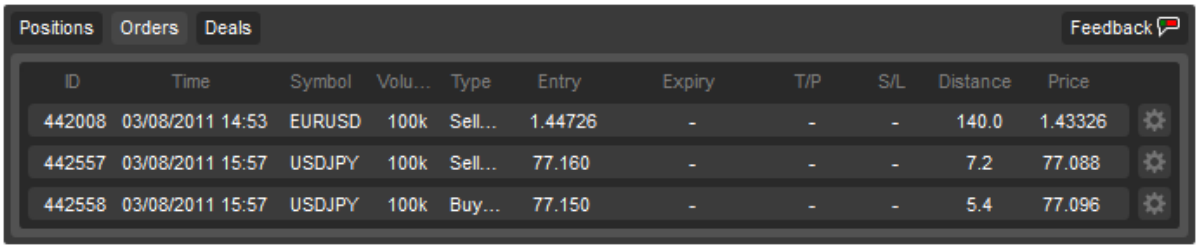
### Закрытие торговой позиции

Открытая позиция  будет автоматически закрыта, когда цена достигнет уровня **Stop Loss** или **Take Profit**.

Для того, чтобы закрыть позицию вручную, нажмите на котировку, в колонке **Close (Закрыть)**, или нажмите **правой кнопкой** на строку ордера и выберите **Close Position (Закрыть позицию)**.

## Orders (Отложенные ордера)

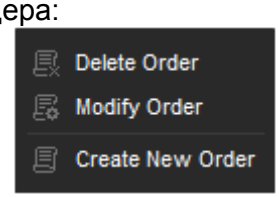
Вкладка **Orders** содержит все отложенные ордера, которые были выставлены вашими роботами. Кликнув на оглавление, вы можете отсортировать ордера.



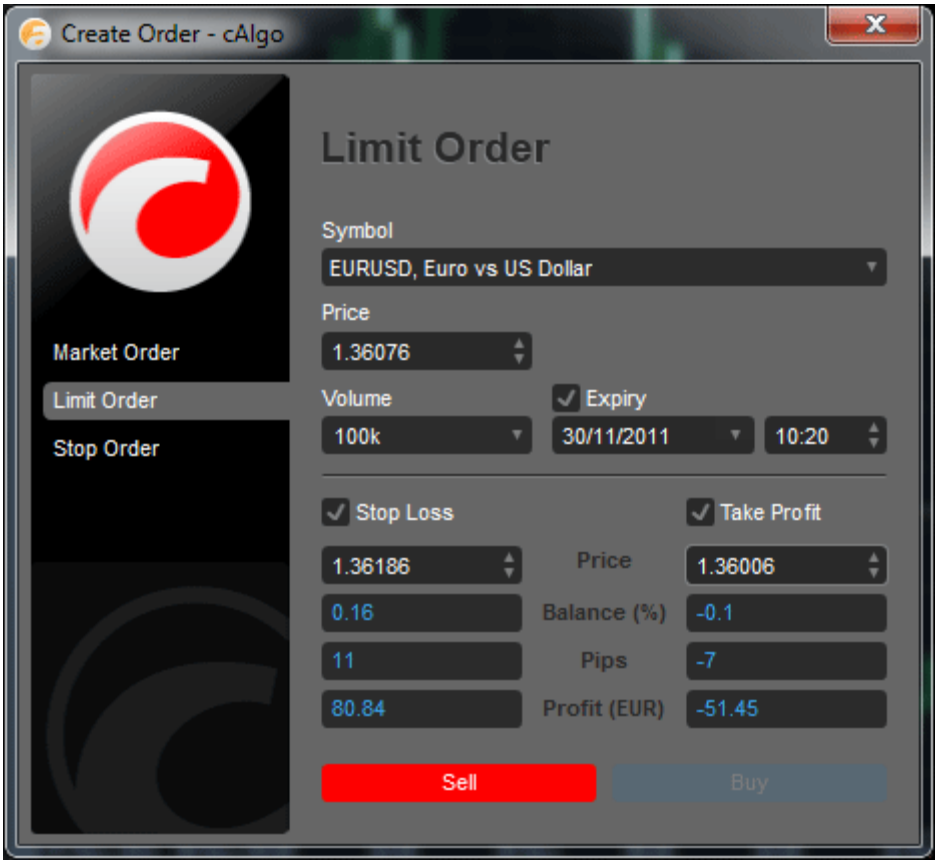

### Данные окна отложенных ордеров

- **ID -** Операционный ticket-номер ордера, уникальный для каждого отложенного ордера.
- **Time -** Дата и время размещения ордера. Отображается как DD/MM/YYYY HH:MM (День/Месяц/Год Час:Минута).
- **Symbol** - Символ валютной пары.
- **Volume -** Общий объем сделки.
- **Type -** Четыре типа ордеров: **Sell Stop**, **Sell Limit**, **Buy Stop** и **Buy Limit**.
- **Entry -** Цена, по которой ордер был установлен.
- **Expiry** – Если параметр установлен, он отображает выбранное вами время, когда ордер будет удален.
- **T/P -** Уровень **Take Profit**. Если он не был выставлен, в этом поле будет отображаться знак ( - ).
- **S/L -** Уровень **Stop Loss**. Если он не был выставлен, в этом поле будет отображаться знак ( - ).
- **Distance** - Дистанция от текущей цены для входа, в .
- **Price/Close** - Отображает текущую рыночную котировку. Нажмите **эту кнопку**, чтобы закрыть сделку..

**Клик правой кнопкой мыши**
Данные действия можно произвести, кликнув правой кнопкой мыши по строке ордера:

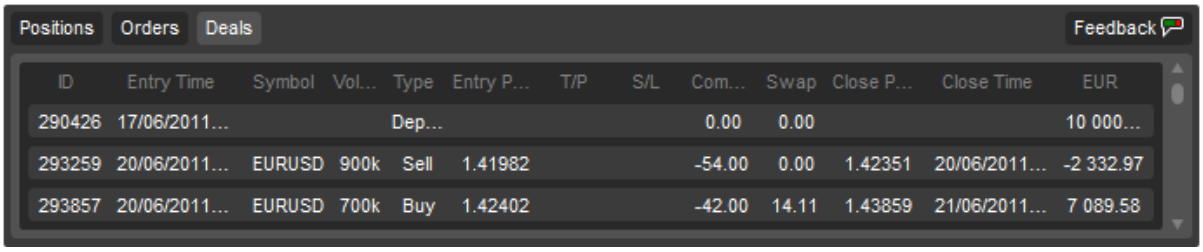



- **Delete order (Удалить ордер):** Ордер будет удален.
- **Modify order (Изменить ордер):** Позволит вам изменить цену открытия ордера, уровни Stop Loss и Take Profit и установить дату отмены ордера. Вы сможете ввести эти же параметры, нажав на иконку:
- **Create new order (Новый ордер):** Появится окно открытия нового ордера.




## History (История)

Раздел «История» содержит информацию о всех закрытых позициях и ордерах. Данная информация представляет финансовые результаты торговли и параметры ордеров, которые включаю: ‘Realized P&L’ (Зафиксированная прибыль/убыток), ‘Deposit’ (Пополнения) и ‘Withdrawal’ (Снятия).




### Параметры раздела История

- **ID -** Уникальный ticket-номер ордера.
- **Time** - Дата и время размещения рыночного или отложенного ордера. Отображается как DD/MM/YYYY HH:MM (День/Месяц/Год Час:Минута).
- **Symbol** - Символ валютной пары.
- **Volume** - Общий объем сделки.
- **Type** – Закрытые рыночные ордера **Buy** or **Sell**. Закрытые отложенные ордера **Sell Stop**, **Sell Limit**, **Buy Stop** or **Buy Limit**
- **Entry** - Цена, по которой ордер был установлен.
- **T/P** - Уровень **Take Profit**. Если он не был выставлен, в этом поле будет отображаться знак ( - ).
- **S/L** - Уровень **Stop Loss**. Если он не был выставлен, в этом поле будет отображаться знак ( - ).
- **Swap** – Общий размер **swap**.

- **Commission** - Общая комиссия, взимаемая брокером за данную торговую операцию
- **Close Price** – Цена, по которой позиция была закрыта
- **Close Time** – Дата и время закрытия позиции. Отображается как DD/MM/YYYY HH:MM (День/Месяц/Год Час:Минута).
- **P&L** – Общая прибыль или убыток закрытой позиции.

Дополнительная информация о торговой истории отображена снизу платформы:

**Realized P&L**: Общая прибыль или убыток всех закрытых позиций.

**Deposit**: Общий размер депозита.

**Withdrawals**: Общая сумма снятий средств.

## Statement (Отчет о торговле)


С помощью кнопки «statement», находящейся в правом нижнем углу, вы можете создавать отчет о истории торговли в формате html. Нажатие на нее откроет папку **My Documents**, куда данный отчет будет сохранен.

Двойной щелчок откроет отчет.Также вы можете создать отчет, нажав правой кнопкой мыши на любую сделку из окна ордеров.

**Positions**

| ID | Time | Symbol | Volume | Type | Entry | T/P | S/L | Pips | Swap | Commissions | EUR |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1678080 | 24/10/2011 16:41 | EURUSD | 100k | Buy | 1.38484 | - | - | 86.8 | 4.14 | -3.00 | 622.88 |
| 1678086 | 24/10/2011 16:42 | EURUSD | 100k | Buy | 1.38494 | - | - | 85.8 | 4.14 | -3.00 | 615.71 |
| 1685521 | 26/10/2011 11:18 | EURUSD | 10k | Buy | 1.39365 | 1.39465 | 1.39265 | -1.3 | 0.00 | 0.00 | -0.93 |
| | | | | | | | | | 8.28 | -6.00 | 1 237.66 |
| | | | | | | | | Unrealized P&L: | | | 1 239.94 |

# Интерфейс программирования приложений: индикаторы

## Создать индикатор

### Indicator class (Класс индикаторов)

Когда вы создаете новый пользовательский индикатор, вы можете видеть предустановленные шаблоны индикаторов в редакторе первичного кода:

```
using System;
using cAlgo.API;
using cAlgo.API.Indicators;

namespace cAlgo.Indicators
{
    [Indicator(IsOverlay = false)]
    public class NewIndicator : Indicator
    {
        [Parameter(DefaultValue = 0.0)]
        public double Parameter { get; set; }

        [Output("Main")]
        public IndicatorDataSeries Result { get; set; }


        protected override void Initialize()
        {
            // Initialize and create nested indicators
        }

        public override void Calculate(int index)
        {
            // Calculate value at specified index
            // Result[index] = ...
        }
    }
}
```

`NewIndicator` класс является базовым классом программы – индикатором `Indicator`. Для того, чтобы использовать этот класс как индикатор, необходимо отметить его атрибутом `[Indicator]`. Он уже отмечен данным образом в шаблоне.

`IsOverlay` означает, накладывается ли данный индикатор на ценовой график или нет. Если это overlay-индикатор (`IsOverlay = true`), он будет накладываться на график, например, все индикаторы moving являются overlay-индикаторами. Если данный индикатор не является overlay-индикатором (`IsOverlay = false`), он будет открыт в новом окне.

### Parameters (Параметры)

`Parameter` - настройка параметров индикатора, которая может быть использована для уточнения некоторых более важных внешних значений, необходимых для построения индикатора. Более подробная информация - в разделе "Заявленные параметры".

### Result series (Исходные данные)

`Result` - это серии данных, хранящие результаты вычислений. Чтобы индикатор отображался на графике, исходные данные индикатора должны принадлежать типу `IndicatorDataSeries`, маркированному как `[Output]`. В дополнение к этому атрибуту вы также должны указать название рисуемой линии индикатора. Несмотря на то, что в существующих шаблонных индикаторах есть только одна линия, вы можете создать их несколько. Вы также можете изменить стиль и цвет линии. В коде индикатора MACD, исходными данными являются:

```
[Output("Main", Color = Colors Turquoise, IsHistogram = true)]
public IndicatorDataSeries Histogram { get; set; }

[Output("Signal", Color = Colors Red, LineStyle = LineStyle Lines)]
public IndicatorDataSeries Signal { get; set; }
```

**Calculation (Вычисления)**

Добавляя индикатор на график, cAlgo запрашивает метод расчета `Calculate(index)` для каждой свечи. Этот метод должен содержать логику просчета каждой единицы значения индикатора. В качестве исходных значений для расчетов, вы можете использовать данные Open, High, Low, Close с выбранного участка графика, к которому прикреплен индикатор. В первом случае, мы используем объект `MarketSeries`, содержащий значения `Open`, `High`, `Low` и `Close`. Например, вычисление "High минус Low" выглядит следующим образом:

```
[Output("Main")]
public IndicatorDataSeries Result  { get; set; }

public override void Calculate(int index)
{
    Result[index] = MarketSeries High[index] - MarketSeries Low[index];
}
```

Если вы хотите позволить пользователям самостоятельно вручную устанавливать исходные серии значений для индикатора, вы должны закрепить исходные данные (**source series**), как параметр. Для этого, задайте свободный доступ к коду `DataSeries` (**public**) и отметьте его с помощью атрибута `[Parameter]`. Далее, мы сможем использовать его для расчетов. Например, ниже указан код для индикатора "Simple Moving Average":

```
[Parameter(DefaultValue = 14)]
public int Periods { get; set; }

[Parameter]
public DataSeries Source { get; set; }

[Output("Main")]
public IndicatorDataSeries Result { get; set; }

public override void Calculate(int index)
{
    double sum = 0.0;
    for (int i = index - Periods + 1; i <= index; i++)
    {
        sum = sum + Source[i];
    }
    Result[index] = sum / Periods
}
```

Все расчеты осуществляются для каждой свечи, по очереди. Если график содержит N свечей, значения исходных данных также будут содержать N численных символов. Сначала вычисление `Calculate(index)` будет осуществлено для последней свечи, index = 0, затем 1, 2, 3, ..., N-1. Затем, когда все вычисления произведены, и сформировался новый тик (последняя свеча изменена), параметр вычисления Calculate(index) будет проводиться снова для index = N-1, так, последнее значение индикатора будет пересчитано после каждого тика. Когда появится новая свеча, новое значение будет добавлено к исходным данным (source series), и параметр вычисления OnCalculate(index) будет проводиться для index = N.

## Использование вложенного индикатора

Некоторые индикаторы могут использовать другие индикаторы, встроенные в них, для подсчета конечных значений. Для создания такого индикатора, создайте вложенный индикатор, добавляя в `Initialize()` объект `Indicators` и оставляя его в поле кода. Высчитывая значения индикатора, мы можем использовать исходные данные данного индикатора. Ниже приведен пример индикатора "Detrended Price Oscilator", который использует для расчетов простые скользящие средние:

```
[Indicator "Detrended Price Oscillator"]
public class DetrendedPriceOscillator : Indicator
{
    [Parameter]
    public DataSeries Source { get; set; }

    [Parameter]
    public int Periods { get; set; }

    [Output("Main")]
    public IndicatorDataSeries Result { get; set; }

    private MovingAverage movingAverage;

    protected override void Initialize()
    {
        movingAverage = Indicators.SimpleMovingAverage(Source, Periods);
    }

    public override void Calculate(int index)
    {
        Result[index] = Source[index] - movingAverage.Result[index   Periods / 2 - 1];
    }
```

Мы также можем создать вложенный индикатор, основанный на расчетах другого вложенного индикатора. Например, индикатор "Chaikin Volatility" использует средние скользящие индикатора "High minus Low":

```
[Parameter(DefaultValue = 14)]
public int Periods { get; set; }

[Parameter("MA Type")]
public MovingAverageType MAType { get; set; }

private HighMinusLow highMinusLow;
private MovingAverage movingAverage;

protected override void Initialize()
{
    highMinusLow = Indicators.HighMinusLow();
    movingAverage = Indicators.MovingAverage(highMinusLow.Result, Periods, MAType);
```

# NaN-арифметика

## NaN-арифметика

Ниже представлен код индикатора «Осциллятор бестрендовой цены» (Detrended Price Oscilator):

```
private MovingAverage movingAverage;

protected override void Initialize()
{
    movingAverage = Indicators.SimpleMovingAverage(Source, Periods);
}

public override void Calculate(int index)
{
    Result[index] = Source[index] - movingAverage.Result[index - Periods / 2 - 1];
}
```

Как вы можете видеть, мы написали основной код для вычислений, но не определили никаких ограничений индикатора. Например, если мы используем среднюю скользящую с периодом 10, первые 9 значений не смогут быть точно просчитаны, и мы не сможем использовать эту формулу для параметра index менее 9. Более того, чтобы просчитать значение данного показателя, мы вводим значение средней скользящей `index - Periods / 2 –1` .В общем, нам следует добавить следующие условия:

```
public override void Calculate(int index)
{
    if (index >= Periods + Periods / 2 + 1)
    {
        Result[index] = Source[index] - movingAverage.Result[index - Periods / 2 - 1];
    }
}
```

Код простой средней скользящей также должен быть проверен параметром `index >= Periods` . Более того, если мы будем использовать другой вложенный индикатор, например, другой тип скользящей средней, данные условия должны быть другими. Исходные значения также могут быть результатом вычислений другого индикатора и состоять из не просчитанных значений, для начала. Все эти вещи делают процесс создания индикаторов более сложным, так как нам необходимо думать об автоматически настраиваемых параметрах индикаторов.
Для решения этой проблемы, можно использовать NaN-арифметику. Дело в том, что если значения индикатора не указаны, есть некоторые специальные значения, "Not a number" или NaN. Если мы используем эти значения в вычислениях, мы всегда получаем результат в виде NaN. Если мы используем не вычисленные значения или отрицательные значения серий данных, мы видим их как NaN.
Плавающие цифры с типом `double` уже имеют данное значение: `double.NaN` , и необходимые параметры счета. Выполняя операции с NaN и обычными цифрами, вы всегда получаете NaN, т.е., `double.NaN + 1.4 == double.NaN` .Если вы запрашиваете негативный показатель, `DataSeries` выводит NaN, поэтому вы можете опустить все данные предварительные условия и ввести только основное правило счета индикатора. Если полученный результат будет содержать NaN в некоторых показателях, эти значения не будут наложены на график.

### Рекурсивные индикаторы

Если вы подсчитываете рекурсивный индикатор, т.е., индикатор, значение которого зависит от предыдущего значения этого индикатора, вы должны ввести значение NaN «explicitly», как в примере ниже – для подсчета экспоненциальной средней скользящей:

```
public override void Calculate(int index)
{
    var previousValue = Result[index - 1];

    if (double.IsNaN(previousValue))
    {
        Result[index] = Source[index];
    }
    else
    {
        Result[index] = Source[index] * _exp + previousValue * (1 - _exp);
    }
}
```

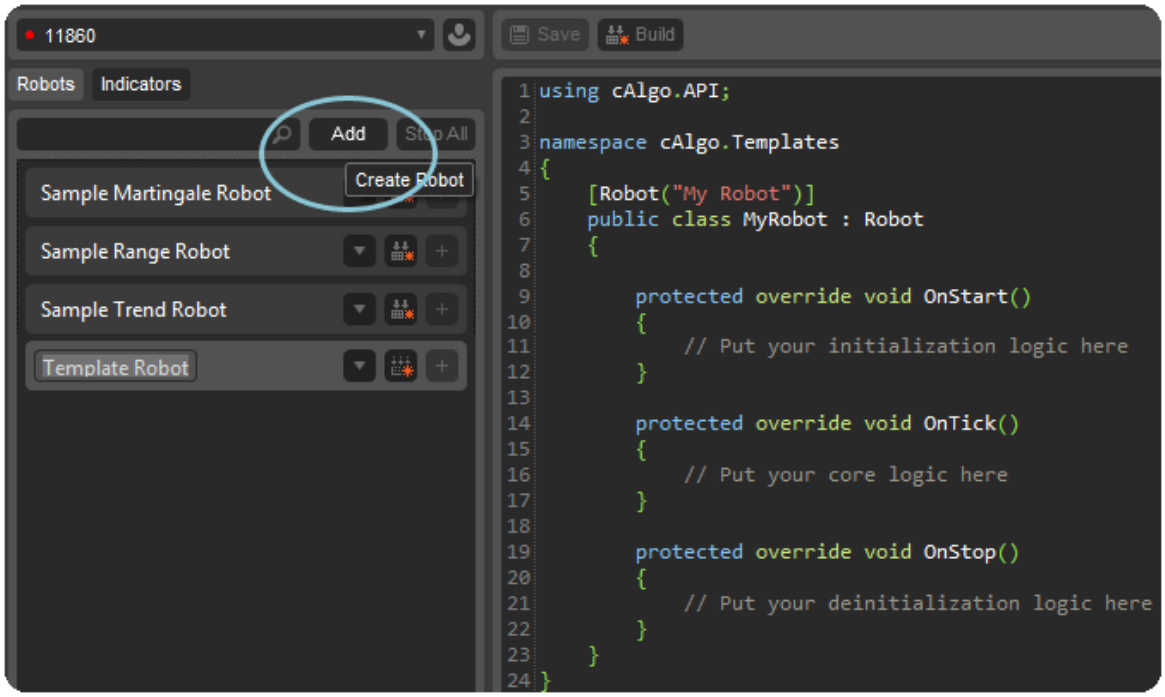
В коде экспоненциальной средней скользящей, где текущее значение зависит от предыдущего, первым значением должно быть установлено «explicitly». Для проверки этого, мы используем метод `double.IsNaN(previousValue)`. Имейте в виду, что мы не можем использовать == данный оператор `previousValue == double.IsNaN`, потому что NaN == x всегда false, даже если x -это NaN.

# Уроки работы с cAlgo

## Создание торгового робота

### Создать нового робота

1. Нажмите на **Add** для создания нового робота и открытия редактора кода:




   Tip: Вы можете переименовать вашего робота, введя новое имя сразу после его добавления, выбрав **Rename** из всплывающего меню робота или нажав клавишу **F2**.

2. **Введите ваш код.**

   Второй и самый важный шаг в создании робота – добавление его алгоритма (или кода). Для этого, вы можете использовать редактор программы cAlgo или скопировать ваш код из другого ресурса и вставить в редактор.

   Вы можете периодически сохранять ваш код, нажав Save или выбрав Save из всплывающего меню.

   Для большей информации по кодированию cAlgo с C#:

   C# Resource

   C# Beginners Guide

3. **Создание роботов**

   Перед тем, как запустить ваш алгоритм для торговли, необходимо конвертировать его в рабочего робота.

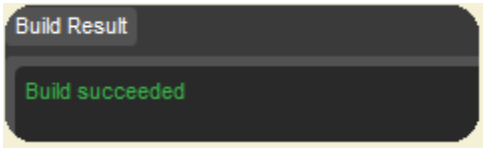
   Для создания робота, нажмите иконку build  .

   **Иконка Build**

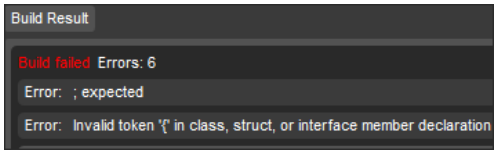
**Означает, что в алгоритм были внесены изменения с последнего build. Перед названием робота также появится звездочка*.**

**Означает, что не было никаких изменений в алгоритме, с последнего build.**

Область **Build Result** снизу платформы покажет, был ли build успешен или же были допущены ошибки в коде.



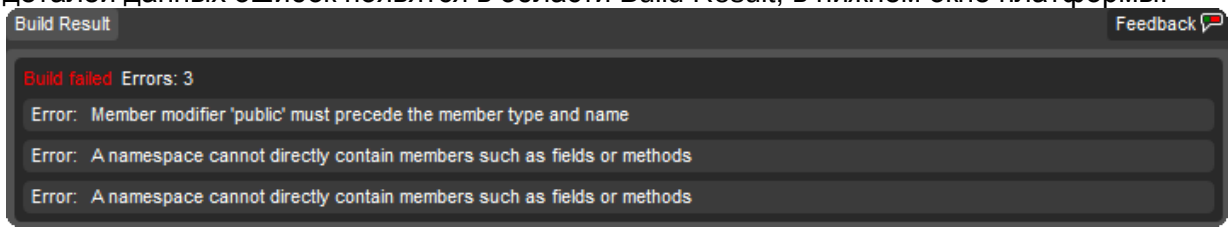

Build успешен




Build не успешен

## Ошибки (Errors)

Робот не может быть создан, если в коде имеются ошибки. Список всех ошибок в коде и деталей данных ошибок появятся в области Build Result, в нижнем окне платформы.




**Совет: Нажатие на строку ошибки** в списке переместит вас в строку кода, в которой была допущена данная ошибка.

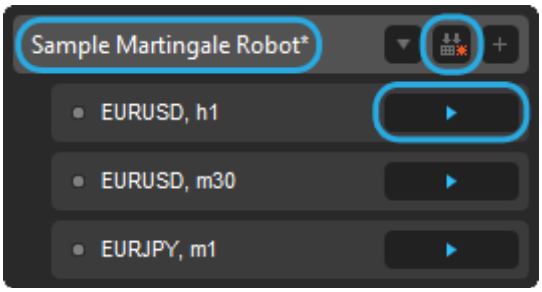
Редактирование

Для редактирования алгоритма для вашего робота, просто выберите робота из списка и внесите изменения в его код, или скопируйте/вставьте код из вашего ресурса. Нажмите Save (сохранить) или Build (создать), для сохранения ваших изменений.

Внимание:>После внесения изменений в код робота, необходимо нажать Build (создать), чтобы робот начал работать в соответствии с внесенными изменениями.

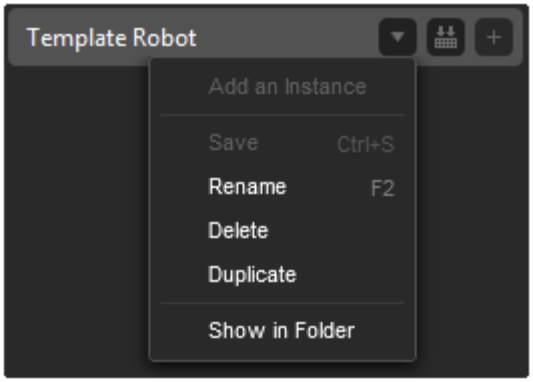
Если исходный код изменен, робот, прикрепленный к графику, будет использовать старый код, пока вы не закрепите изменения с помощью опции Build.

Как показано на примере ниже, звездочка* и иконка build означают, что были сделаны изменения в коде со времени прошлого build'а. Однако существует три уже прикрепленных параметра роботов, которые могут использовать старый код после запуска робота (нажатия play).



Робот не может быть создан, если он в данный момент он активирован для торговли.


**Нажмите стрелочку для выбора свойств роботов:**




**Добавить робота на график (Add an Instance)**
Прикрепляет вашего робота или индикатор к графику, с возможностью изменения его параметров. Изменить можно только параметры, обозначенные в коде робота.

**Созранить (Save)**
Сохранить все изменения, сделанные в коде робота **(Ctrl + S)**.

**Переименовать (Rename)**
Нажмите, чтобы переименовать робота. Это не повлияет на текущую торговлю данного робота. Горячая клавиша: F2

**Удалить (Delete)**
Удалить робота.

**Создать копию (Duplicate)**

Создать копию данного робота.

Примеры использования:

Тестирование множественных ‘версий’ робота.

Изменение кода работающего робота.

Хотя невозможно изменить сам код торгующего робота, прикрепленного к графику, вы можете создать копию данного робота, пока он активен и работать над данной копией кода.

## Показать в папке (Show in folder)

Открыть папку с кодом робота.

## Торговля

Если у вас есть работающий алгоритм без ошибок, вы можете немедленно использовать его для торговли. Робот запустится с новым тиком графика выбранного торгового инструмента. Робот может открывать ордера, закрывать ордера и устанавливать уровни take profit и stop loss.

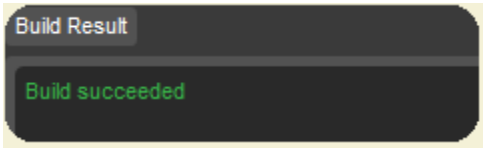
Все шаги по установке торгового робота:

1. **Создание (Build)**

   Нажмите на иконку 

   Это переведет ваш код в рабочего робота.

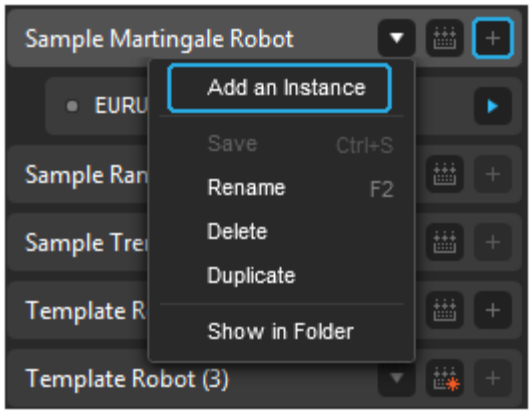
   Если в коде не содержится ошибок, вы увидите сообщение «Build succeeded» в списке результатов создания робота «Build Result», внизу платформы . Далее вы можете добавить робота на график (**add an instance**).




2. **Добавление робота на график (Add an instance)**

   Для использования вашего торгового робота, для начала вам необходимо добавить его на график (**add an instance)**.

   Добавляя робота или индикатор на график, вы выбираете валютную пару для торговли, параметры которой, такие как цены открытия (open), закрытия (close), максимума (high), минимума (low) и тика (tick), сформируют информационный базис для торговли робота. Добавив робота на график, вы также сможете изменять любые его изменяемые параметры, учтенные в коде, например, периоды скользящей средней или уровень отклонения Bollinger band.



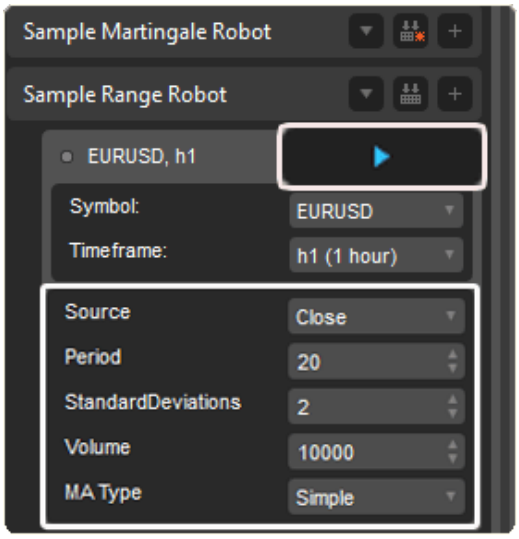

   Выберите **Add an instance** из данного меню или нажмите значок **+** напротив робота, которого вы хотите запустить.

3. **Изменение параметров**

   Код робота позволит пользователям самостоятельно изменять и уточнять его торговые параметры через специальную форму.

   Например: временные периоды (Period), которые будут использованы для

отображения средней скользящей, применение к параметрам графика (Source), и т.д..




После установки/изменения параметров, вы можете продолжить запустить робота.

4. **Запустить робота (Start Robot)**

   Нажмите кнопку  для запуска робота

   Если вы используете данного робота первый раз, появится диалоговое окно, предупреждающее, что ваш робот активирован и может совершать торговые операции, основанные на его коде и параметрах.

   Робот будет выполнять свои функции, после запуска, основываясь на тиках котировок. Большая часть кода и данных для работы робота, рассчитываются на один тик.
   Когда данные рынка соответствуют триггеру открытия торговых позиций роботом, прописанных в его коде, ордер будет исполнен.

   После открытия роботом торговой позиции, появится окно подтверждения в правом верхнем углу платформы cAlgo.

Все открытые торговые позиции будут отображены в разделе торговли.

### Данные раздела торговли

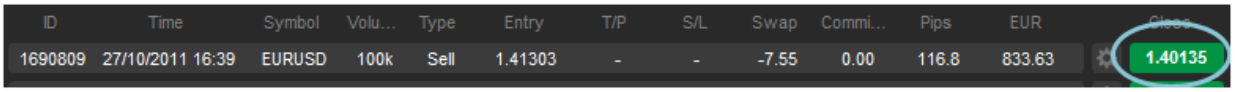
- **ID -** Уникальный ticket-номер ордера.
- **Time** - Дата и время размещения ордера. Отображается как DD/MM/YYYY HH:MM (День/Месяц/Год Час:Минута).
- **Symbol** - Символ валютной пары.
- **Volume** - Общий объем сделки.
- **Type** - Тип ордера: **Buy (Покупка)** или **Sell (Продажа)**.
- **Entry** - Цена, по которой ордер был установлен.
- **T/P** - Уровень **Take Profit**. Если он не был выставлен, в этом поле будет отображаться знак ( - ).
- **S/L** - Уровень **Stop Loss**. Если он не был выставлен, в этом поле будет отображаться знак ( - ).
- **Swap** - Общий размер **swap**.
- **Commission** - Общая комиссия, взимаемая брокером за даннуб торговую операцию.
- **Pips** - Прибыль или убыток, в пунктах.
- **Base Currency P&L** - Прибыль или убыток, в базовой валюте.

- **Close** - Отображает текущую рыночную котировку. Нажмите **close**, чтобы закрыть сделку.

### Закрытие торговой позиции

Открытая позиция  будет автоматически закрыта, когда цена достигнет уровня **Stop Loss** или **Take Profit**.

Для того, чтобы закрыть позицию вручную, нажмите на котировку, в колонке **Close (Закрыть)**, или нажмите **правой кнопкой** на строку ордера и выберите **Close Position (Закрыть позицию)**.




### Частичное закрытие позиций

Вы можете закрыть часть открытой позиции, с минимальным шагом в 10.000 (10k) единиц базовой валюты. Для этого:

1. Нажмите правой кнопкой мыши на строку торговой позиции
2. Выберите Modify Position (Изменить позицию)
3. Выберите сумму для закрытия и нажмите Close (Закрыть).

### Изменение уровней Stop Loss и Take Profit

Уровни Take Profit и Stop Loss могут быть изменены во время торговли. Для изменения уровня Stop Loss, нажмите правой кнопкой мыши на торговой позиции и выберите **Modify Position (Изменить позицию)**.

# Поддержка

## cTDN

**cTrader Developers Network** – это интернет-ресурс разработчиков программы cAlgo, где вы можете найти форумы, блоги и ресурсы по работе с кодом. cTDN создан для того, чтобы пользователи в профессиональной атмосфере, могли делиться знаниями, навыками и материалами о пользовательских роботах и индикаторах.

**Нажмите на эту ссылку для того, чтобы перейти вTrader Developers Network**

## Контакты

**Торговые счета**
Свяжитесь с вашим брокером

**cTrader Email**
info@ctrader.com

**Телефон для связи**
+44 (0) 207 743 8801

## Обратная связь


Следуя отзывам и пожеланиям, поступающим от трейдеров, мы улучшаем работу наших программных продуктов. Пожалуйста, сообщите нам Ваши потребности и пожелания в торговле, Ваш опыт работы с нами и Ваши ожидания от дальнейшего развития торговых программ **cAlgo** и **cTrader**. Спасибо Вам.

**Нажмите на кнопку Отзыв (Feedback), расположенной справа внизу торговой платформы, чтобы отправить нам свой отзыв.**

# Интерфейс программирования приложений: роботы (cAlgo API)

## Введение

cAlgo – это новейшая программа для автоматической торговли. Она позволяет вам создавать роботов, в виде алгоритмов, удовлетворяющих вашей личной торговой стратегии. В дополнении к роботам, на платформе cAlgo вы сможете создавать пользовательские индикаторы.

Эта инструкция содержит описание программного ядра функционала cAlgo API. С помощью поиска элементов cAlgo, вы можете уточнить детали и описания необходимых элементов программы.

## Язык программирования

cAlgo использует .NET Framework 4 и C# в качестве основных языков программирования. Чтобы создать робота или индикатор, вы должны использовать язык C# на базе собственного класса API.
Данный пользовательский класс может заменять собой любые иные методы написания алгоритмов. Вы можете использовать любые конструкции и классы языка C#.
Метод создания кода представляет собой серию разделов. Таким образом, на языке C# оформлены и осуществляются все заданные в коде параметры.

Вы можете обратиться к Microsoft C# http://msdn.microsoft.com/en-us/library/67ef8sbd.aspx

# Начало

Для облегчения создания торгового робота, в cAlgo имеются шаблоны различных классов роботов:

```
using System;
using cAlgo.API;
using cAlgo.API.Indicators;

namespace cAlgo.Robots
{
    [Robot "New Robot")]
    public class NewRobot : Robot
    {
        protected override void OnStart()
        {
            // Put your initialization logic here
        }

        protected override void OnTick()
        {
            // Put your core logic here
        }

        protected override void OnStop()
        {
            // Put your deinitialization logic here
        }
    }
}
```

`NewRobot` - это класс, произошедший от базового класса – `Robot`. Чтобы использовать этот класс как робот, мы должны его маркировать атрибутом `[Robot]`. Он уже применен к роботу в шаблоне.

Теперь вы можете переименовать этот класс. Давайте назовем его `MyFirstRobot` и изменим его имя:

```
[Robot("First Robot!")]
public class MyFirstRobot : Robot
```

Жизненный цикл робота

Базовый класс робота имеет ряд виртуальных методов которые можно отнести к вашему роботу. В шаблоне робота «по умолчанию» вы можете видеть 3 из них:

- `OnStart()` - это обработчик событий, к которому робот обращается во время своего запуска (когда робот активирован на графике). Здесь вы можете добавить логику запуска робота, например, создать индикатор или исполнить ордер, которая должна быть исполнена при старте .
- `OnTick()` запускается при каждом тике, проверяет сигналы от индикаторов, исполняет, модифицирует или закрывает ордера и т.д..
- `OnStop()` запускается, если робот остановлен. Вы можете добавить сюда логику деинсталляции робота.

Вы также можете вручную установить несколько торговых методов, переняв их из шаблонов, для обработки торговых событий.

## Торговля

Исполнение торговых операций – это жизненно важная функция любого робота. Роботы могут открывать рыночные и отложенные ордера, устанавливать уровни Stop Loss / Take Profit, модифицировать и закрывать ордера.

### Как открыть торговую позицию

Для открытия торговой позиции, используется оператор `Trade`. Он содержит несколько методов отправки запросов на сервер, для открытия торговых позиций.

Например, чтобы робот создал и выполнил рыночный ордер на покупку100k единиц выбранного торгового инструмента, необходимо задать следующие параметры:

```
protected override void OnStart()
{
   Trade.CreateBuyMarketOrder(Symbol, 100000);
}
```

В данном примере мы используем объект `Trade` и его метод `CreateBuyMarketOrder`, для создания и отправки рыночного ордера на сервер. Этот метод использует два параметра: `symbol` (символ) и `volume` (объем). Первым из них, мы используем `Symbol` - символ торгового инструмента, к которому прикреплен робот. Объем постоянен (в примере выше он равен 100k). В результате, новый рыночный ордер будет создан и отправлен на сервер.

Для создания различных ордеров, вы можете использовать следующие методы:

- CreateBuyMarketOrder(symbol, volume)
- CreateSellMarketOrder(symbol, volume)
- CreateMarketOrder(tradeType, symbolCode, volume)

- CreateBuyLimitOrder(symbol, volume, targetPrice)
- CreateSellLimitOrder(symbol, volume, targetPrice)
- CreateBuyStopOrder(symbol, volume, targetPrice)
- CreateSellStopOrder(symbol, volume, targetPrice)

Как вы можете видеть, имеются разные методы для разных видов ордеров. У этих методов могут быть различные параметры, например, если вы хотите создать лимитный ордер (limit order), цена открытия (target price) должна быть задана заранее.

Открыв позицию, вы можете хранить ее в разделе переменных, чтобы иметь возможность ее закрыть или выполнять другие операции с ней в будущем:

```
private Position myPosition;

protected override void OnStart()
{
    Trade.CreateBuyMarketOrder(Symbol, 100000);
}

protected override void OnPositionOpened(Position openedPosition)
{
    myPosition = openedPosition;
    Print("Position {0} is opened, entry price is {1}", myPosition.Id  myPosition.EntryPrice);
}
```

Здесь мы работаем с методом `OnPositionOpened` для получения кода новых открытых позиций, затем мы сохраняем это и печатаем сообщение в log робота.

### Как закрыть торговую позицию

В примере ниже, я закрываю позицию, хранящуюся в `myPosition`, когда цена достигает отметки выше установленного уровня:

```
private Position myPosition;

protected override void OnTick()
{
    if (myPosition != null && Symbol.Bid > 1.3808)
    {
        Trade.Close(myPosition)
    }
}
```

Метод `OnTick()` активируется при появлении каждого нового тика, проверяет уровень bid цены и, если он выше установленного уровня, позиция закрывается. Также проверяется, имеется ли для данной позиции условие `position != null`, и если торговой позиции нет, закрыть ее невозможно. Для того, чтобы это сделать, мы используем оператор `Close(position)`, который направляет запрос на закрытие позиции на сервер.

### Как установить уровни Stop Loss / Take Profit

Уровни stop loss и take profit могут быть установлены для существующих торговых позиций, мы можем их только модифицировать, пока они открыты. Для этого, мы используем метод *OnPositionOpened*. В примере ниже, установлены Stop Loss и Take Profit на 30 пунктов от цены открытия нового ордера:

```
protected override void OnPositionOpened(Position openedPosition)
{
    double newStopLoss = position.EntryPrice - 30 * Symbol.PipSize;
    double newTakeProfit = position.EntryPrice + 30 * Symbol.PipSize;
    Trade.ModifyPosition(position, newStopLossPrice, newTakeProfit);
}
```

Если вы не хотите устанавливать уровни Stop Loss и Take Profit для ваших торговых позиций, необходимо также вписать *null*. В примере ниже уровень Take Profit отмечен без установки значения Take Profit:

```
Trade.ModifyPosition(position, null, newTakeProfit);
```

Если вы хотите установить новый уровень Stop Loss, но оставить Take Profit таким, каким он был ранее, то мы можем взять значение старого уровня Take Profit:

```
Trade.ModifyPosition(position, newStopLoss, position.TakeProfit);
```

## Рабочие параметры

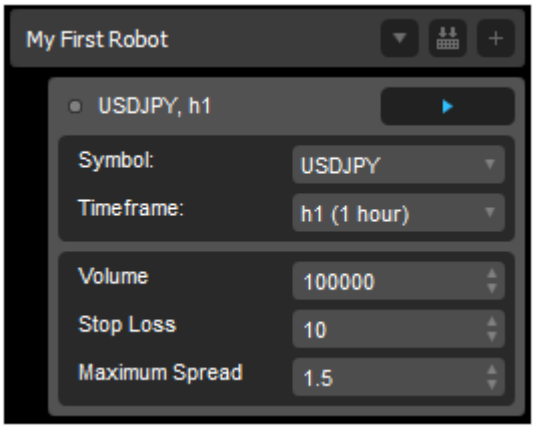
Обычно роботы используют внешние параметры для своей торговли. В последующем примере робот принимает торговые параметры, такие как объем `Volume`, `StopLoss` и максимальный спред `MaxSpread`. Данный алгоритм откроет рыночный ордер заданного объема и с заданным уровнем Stop Loss, пока текущий спред на рынке ниже заданного максимума.

```
[Robot("First Robot")]
public class FirstRobot : Robot
{
    [Parameter("Volume", DefaultValue = 100000, MinValue = 10000, MaxValue = 10000000)]
    public int Volume { get; set; }

    [Parameter("Stop Loss (pips)", DefaultValue = 10, MinValue = 1)]
    public int StopLoss { get; set; }

    [Parameter("Maximum Spread", DefaultValue = 1.5, MinValue = 0.1)]
    public double MaxSpread { get; set; }

    protected override void OnStart()
    {
        if (Symbol.Spread < MaxSpread)
          {
             Trade.CreateBuyMarketOrder(Symbol.Code, Volume);
           Trade.Execute(order);
        }
    }

    protected override void OnPositionOpened(Position openedPosition)
    {
        double? stopLossPrice = position.EntryPrice - StopLoss * Symbol.PipSize;
        Trade ModifyPosition(position, stopLossPrice, null);
```

Имеется несколько поддерживаемых типов данных для заданных параметров:

- int – Целое значение, т.е. дистанция в пунктах, объем торговой позиции.
- Double – Плавающий номер, т.е. цена, значение индикатора
- DataSeries – Ценовые данные рынка. Пользователь может задать, значения каких параметров советнику использовать: цену открытия (Open), максимума (High), минимума (Low) и закрытия (Close).
- MovingAverageType – Параметры средней скользящей (moving average), которая может быть встроена в индикаторы (см. раздел индикаторов).

Добавив выбранного робота на график, вы сможете вручную изменять его параметры:




Значения символа (Symbol) и временного интервала (Timeframe) являются общими параметрами, после добавления робота на график. Эти параметры указаны в левом меню для каждого робота.

## Использование индикаторов

Большинство торговых стратегий основаны на анализе данных технических индикаторов. Вы имеете доступ ко встроенным индикаторам из вашего робота или пользовательских индикаторов. С каждым тиком, данные этих индикаторов пересчитываются торговой платформой, перед тем, как робот получает сигналы.

Если ваш робот использует индикатор, он должен быть создан методом OnStart(). Для этого используйте объект Indicators, включающий в себя все методы для встроенных индикаторов:

```
Indicators.BollingerBands(source, periods, deviations, maType)
```

В следующем примере робот создает два индикатора Simple Moving Average. С помощью метода OnTick() робот совмещает последние значения каждого индикатора. (значения сортируются в хронологическом порядке, так чтобы значения последнего индикатора согласовывались с последним баром текущего таймфрейма). Ордер будет открыт в том случае, если значение более быстрой средней скользящей (moving average) с меньшим периодом больше последнего значения медленной средней скользящей.

```
private MovingAverage slowMa;
private MovingAverage fastMa;

protected override void OnStart()
{
    fastMa = Indicators.SimpleMovingAverage(MarketSeries.Close, 15);
    slowMa = Indicators.SimpleMovingAverage(MarketSeries.Close, 40);

protected override void OnTick()
{
    if (fastMa.Result.LastValue > slowMa.Result.LastValue)
    {
        Trade.CreateBuyMarketOrder(Symbol, 10000);
    }
}
```

## Свойства символов

Каждый робот прикреплен к заданному графику, символу и таймфрейму. Для лоступа к значениям символа (торгового инструмента), необходимо использовать объект Symbol. Объект Symbol имеет ряд зависимых от него параметров:

- Ask – Текущая цена ask
- Bid – Текущая цена bid
- Spread – Текущий спред, разница между ценами Ask и Bid
- Code – код символа, например “USDJPY”
- PipSize – размер одного пункта, например 0.0001, если котировки четырехзначные
- PointSize – размер одного тика, например 0.00001 для 5-значных котировок
- Digits – Количество цифр после точки, 4 или 5

Следующий код создаст ордер Sell Limit для выбранного торгового инструмента, с целью цены 100 пунктов выше текущей цены bid.

```
double targetPrice = Symbol.Bid + Symbol.PipSize * 100;
Trade.CreateSellLimitOrder(Symbol, 200000, targetPrice);
```

## Информация по счету

Если торговому роботу необходимо иметь доступ к текущим настройкам торгового счета, необходимо использовать объект Account. Он включает в себя основную информацию по счету и текущее его состояние:

- Balance – текущий баланс счета
- Currency – валюта счета, например, “EUR”
- Equity – текущий размер свободных средств (баланс плюс текущая прибыль/убыток)
- Margin – текущая маржа
- FreeMargin – текущая свободная маржа
- MarginLevel – уровень маржи

В примере ниже  ордер будет установлен только в том случае, если маржа меньше, чем половина размера свободных средств:

```
if (Account.Margin > Account.Equity / 2)
{
    // Continue trading
```

## Скопировать в log робота

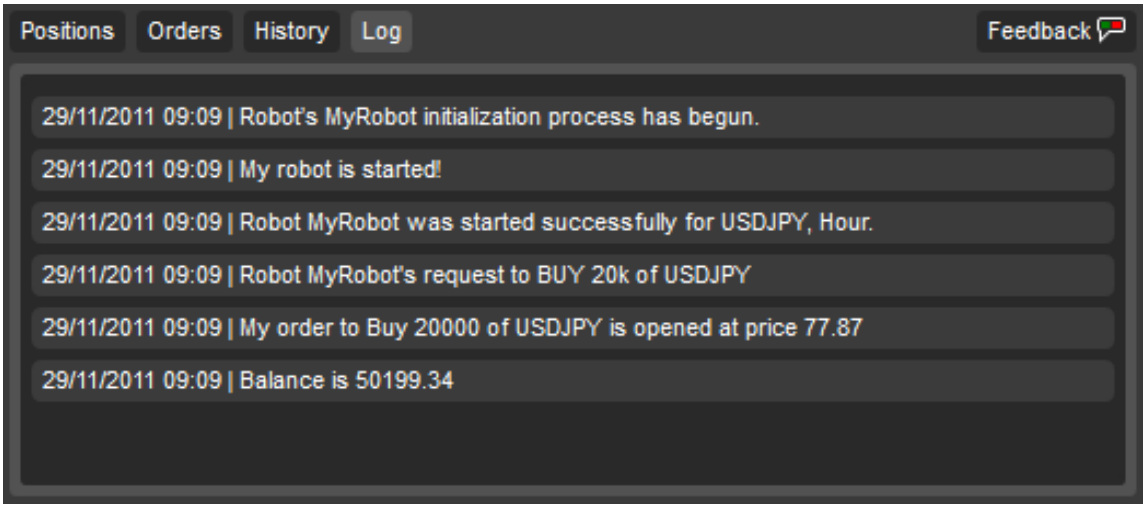
Для копирования сообщений в log робота используется метод Print(). Например, если нужно скопировать в log робота информацию, когда он запущен и позиции открыты, можно добавить сообщения, как указаны ниже:

- My order to Buy 20000 of EURUSD is opened at price 1.38792
- Balance is 1069.29

И также добавить данные сообщения в текущий код:

```
protected override void OnStart()
{
    Print("My robot is started!");
}

protected override void OnPositionOpened(Position position)
{
    Print("My order to {0} {1} of {2} is opened at price {3}",
        position.TradeType, position.Volume, position.SymbolCode, position.EntryPrice);

    Print("Balance is {0}", Account.Balance ;
}
```

Вы можете выбрать формат сообщения из стандартных форматов .NET format, с параметрами заполнения и значениями из списка параметров. В результате, log робота будет выглядеть следующим образом:




Вы также можете увидеть некоторые системные сообщения, которые автоматически печатаются в log.

# FAQ

**Могу ли я одновременно подключиться к нескольким счетам?**

Да. Хотя Вы не можете подключиться к нескольким счетам в одном приложении cAlgo, но вы можете запустить несколько приложений cAlgo (отдельно запущенные платформы cAlgo на Вашем компьютере в один момент времени) и в каждом подключиться к различным счетам.

**Могу ли я подключаться к одному счету с разных компьютеров?**

Да. При использовании корректных данных торгового счета Вы можете подключаться к нему с разных компьютеров, если верны настройки подключения и соединение не блокируется файерволом или Интернет-провайдером.

**Как долго работает демо-счет cAlgo / cTrader?**

Срок действия демо-счета cAlgo не ограничен.

**Могу ли я изменить часовой пояс в торговой платформе?**

Серверное время установлено брокером и не может быть изменено Вами. Если у Вас есть вопросы относительно Вашего часового пояса, Вы можете уточнить его у Вашего брокера.

**Откуда поступают котировки?**

Каждый брокер имеет поставщиков ликвидности (Barclays, UBS, Reuters и др.), которые поставляют котировки в режиме реального времени.

**Где я могу посмотреть историю моей торговли?**

Для просмотра истории торговли нажмите вкладку “History” в окне ‘Trade Watch’ в нижней части платформы. Также Вы можете сохранить история Вашей торговли как отчет, нажав кнопку ‘Statement’.

**Могу ли я использовать cAlgo для размещения отложенных ордеров?**

Да. Функция добавления limit или stop ордеров, основанных на технических индикаторах, могут быть вписаны в исходный код.

**Что случится с моей позицией, когда я выключу cAlgo?**

Открытая позиция останется активной и не будет закрыта до достижения установленного уровня Stop Loss, Take Profit или Stop out.

Для того чтобы робот выполнял все записанные в него функции (открытие и закрытие позиций, изменения уровня Stop Loss и др.) cAlgo требуется постоянное интернет-подключение. Таким образом, открытые позиции не будут закрыты, если Вы отключите

торговый счет или закроете торговую платформу, а роботы не смогут изменить или закрыть любую действующую позицию.

Существующие limit ордера все равно будут исполнены, если текущая цена достигнет установленного значения.

**Могу ли я открыть реальный счет?**

Да. Пожалуйста, свяжитесь со службой поддержки одного из наших брокеров.

**Могу ли я с помощью cAlgo создать собственные технические индикаторы?**

Да. cAlgo позволяет Вам создавать ‘Роботов’ и собственные Индикаторы. Ваш собственный индикатор может быть использован как в cAlgo, так и в cTrader.

**Может ли робот ссылаться на пользовательские индикаторы?**

Да. Ссылки на технические индикаторы могут быть встроены в алгоритмы Вашего Робота.

**Сколько алгоритмов я могу добавлять и запускать в одно время?**

Нет ограничений по количеству запусков алгоритмов.

**Я все еще могу торговать в cAlgo вручную?**

Да, cAlgo позволяет вручную закрывать, открывать и изменять открытые и отложенные ордера на графиках. cTrader – наша ручная торговая платформа, к которой Вы можете быстро подключиться нажав кнопку cTrader в правом верхнем углу приложения cAlgo..

**Обязательно ли запускать cTrader для работы в cAlgo?**

Нет. Обе платформы могут работать одновременно независимо друг от друга.

**Какой язык используется при написании Роботов и Пользовательских Индикаторов в cAlgo?**

Ваши роботы и индикаторы должны быть написаны на языке программирования C# .NET.

# Начало работы

## cAlgo

cAlgo – это новая платформа для автоматической торговли. Работая вместе с **cAlgo** и **cTrader**, пользователи получают доступ ко всем важным функциям, необходимым для технического анализа и автоматической торговли.

Используя редактор исходного кода cAlgo, пользователи могут создавать алгоритмы на языке программирования C#, и начать из сразу использовать для реализации торговых стратегий на платформе.

Помимо Роботов, в cAlgo можно создавать пользовательские индикаторы для технического анализа.

## Алгоритмическая торговля

Алгоритмическая торговля – это форма торговли на финансовых рынках, когда ордера на покупку или продажу открываются (размещаются) автоматически на основе алгоритма. Робот будет исполняться и закрывать ордера, основываясь на таких факторах, как текущая цена, время, историческое движение цены и значения индикатора.

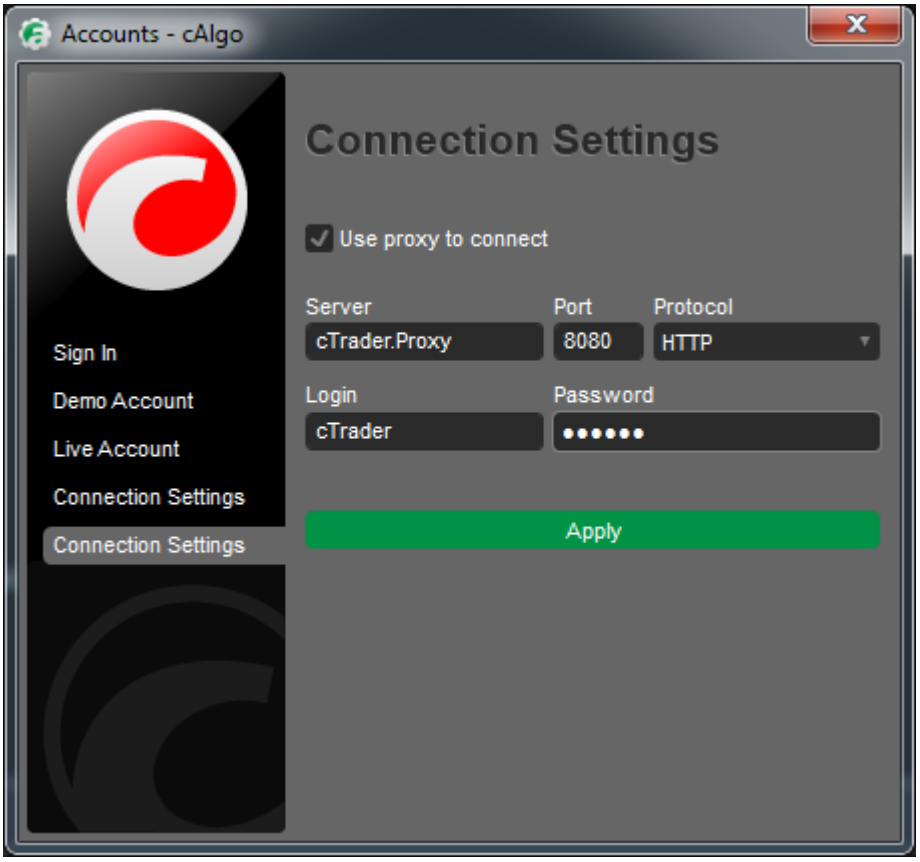
Пользовательский технический индикатор, созданный в cAlgo, будет также доступен в cTrader в списке индикаторов. На все индикаторы можно ссылаться в алгоритмах ваших **Пользовательских Индикаторов** и **Роботов**.

# Настройки подключения

В данном разделе вы можете вручную установить настройки прокси. Ручная настройка может быть необходима при подключении через некоторые офисные сети или провайдеры.

Для установления соединения вам необходимо указать следующую информацию:

- **Server** – Имя/IP вашего локального прокси сервера
- **Port** – Порт вашего локального прокси сервера
- **Protocol** – Поддерживаемый протокол прокси
- **Login** – Логин вашего локального прокси
- **Password** – Пароль вашего локального прокси




*Если вы не знаете настройки Вашего прокси. Обратитесь к интернет провайдеру или системному администратору.*

## cAlgo: автоматическое обновление

В платформу cAlgo встроено автоматическое обновление. При каждом запуске cAlgo все доступные обновления будут автоматически загружены и установлены.

Процесс обновления может занять несколько секунд, после чего cAlgo запустится со всеми успешно установленными обновлениями.

## cAlgo: установка и запуск

### Установка cAlgo

Для установки cAlgo необходимо выполнить следующие действия:

1. Скачать платформу.
2. Нажать **Run** (**Выполнить**) для начала установки.
3. Следуйте инструкциям на экране.

В процессе установки Вам может быть предложено обновить / установить .Net4. Необходимо согласиться.

Также может потребоваться перезагрузка компьютера. Установка cAlgo будет завершена после перезагрузки.

### Запуск cAlgo

cAlgo будет автоматически запущена после полной установки. Вы можете запустить cAlgo вручную, нажав на соответствующий значок на рабочем столе и в меню Пуск Вашего компьютера.

## Демо-счета

**Демо-счет** предоставляет безрисковые условия торговли для практики. Функционал и рыночные котировки соответствуют реальным торговым счетам, но использоваться будут ‘виртуальные’ средства. Демо-счет совершенно бесплатен и вы не можете выводить любые виртуальные средства полученные в ходе торговли.

Создать счет
Войти
Удалить счет

### Создание Демо-счета

1. Нажмите **cAlgo** > **Open Demo Account (Открыть демо-счет)** ИЛИ нажмите  и выберите **Demo Account (Демо-счет)**.

2. Заполните персональные данные.
   **Name**: Ваше полное имя
   **Email**: Регистрационные данные вашего счет будут отправлены на этот адрес
   **Country**: Страна проживания
   **Deposit**: Сумма депозита (виртуальные средства).
   **Currency**: Валюта ваших виртуальных средств. Используйте ту же валюту, какую планируете использовать на реальном счете.
   **Leverage**: Максимальное кредитное плечо 1:100.

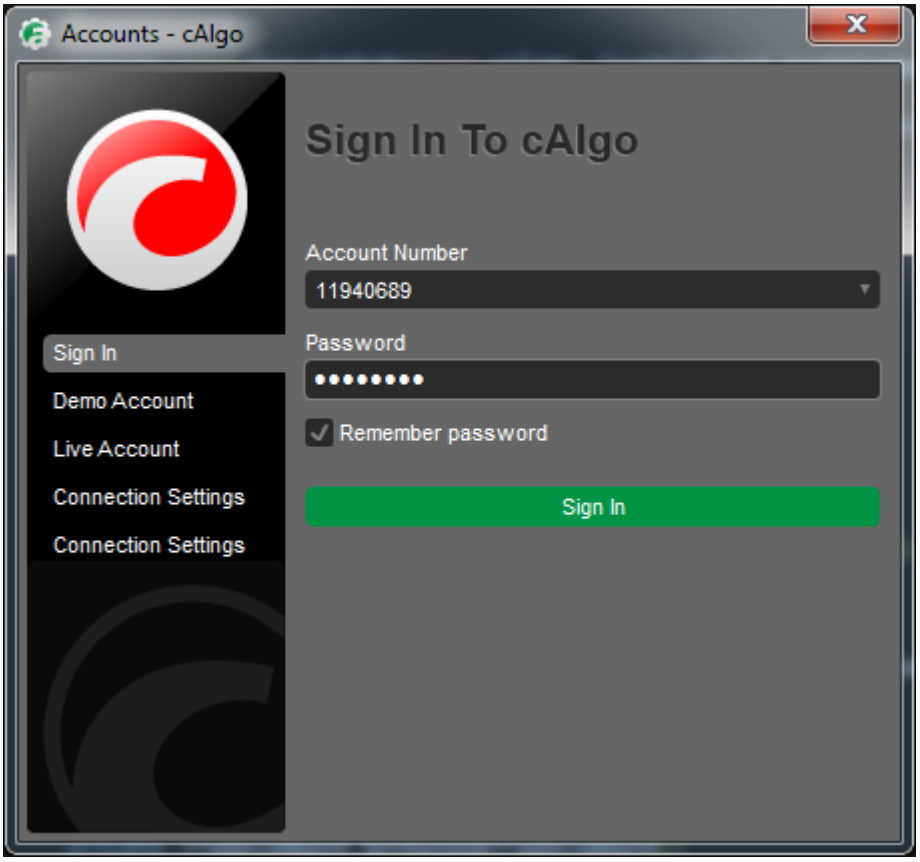
3. После ввода всех данных нажмите **I Agree - Create Account (Я согласен – Открыть счет)**.

Регистрационные данные Вашего счета будут отправлены по указанному Вами адресу электронной почты.

### Войти

После запуска cAlgo появится окно подключения. Если вы не видите этого окна, нажмите cAlgo > Login (Логин).

Для подключения к демо-счету введите **Номер счета** и **Пароль**, затем нажмите кнопку **Войти**.

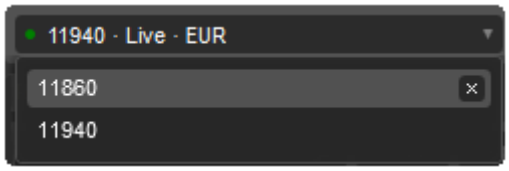



**Remember Password (Запомнить пароль)**
Отмечая галочкой данную опцию, вы сохраняете данные вашего торгового счета на платформе cAlgo, и каждый раз при открытии платформы, подключение к вашему торговому счету будет происходить автоматически. Если у вас несколько счетов, cAlgo сохранит данные последнего счета, для которого вы применили опцию запоминания пароля.

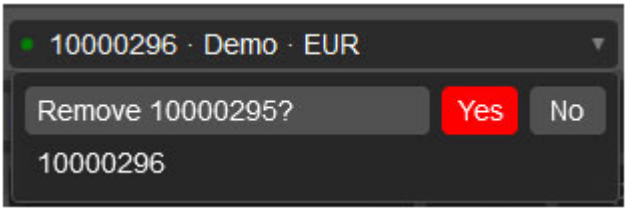
## Удаление счетов

Для того, чтобы удалить счет из списка счетов торговой платформы cTrader необходимо:

1. Подвести курсор мышки к номеру торгового счета, как это указано на картинке снизу, и нажать **X**.




2. Подтвердить удаление счета из списка нажатием **Yes (Да)**.




После подтверждения удаления, счет будет удален из списка счетов платформы.

# Торговые счета

## Create a Live Account

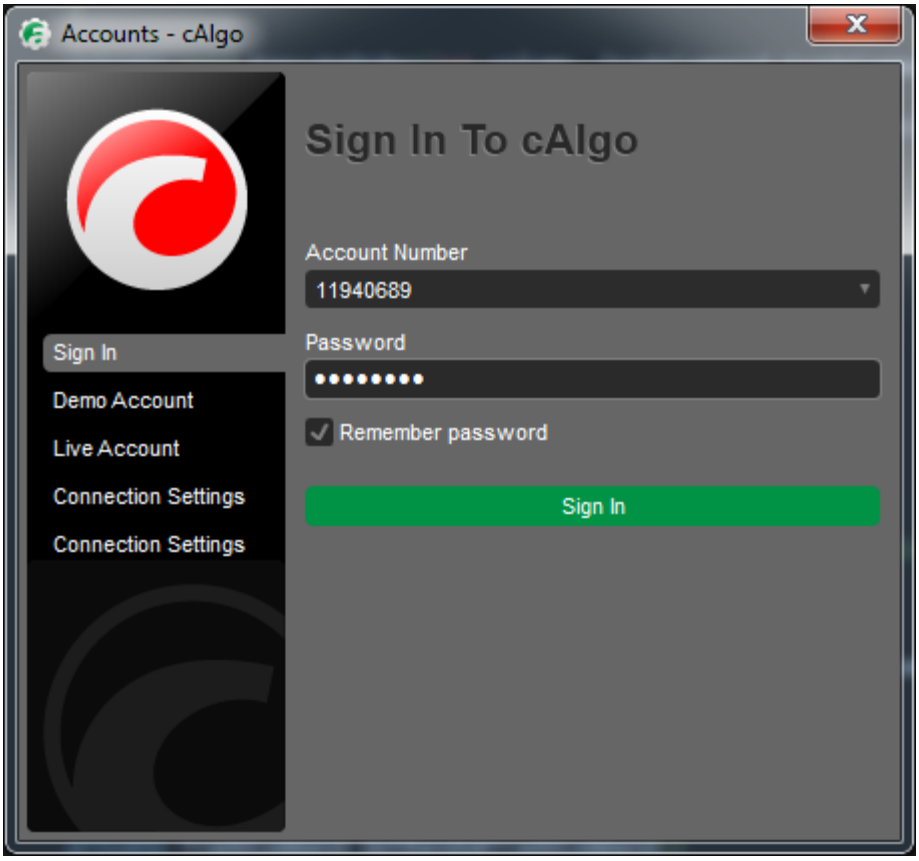
## Создать реальный торговый счет

Для того, чтобы открыть реальный торговый счет на платформе cAlgo, пожалуйста, свяжитесь с вашим брокером.

## Подключение к торговому счету

Когда вы запускаете cAlgo, появится соответствующий значок в поле окна. Если вы не видите экран подключения к торговому счету, нажмите cAlgo > Login.

Для подключения к вашему демо-счету, введите ваш **Номер счета  (Account Number)** и пароль (**Password)**, затем нажмите на кнопку **Sign In (Войти)**.



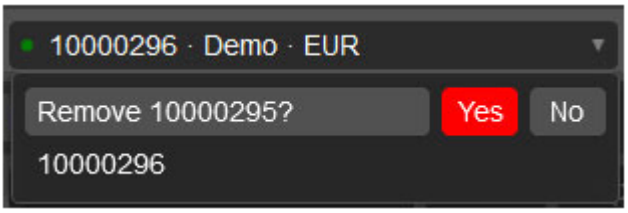

**Remember Password (Запомнить пароль)**
Отмечая галочкой данную опцию, вы сохраняете данные вашего торгового счета на платформе cAlgo, и каждый раз при открытии платформы, подключение к вашему торговому счету будет происходить автоматически. Если у вас несколько счетов, cAlgo сохранит данные последнего счета, для которого вы применили опцию запоминания пароля.

## Удаление счетов

Для того, чтобы удалить счет из списка счетов торговой платформы cTrader необходимо:

1. Подвести курсор мышки к нонеру торгового счета, как это указано на картинке снизу, и нажать **X**.


2. Подтвердить удаление счета из списка нажатием **Yes (Да)**.




После подтверждения удаления, счет будет удален из списка счетов платформы.

**Внимание**: *Удаление счета из списка счетов программы не удалит ваш счет у вашего брокера. Даже после удаления счета из платформы, вы вновь сможете к нему подключиться, введя его данные при подключении к счету. Если вы желаете полностью закрыть ваш торговый счет, обратитесь к вашему брокеру.*