**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ**

Государственное образовательное учреждение высшего профессионального образования

**“Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”**

**им. В. И. Ульянова (Ленина)” (СПбГЭТУ)**

|  |
| --- |
|  |

Факультет КТИ УТВЕРЖДАЮ

Кафедра АСОИУ Зав. кафедрой Советов Б.Я.

Специальность 230102 «…….»………………………………

**ЗАДАНИЕ**

**на дипломное проектирование**

|  |
| --- |
| Студенту Мейстельману Виталию Александровичу |
| Место дипломного проектирования СПбГЭТУ «ЛЭТИ» |
|  |
| 1. Тема дипломного проекта (работы) |
| Разработка системы для автоматической торговли на фондовой бирже |
|  |
|  |
| 2. Назначение разработки |
| Повышение эффективности работы трейдеров на бирже |
|  |
| 3. Перечень подлежащих разработке разделов по теме |
| Определения |
| Введение |
| Анализ предметной области и разработка технического задания |
| Программная реализация разработанной системы |
| Тестирование и подготовка презентации системы |
| Экономическое обоснование |
| Охрана интеллектуальной собственности |
|  |
| 4. Состав технической документации проекта |
| Пояснительная записка к дипломному проекту |
| Презентация |
|  |
| 5. Консультанты ( с указанием относящихся к ним разделов) |
| Консультант по экономическому обоснованию: Швецова О.А. |
| Консультант по охране интеллектуальной собственности: Берковская К.И. |
|  |

Дата выдачи задания Дата сдачи проекта (работы)

«……»………………. «……»……………….

Руководитель………………………………………… Степулёнок Д.О.

Консультант от кафедры…………………………….. Степулёнок Д.О.

Студент……………………………………………….. Мейстельман В.А.

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ**

Государственное образовательное учреждение высшего профессионального образования

**“Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”**

**им. В. И. Ульянова (Ленина)” (СПбГЭТУ)**

|  |
| --- |
|  |

Факультет      КТИ                УТВЕРЖДАЮ

Кафедра    АСОиУ               Зав. кафедрой     Советов Б.Я

Специальность     230102      «…….»………………………………

**КАЛЕНДАРНЫЙ ПЛАН**

**дипломного проектирования**

|  |
| --- |
| Студента **Мейстельмана Виталия Александровича** |
| Тема дипломного проекта (работы) |
| Разработка системы для автоматической торговли на фондовой бирже |
|  |
| Срок дипломного проектирования: 4 месяца |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование раздела | Срок выполнения | Отметка  о выполнении |
| 1.  2.  3.  4.  5. | Анализ предметной области, аналогичных программ и разработка технического задания  Разработка и реализация системы  Тестирование  Технико-экономическое обоснование  Охрана интеллектуальной собственности | 19.10.11  20.12.11  05.01.12  18.01.12  25.01.12 | Выполнено  Выполнено  Выполнено  Выполнено  Выполнено |

Руководитель                                     Степулёнок Д.О.

Консультант от кафедры                  Степулёнок Д.О.

Студент                                               МейстельманВ.А.

**Abstract**

The goal of this diploma is to develop program-robot for automated trading in stock exchange.

Objectives:

* Market Analysis automation of trading strategies.
* To develop of the connector to connect to the trading terminal.
* To develop platform for the trading robots development.
* Implementation user functionality of the system.
* Determine the prospects for the development of the system.

The introduction of automated trading system allows you to: Supervise execution of trading strategies. Automatic historical data analysis on basis technical analysis. Accelerate decision about buying / selling. Evaluate the success of the chosen trading strategy.

Оглавление

[Определения 6](#_Toc317173890)

[Введение 11](#_Toc317173891)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ 13](#_Toc317173892)

[1.1 Обзор устройства российского фондового рынка 13](#_Toc317173893)

[1.2 Обзор технологии интернет-трейдинга. 13](#_Toc317173894)

[1.3 Обзор существующих торговых терминалов. 15](#_Toc317173895)

[1.4 Обзор существующих систем для создания торговых роботов. 17](#_Toc317173896)

[1.5 Техническое задание 20](#_Toc317173897)

[1.6 Виды анализа ситуации на рынке (котировок) 20](#_Toc317173898)

[1.7 Понятие стратегии 21](#_Toc317173899)

[1.8 Средства разработки 22](#_Toc317173900)

[2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ 24](#_Toc317173901)

[2.1 Разработка общей архитектуры системы 24](#_Toc317173902)

[2.2 Выбор способа подключения системы к терминалу 25](#_Toc317173903)

[2.3 Разработка формата описания стратегий поведения робота 25](#_Toc317173904)

[2.4 Последовательность действий при работе с Framework и необходимые навыки разработчика стратегий 25](#_Toc317173905)

[2.5 Разработка коннектора 26](#_Toc317173906)

[3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ 29](#_Toc317173907)

[3.1 Выбор используемых программных средств 29](#_Toc317173908)

[3.2 Реализация коннектора для подключения к торговому терминалу, шлюзу 29](#_Toc317173909)

[3.3 Реализация платформы для разработки торговых роботов 29](#_Toc317173910)

[3.4 Реализация пользовательского функционала системы 29](#_Toc317173911)

[3.5 Перспективы развития системы 29](#_Toc317173912)

[4. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 30](#_Toc317173913)

[4.1 Концепция технико-экономического обоснования 30](#_Toc317173914)

[4.2 Потребительские свойства 31](#_Toc317173915)

[4.3 Затраты на разработку 31](#_Toc317173916)

[4.3.1 Отчисления на социальные нужды 31](#_Toc317173917)

[4.3.2 Организационный план проекта 36](#_Toc317173918)

[4.4 Экономическая эффективность 42](#_Toc317173919)

[5 ОХРАНА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ 44](#_Toc317173920)

[5.1 Нарушение прав на программу для ЭВМ и базу данных 48](#_Toc317173921)

[5.2 Право на официальную регистрацию 48](#_Toc317173922)

[Выводы 59](#_Toc317173923)

[СПИСОК ЛИТЕРАТУРЫ 60](#_Toc317173924)

[ПРИЛОЖЕНИЕ 61](#_Toc317173925)

Определения

**Финансовый рынок** — это рынок, который опосредует распределение денежных средств между участниками экономических отношений. Можно сказать, это самое общее (широкое) название рынка, товаром на котором являются деньги в той или иной форме. На финансовом рынке происходит мобилизация капитала, предоставление кредита, осуществление обменных денежных операций и размещение финансовых средств в производстве. А совокупность спроса и предложения на капитал кредиторов и заёмщиков разных стран образует мировой финансовый рынок.

**Рынок ценных бумаг** (фондовый рынок) — это рынок, который опосредует кредитные отношения и отношения совладения с помощью ценных бумаг. Фондовый рынок это сегмент финансового рынка. Особенностью привлечения средств с помощью ценных бумаг является то, что,  как правило, они могут свободно обращаться на рынке. Поэтому лицо, вложившее деньги в какое-либо производство посредством приобретения ценных бумаг может вернуть их (полностью или частично), продав бумаги. В то же время, его действия не нарушают сам процесс производства: средства не изымаются из предприятия, и оно продолжает функционировать. Возможность свободной купли продажи ценных бумаг позволяет вкладчику гибко определять время, на которое он желает разместить свои средства в тот или иной хозяйственный проект.

**Денежный рынок**– это рынок, на котором обращаются краткосрочные ценные бумаги. Временным критерием обычно считается 1 год.

**Рынок капитала** – это рынок, на котором обращаются бессрочные ценные бумаги или бумаги, до погашения которых остается более года.

**Первичный рынок ценных бумаг**– это рынок, на котором происходит первичное размещение ценной бумаги. Любая ценная бумага первый раз продается именно на первичном рынке. Таким образом, термин «первичный рынок» относится к продаже новых выпусков ценных бумаг. В результате продажи бумаг на первичном рынке выпустившее их лицо получает необходимые ему финансовые ресурсы, а бумаги поступают в руки первоначальных держателей. Таким образом, функция первичного рынка состоит в мобилизации новых капиталов.

**Вторичный рынок ценных бумаг**– это рынок, на котором происходит обращение ценных бумаг после того, как их приобрели на первичном рынке. На вторичном рынке ценные бумаги перепродаются инвесторами друг другу.

**Биржевой рынок ценных бумаг** – рынок, включающий в себя обращение ценных бумаг на биржах.

**Спотовый** (кассовый) **рынок** – это рынок, на котором сделка заключается и сразу же исполняется (то есть сразу же происходит поставка ценной бумаги и платеж денежных средств по ней).

**Срочный рынок** – рынок, на котором заключаются срочные сделки. Срочная сделка представляет собой договор между контрагентами о будущей поставке предмета контракта в срок и на условиях, которые оговариваются в момент ее заключения.

**Эмиссионная ценная бумага** – это бумага, которая одновременно характеризуется следующими признаками:

* закрепляет совокупность имущественных и неимущественных прав
* размещается выпусками
* имеет равные объем и сроки осуществления прав внутри одного выпуска вне зависимости от времени приобретения ценной бумаги

**Акция** – это эмиссионная ценная бумага, которая закрепляет право ее владельца на получение части прибыли акционерного общества в виде дивидендов, на участие в управлении акционерным обществом и на часть имущества, остающегося после его ликвидации. Она является бессрочной, т.е. обращается на рынке до тех пор, пока существует выпустившее ее акционерное общество. Акционерное общество не обязано ее выкупать. Акции могут быть именными и на предъявителя. Выделяют две категории акций: обыкновенные (иногда их называют простыми) и привилегированные.

**Фондовый рынок (рынок ценных бумаг)** – является одним из сегментов финансового рынка. Предметом сделок купли-продажи на нем являются ценные бумаги, которые удостоверяют кредитные отношения и отношения совладения между участниками экономических отношений. Особенностью рынка ценных бумаг является то обстоятельство, что, как правило, эти ценные бумаги могут свободно обращаться на рынке. То есть лицо, вложившее деньги в какое-либо производство посредством приобретения ценных бумаг, может вернуть их (полностью или частично), продав бумаги другому участнику экономических отношений, в то время как сами денежные средства не изымаются из процесса производства и предприятие продолжает функционировать. Такая возможность свободной купли-продажи ценных бумаг позволяет вкладчику самостоятельно выбирать время, на которое он желает разместить свои средства в тот или иной бизнес-проект.

**Интернет-трейдинг** – современная технология, позволяющая упростить и ускорить процесс приема заявок от инвесторов на операции с ценными бумагами. Она позволяет Вам управлять Вашими инвестициями, покупать и продавать ценные бумаги посредством возможностей глобальной информационной сети Интернет. В общем случае под термином «Интернет-трейдинг» понимается возможность получения удаленного доступа к торгам через Интернет посредством специально созданного для этих целей программного обеспечения.

**Клиент-сервер —** вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением. Программный код, обеспечивающий интерактивность web-страниц, называется сценарием (скриптом).

**Алгоритм** – совокупность операций, выполняемых в строго установленном порядке, для решения поставленной задачи.

**DDE –**  (Dynamic Data Exchange) – механизм передачи обновленной информации между приложениями в операционных системах [Microsoft Windows](http://ru.wikipedia.org/wiki/Microsoft_Windows).

**Спрэд** – разница между лучшей ценой покупки и лучшей ценой продажи актива.

**Транзакция** – поручение клиента брокеру на совершение какой-либо операции. Примеры транзакций: ввод новой заявки, снятие заявки.

**Заявка** – согласие купить/продать ценные бумаги на указанных в поручениях условиях. Заявки на торги по условиям своего выполнения могут быть адресными и безадресными.

* **Адресная *заявка*** выражает намерение заключить сделку с определенным партнёром.
* **Безадресная заявка** выражает согласие заключить сделку, адресованное всем участникам торгов.

**Волатильность** – мера изменчивости цены определенного финансового инструмента за выбранный промежуток времени.

**OLE -** Object Linking and Embedding- технология связывания и внедрения объектов в другие документы и объекты.

**Стратегия** — это заранее формализованный алгоритм действий трейдера, автоматической системы на рынке, результат которого определен для определенных сценария развития событий.

**Короткая продажа (**Продажа без покрытия**) –** продажа ценных бумаг, товаров или валюты, которыми торговец на момент продажи не владеет.

Такая операция возможна, если условия контракта предусматривают его исполнение (поставку) через некоторое время или при маржинальной торговле, когда разрешено продавать взятый у брокера в кредит ценные бумаги с предполагаемой последующей покупкой аналогичного товара.

**Открытие позиции** – сделка покупки или продажи финансового инструмента, в результате которой у трейдера возникают обязательства осуществить противоположную сделку в том же объеме по тому же финансовому инструменту.

**Закрытие позиции** – сделка покупки или продажи финансового инструмента, в результате которой выполняются обязательства, возникшие у трейдера по предшествующей сделке открытия позиции, и производится расчет прибыли/убытка.

Введение

Сегодня все больше людей хотят работать и зарабатывать на фондовом рынке. Сейчас сложно представить работу трейдера и инвестора без использования современных компьютерных программ. Рынок программного обеспечения с каждым годом развивается и на сегодняшний день имеет все необходимое для полноценной работы трейдера, главное, не растеряться в таком большом количестве программ. Однако в России программы популярные в других странах практически не используются из-за отсутствия русскоязычной поддержки и высокой стоимости, что делает разработку такого рода программ актуальной задачей.

Как правило, российские трейдеры используют Excel-файлы для автоматизации торговли. При этом используется скрипт на VBA, который каждую секунду отправляет данные в торговый терминал, а сам расчет стратегий производится при помощи формул Excel. Используются только возможности обычных Excel-формул и условия. Понятно, что при таком подходе трудно реализовать сколько-нибудь сложные стратегии торговли.

При выборе средств автоматизации нужно руководствоваться классификацией, разделяющей программы для трейдеров на две категории:

К первой относятся программы для технического анализа исторической ценовой информации. После этого идут программы для интернет-торговли, которые позволяют покупать и продавать акции на бирже.

Одно преимущество, которое дает автоматическая торговая система трейдерам - это возможность вести круглосуточную торговлю. Вдобавок ко всему, используя автоматическую торговую систему, трейдер оберегает себя от постоянного *психологического давления* со стороны фондового рынка, а так же исключает человеческий фактор из торговли на рынке.

**Инвестор на фондовом рынке принимает участие** в целом комплексе сложных экономических отношений, связанных с производством и потреблением, конечным результатом которого является создание богатства. Инвестирование — это часть экономического процесса создания богатства, заключающаяся в предоставлении денежных средств на определенный (длительный) срок для целей обеспечения нормального функционирования систем производства, сбыта и управления компанией.

Неотъемлемым элементом инвестирования являются деньги (денежные средства), представленные в наличной и безналичной формах. Деньги являются товаром, необходимым для организации всего цикла производства. Без денег экономическая система перестает нормально функционировать. Многие экономисты условно сравнивают роль денег в экономической системе с функцией крови в организме человека: она переносит жизненно важный элемент — кислород, необходимый для жизнедеятельности всей системы. Следовательно, финансовый рынок представляет некое подобие кровеносной системы. Если она перестает работать — останавливается деятельность всего механизма. Именно поэтому органы государственной власти в настоящее время придают большое значение развитию финансового рынка в нашей стране. Рынок, на котором деньги (денежные средства) являются товаром, называется финансовым рынком. Финансовый рынок – это наиболее широкое понятие, которое включает в себя все возможные виды и способы торговли деньгами.

Данная дипломная работа посвящена разработке автоматизированной системы для устранения описанных проблем.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ
   1. Обзор устройства российского фондового рынка

Сейчас никого не удивишь наличием многочисленными ОАО и ЗАО, берущими свое начало в 90-х годах. Благодаря развитию интернет-технологий упростился доступ инвесторов к фондовому рынку.

Объектом сделок на фондовом рынке является ценная бумага. Ценную бумагу можно определить как денежный документ, который удостоверяет отношения совладения или займа между его владельцем и эмитентом.

Ценные бумаги могут выпускаться как в индивидуальном порядке, например векселя, так и сериями, например акции. В последнем случае законодательство говорит об эмиссионной ценной бумаге.

Ценная бумага может выпускаться в документарной и бездокументарной формах.

* 1. Обзор технологии интернет-трейдинга.

Исторически, покупка или продажа ценных бумаг осуществлялась в специально отведенных для этого местах, где собирались как продавцы, так и покупатели. С развитием финансовых рынков и тех и других стало слишком много. Выход был найден - создали биржи и ввели должность - профессиональный участник. Для приобретения ценных бумаг теперь приходилось идти к брокеру и подавать заявку через него (Рис 1). Долго, но надежно. С появлением телефона процесс стал более динамичным. Время, которое уходило на заключение сделки, сократилось до 20-30 минут. Источники информации, на основе которых принимались решения, остались те же – пресса, слухи, советы профессионального консультанта. Потом появился Интернет и как следствие Интернет-трейдинг.

По сути интернет-трейдинг - это деятельность по управлению инвестициями посредством Интернет, покупка и продажа ценных бумаг через Интернет.  В общем случае под термином "Интернет-трейдинг" понимается возможность удаленного доступа к торгам через Интернет, посредством специально созданного для этих целей программного обеспечения. Причём интернет-трейдинг может быть двух видов:

* **«Посреднический».** Хронологически – первый вид, по сути являющийся электронным отражением реального процесса торговлив ценными бумагами. Вкратце, суть интернет-трейдинга этого вида такова: американский брокер является номинальным держателем ценных бумаг клиента, и открывает ему доступ к своим торговым терминалам, подключенным к торговым системам и биржам. Доступ к системам брокера осуществляется через Интернет, и клиент может отдавать распоряжения по сделкам в режиме реального времени. Учитывая развитость рынка, клиенты осуществляют сделки и получают отчеты от брокеров, сами сделки регистрируются, а ценные бумаги реально меняют владельцев в депозитарно-клиринговой системе между счетами клиентов за весьма малое время, что позволяет заключать сделки на короткое время, открывая тем самым широкие возможности биржевой игры.

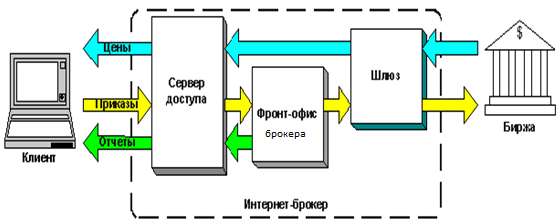


Рис.1 Схема работы интернет-трейдинга.

* **Самостоятельный.** Суть и главная особенность состоит в том, что практически инвестор самостоятельно действует на бирже в режиме реального времени с помощью специального программного обеспечения, а не через посредничество интернет-брокера, которому необходимо послать заявку на выполнение какой-либо операции и ждать её исполнения. Еще одно преимущество такого вида интернет-трейдинга – оперативность, т.е. есть возможность не только совершить сделку немедленно по текущей цене, но и немедленно отозвать сделку, если что-то изменилось
  1. Обзор существующих торговых терминалов.

Для предоставления услуг интернет-трейдинга брокеры предоставляют своим клиентам торговые терминалы различных систем. Благодаря этим программам можно видеть на мониторе текущие котировки по избранным инструментам, совершаемые в торговой системе сделки, динамически изменяющиеся после каждой сделки сведения о лимитах. Данные, экспортируемые с биржи, можно сохранять в архивах, строить графики, анализировать в программах технического анализа. Торговый терминалы позволяют в режиме реального времени формировать заявки и видеть их исполнение непосредственно в торговой системе. Наиболее важные показатели: информационное обеспечение и время исполнения заявки - практически аналогичны тем, что имеют профессиональные участники на бирже. На российском рынке присутствуют несколько различных торговых терминалов.

*Информационно-торговая система Quik* – Разработка компании ARQA Technologies (рис. 2). В России используется большинством брокеров. Основные возможности системы:

* Удобный интерфейс: все данные отображаются в виде таблиц различного назначения и графиков. Для удобства расположения большого числа окон предусмотрены "экранные закладки".
* Графическое предоставление данных. Возможность отобразить историю торгов за предыдущие торговые сессии.
* Многообразие индикаторов технического анализа.
* Отображение новостей в специальном окне.
* Просмотр текущего состояния портфеля клиента.
* Данные из большинства таблиц можно экспортировать в MS Excel (через протокол DDE) или в базы данных (посредством ODBC) в режиме реального времени, а также доступен экспорт данных для технического анализа.
* Возможность программировать дополнительные таблицы благодаря встроенному языку QPILE.
* Панель ввода заявок позволяет вводить новые заявки одним кликом, а функция "drag-and-drop" - заменять активные заявки "перетаскиванием". QUIK позволяет использовать условные заявки следующих типов: "стоп-лимит", "тэйк-профит", "тэйк-профит и стоп-лимит", "с условием по другому инструменту", "со связанной заявкой", а также тип заявки "по исполнению".

*Альфа-Директ* – разработка инвестиционной компании Альфа Капитал

* обладает высокой скоростью обновления котеровок(менее 0,4 секунд).
* Экспорт данных в программы технического анализа осуществляется по технологии OLE.
* Осуществления операций продажа/покупка одним кликом мыши.
* Технология работы с архивами «Data-on-demand» -«предоставление данных по требованию», реализованной в Альфа-Директ. Данная технология позволяет не хранить на локальном компьютере файлы с необходимыми данными, а получать всю информацию с сервера Альфа-Директ в любой момент, как только возникает такая потребность. Нет необходимости ежедневно тратить время на загрузку файлов данных для ведения собственных архивов. Чтобы получить в Альфа-Директ данные для построения графиков или загрузки их в систему технического анализа (MetaStock), достаточно просто указать инструмент и период – данные загрузятся автоматически.
* Система мониторинга - Альфа-Директ отображает состояние всех активов и обязательств трейдера, возникающих не только в процессе заключения сделок, но и расчетов по ним, а также в процессе переводов и других неторговых операций.(рис. 3)

|  |  |
| --- | --- |
| d_112_038.jpg | http://stockgraphics.narod.ru/AlfaDirectTerminal.GIF |
| Рис. 2 Quik | Рис. 3 Альфа-Директ |

* 1. Обзор существующих систем для создания торговых роботов.

Автоматизированные торговые системы дают большое преимущество трейдерам, которые используют их в торговле, по сравнению с теми, которые торгуют на рынке вручную. Трейдеру, который настроил свою автоматическую систему и запустил ее для торговли, больше не требуется сидеть часами у монитора и следить за состоянием на валютном рынке. Все, что требуется от такого трейдера – это проверять, как работает его автоматическая торговая система и, если требуется, корректировать ее настройки. Наиболее распространены в России системы:

*MetaStock* - это программа для проведения углубленного технического анализа, используемая как частными инвесторами, так и профессиональными трейдерами и аналитиками инвестиционных компаний и банков. MetaStock является признанным стандартом для рынка, используется более чем 300 000 пользователей.

MetaStock позволяет решить три основные задачи:

* Сканировать котировки фондового рынка.
* Анализировать данные
* Тестировать торговых стратегий.

Работая с MetaStock можно получить:

* более точное определение моментов открытия и закрытия позиции, чем в торговом терминале;
* системный подход к торговле;
* возможности торговли при помощи роботов.

В частности, к основным возможностям MetaStock относятся:

* Написание пользовательских индикаторов (Indicator Builder)
* Разработка и оптимизация торговых систем (Trade System)
* Обработка данных в режиме онлайн (подключать данные)

*TradeMatic Strategy Trader* – программа автоматической торговли, которая не требует специфических знаний и при этом позволяет задавать индивидуальные параметры и использовать собственную инвестиционную стратегию. Позволяет**:**

* использовать готовые торговые стратегии от разработчиков системы**;**
* торговать в автоматическом режиме**;**
* более точное определение моментов открытия и закрытия позиции**,** чем в торговом терминале;

Имеет функцию выбора режима торгов торговли:

* Автоматический — все сигналы на покупку/продажу исполняются автоматически, не требуя участия трейдера.
* Полуавтоматический — сигналы на покупку/продажу только после подтверждения**.**
* Ручной — для исполнения сигналов требуется ручное выставление заявок**.**
  1. Техническое задание

Тема разработки: Разработка системы для автоматической торговли на фондовой бирже.

Система будет реализована в виде фремворка, который будет установлена на сервере локальной сети предприятия.

Фреймворк должнен позволять выполнять следующие действия:

1. Определения наличие соединения с торговым терминалом.
2. Устанавливать/разрывать соединения с торговым терминалом.
3. Отправлять/отзывать заявки на покупку-продажу ценных бумаг.
4. Получать данные из торгового терминала и обрабатывать их.

Данная задача разбивается на следующие подзадачи:

1. Разработка коннектора для подключения к торговому терминалу.
   1. Прием данных по протоколу DDE.
   2. Обработка формата XLTable
   3. Создание библиотеки-обертки для API Quik под C#.
2. Разработка платформы для разработки торговых роботов.
   1. Виды анализа ситуации на рынке (котировок)

**Анализ** – расщепление целого на части. Позволяет собрать некоторую дополнительную информацию, знания об исследуемой системе. Однако анализ сам по себе не дает прямых оснований для предсказывания (прогнозирования) будущего.

*Фундаментально* - анализируется общая ситуация в экономике и стране в целом (политические заявления, экономические прогнозы). Это исключительно экспертные оценки, т.к. не существует научной теории, позволяющей точно предсказывать влияние на котировки вышеуказанных событий (интерпретация такого рода событий является творческим процессом, проявлением мастерства экспертов и пока что не представляется возможным его формализовать).

*Технический анализ* - прогнозирование изменений цен в будущем на основе анализа изменений цен в прошлом. В его основе лежит анализ временны́х рядов цен. Помимо ценовых рядов, в техническом анализе используется информация об объёмах торгов и другие статистические данные. Наиболее часто методы технического анализа используются для анализа цен, изменяющихся свободно, например, на биржах.

Природа этого в том, что в жизни, в частности, на финансовых рынках распространено такое явление: закономерности поведения системы, наблюдаемые в прошлом, вовсе не обязательно сохранятся и в будущем. Анализ позволяет получать информацию о прошлом, тем самым позволяет исследовать законы системы, существовавшие в прошлом. Таким образом, даже самый качественный, тщательный, детальный анализ не застрахован от того, что его результаты окажутся ошибочными. Именно в силу того свойства, что в будущем закономерности поведения системы изменятся по сравнению с анализируемым периодом.

Поэтому прогнозирование будущего всегда осуществляется строго посредством гипотезы. Выдвигается базовая гипотеза о том, что в будущем выявленные анализом закономерности поведения системы останутся неизменными и новые, еще неизведанные закономерности не возникнут. Только в таком случае можно построить обоснованный анализом прогноз поведения системы. Эксперимент, то есть действительное развитие событий может либо подтвердить (правильно сказать: не опровергнуть), либо опровергнуть гипотезу. Во втором случае прогноз оказывается ошибочным.

* 1. Понятие стратегии

В процессе научного познания важная роль принадлежит стратегиям, то есть набору (алгоритму) действий, правил поведения трейдера (исследователя) в отношении исследуемой системы, с помощью которого проводится эксперимент.

**Стратегия** — это заранее формализованный алгоритм действий трейдера (инвестора) на рынке, результат которого определен для различных сценариев развития событий.

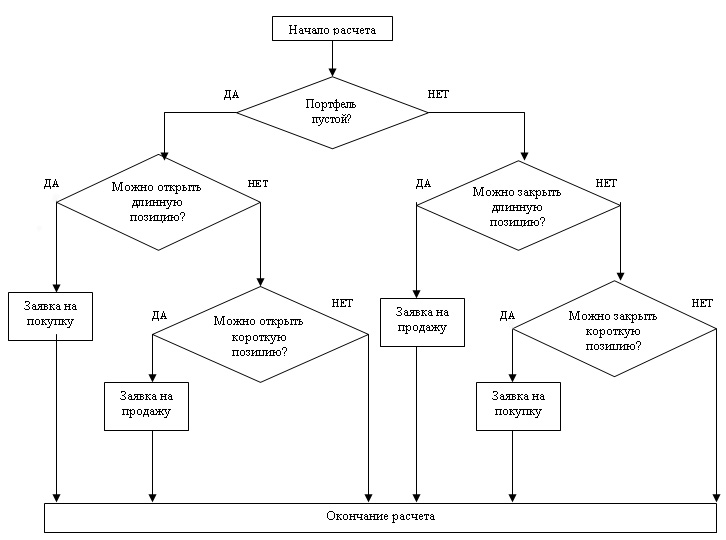


Рис. 4. Общая схема типовой стратегии

* 1. Средства разработки

**.NET Framework** – программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является исполняющая среда Common Language Runtime (CLR), способная выполнять как обычные программы, так и серверные веб-приложения. NET Framework поддерживает создание программ, написанных на разных языках программирования. Основным языком разработки является C#.

**C#** – объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

**Microsoft Visual Studio** – линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile.NET Framework, .NET Compact Framework и Microsoft Silverlight.

1. ПРОЕКТИРОВАНИЕ СИСТЕМЫ
2. Разработка общей архитектуры системы

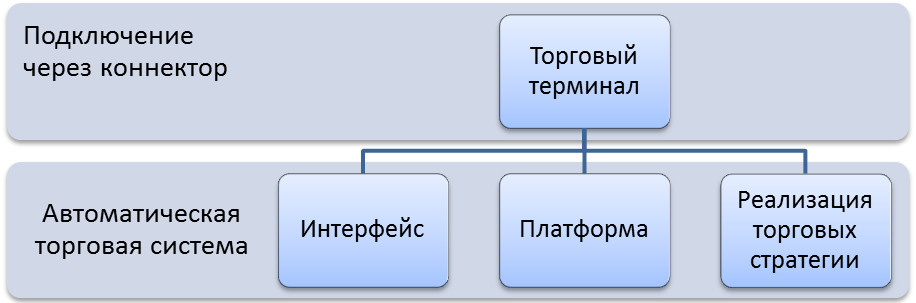


Рис. 5. Общая схема разрабатываемой системы.

*Платформа -* фреймворк для реализации торговых стратегий.

*Интерфейс -* визуализация работы платформы.

*Реализация торговых стратегий* - информационно-советующая или исполняющая система.

*Коннектор* - программная прослойка обеспечивающая независимость от системы торгового терминала. В качестве базового торгового терминала буду использовать Quik.

На рисунке 5 приведено концептуальная схеме разрабатываемой системе.

1. Выбор способа подключения системы к терминалу

|  |  |
| --- | --- |
| Для передачи данных из таблиц QUIK в другие приложения для их дальнейшего использования (хранения, обработки): | Скорость обработки  800 записей таблице сделок |
| Экспорт осуществляется в формате XLTABLE через метод обмена данными DDE | 0,6 секунд |
| Экспорт по ODBC в любую базу данных. | 0,9 секунда |

1. Разработка формата описания стратегий поведения робота

Стратегии представляют собой класс Strategy, который содержит основные торговые параметры такие как: портфель, инструмент, текущая позиция, прибыль-убыток и т.д.

Класс Strategy использует подход, основанный на [событийной модели](http://stocksharp.com/doc/html/8c0327db-494a-4fe0-aba8-0d4b93dea60e.htm). Такой подход отличается от обычного программирования.

В классе стратегии должны быть реализованы методы:

* Условия, при которых мы покупаем.
* Условия, при которых мы продаём.

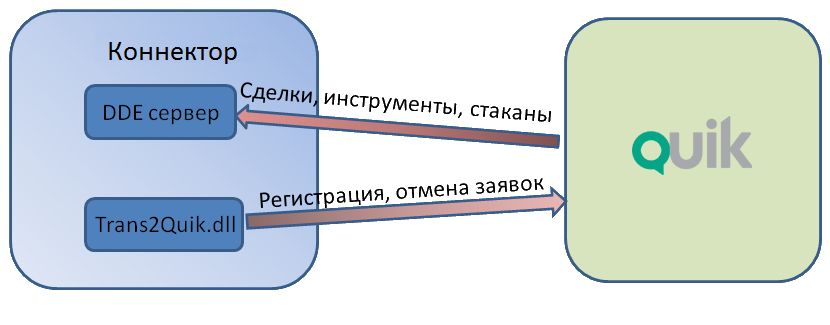
1. Последовательность действий при работе с Framework и необходимые навыки разработчика стратегий

При работе с созданным Framework разработчику необходимо:

* Вызвать функцию соединения с торговым терминалом;
* Подписаться на события получения данных из таблицы Quik;
* В обработчике события описать последовательность действий для обработки полученных данных (подготовить данные);
* Обработать полученные данные;
* Принять одно из следующих решений:
  + Решение о заявке на куплю-продажу.
  + Решение об отмене заявки о купле-продажи.
  + Ожидать.
* Передать принятое решение на торговый терминал.

Для реализации своей стратегии пользователю необходимы навыки программирования под платформу .NET на языке C#.

1. Разработка коннектора



Коннектор представляет набор функция для работы автоматической торговой система с торговым терминалом:

* Для получения данных с терминала в формате XlTable.
* Для отправки торговых приказов на терминал.

Коннектор представляет собой библиотеку на языке C# и представляет собой статический класс **QUIKAPI**. Коннектор предоставляет C# описание функций библиотеки **Trans2Quik.DLL**.

**Пример использования коннектора:**

using System;

using XlDde;

// Для работы данного примера требуются файлы XlDdeServer.cs, XlTable.cs

// и ссылка на библиотеку NDde.dll (http://ndde.codeplex.com/).

namespace QuikDDE\_sample

{

class sample

{

const string service = "DDESample";

const string tradesTopic = "trades[set3]";

const string stockTopic = "stock";

static void Main()

{

using(XlDdeServer server = new XlDdeServer(service))

{

// Создадим и зарегистрируем обработчики передаваемых Квиком данных

// для каждого нашего DDE канала

server.AddChannel(tradesTopic, new TradesChannel());

server.AddChannel(stockTopic, new StockChannel());

// Зарегистрируем сам DDE сервер

server.Register();

// DDE сервер готов к работе и каждый раз при поступлении

// данных он будет передавать их методу ProcessTable() в

// соответствующем каналу экземпляре класса.

Console.WriteLine("DDE server ready. Press Enter to exit.\n\n");

Console.ReadLine();

}

}

class TradesChannel : XlDdeChannel

{

protected override void ProcessTable(XlTable xt)

{

for(int row = 0; row < xt.Rows; row++)

{

for(int col = 0; col < xt.Columns; col++)

{

xt.ReadValue();

switch(xt.ValueType)

{

case XlTable.BlockType.Float:

Console.Write("D:{0}\t", xt.FloatValue);

break;

case XlTable.BlockType.String:

Console.Write("S:{0}\t", xt.StringValue);

break;

default:

Console.Write("{0}:{1}\t", xt.ValueType, xt.WValue);

break;

}

}

Console.WriteLine();

}

Console.WriteLine("\nIsConnected: {0}, Data received: {1}\n",

this.IsConnected, this.DataReceived);

}

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class StockChannel : XlDdeChannel

{

protected override void ProcessTable(XlTable xt)

{

Console.Write(".");

}

}

}

}

1. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ
2. Выбор используемых программных средств

Платформа и коннектор реализованы на языке программирования C#.

1. Реализация коннектора для подключения к торговому терминалу, шлюзу
2. Реализация платформы для разработки торговых роботов
3. Реализация пользовательского функционала системы
4. Перспективы развития системы
5. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ
6. Концепция технико-экономического обоснования

Возникнув сравнительно недавно, в последней трети ХХ века, и стремительно развиваясь фондовый рынок быстро ставит себе на службу все новые технические и информационные достижения. Работа многих инструментов валютной онлайн-биржи сегодня доведена до полного автоматизма: [прогноз рынка ФОРЕКС](http://www.systemforex.ru/analytics/forecast/), расчет стратегии трейдера, использование сигналов – все это может формироваться и действовать совершенно автономно, как будто бы оставляя самому участнику рынка только одну задачу – получение прибыли. В действительности, конечно, возможность поручения всех операций по торговле валютой компьютеру – это лишь иллюзия. Системы автоматической торговли требуют постоянного мониторинга и обязательного регулирования, поэтому на деле доверить системе распоряжение своими капиталами на валютной бирже может лишь очень опытный трейдер.

В реальности, подстроить свою торговлю на валютном рынке под действия готовой автоматической системы не так просто. Для этого нужно просто идеально разбираться в принципах ее работы. На самом деле, лучше совершить обратный ход: разработать (или заказать разработку) автоматической системы торговли на основе собственной стратегии. Дело за «малым» – разработать эту самую стратегию. А на это как раз могут уйти годы кропотливой подготовки.

Разработка подобного комплекта инструментов, готового всю вашу работу на бирже взять на себя, предполагает учет множества факторов, разнообразные тенденции и реакцию на них других участников рынка и т.п. В идеале, прежде чем запускать такую систему в работу, неплохо представить себя на месте компьютера и тщательно спрогнозировать все действия, которые будут предприниматься машиной в той или иной биржевой В последние годы  появилось много новых торговых систем. Большинство этих торговых систем направлены для того, чтобы автоматизировать торговую деятельность трейдера на валютном рынке и значительно упростить его работу и свести к минимуму возможность принятия поспешных решений. Большое многообразие всевозможных торговых систем определено тем, что на валютный рынок постоянно приходят всё новые трейдеры и многие из них ищут новые достаточно надежные торговые системы.

Разработкой торговых систем занимаются многие программисты со всего мира, часто этим вопросом занимаются целые коллективы опытных программистов. При разработке каждой торговой системы преследуется определенная цель, нередко новая торговая система создается с целью ее последующей реализации на определенном рынке, часто разработчики создают торговую систему для себя, и уже поняв ее недостаточную ценность, выкладывают ее для всеобщего ознакомления.

1. Потребительские свойства

Разработанный программный продукт, встраиваемая библиотека методом, обрабатывает данные от торгового терминала и отправляет заявки на сделки. Библиотека работает через протокол DDE, поэтому не требует вложений в специализированное платное ПО. Единственными ресурсами, необходимыми для использования библиотеки, является компьютер с любой современной операционной системой, подключенный к интернету. Библиотека предназначена для программистов-трейдеров торговых систем.

1. Затраты на разработку

4.3.1 Отчисления на социальные нужды

Рассчитаем величину отчислений на социальные нужды за четыре месяца.

С 1 января 2010 года работодатели вместо ЕСН будут платить страховые взносы в государственные внебюджетные фонды. Страховые взносы на обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний сохраняются. Порядок их расчетов установлен Федеральным законом от 24.07.1998 № 125-ФЗ.

Тарифы взносов во внебюджетные фонды для работодателей, применяющих общую систему налогообложения, в 2010 году останутся в размерах действующих в 2009 году ставок ЕСН (ст. 57 Закона № 212-ФЗ). Отметим, что "старые" ставки на 2010 год оставлены исключительно ввиду кризиса.

Законодатели решили увеличить ставки только с 2011 года (ст. 12 Закона № 212-ФЗ). Для наглядности увеличение и разбивку новых ставок по фондам представим в виде таблицы:

*Производство продукта*

Таблица 1

Технологический процесс производства продукции

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование технологической операции | Норма времени, час/шт | Технологическое оборудование | | | Производственные площади, кв.м. |
| Наименование (тип, марка) | Стоимость руб. | Способ приобретения |  |
| Разработка | 285 | Компьютер  Pentium Core 2 Duo 1 шт. | 23000 | Покупка | 12 |
|
|
|
| Подключение к интернету 15 Мбит/с, 2 мес. | 1000 | Аренда |

Число рабочих мест на каждой операции () рассчитывается с учетом прогнозируемого объема производства:

, 

где  - норма времени выполнения операции *i*, час/шт.;  - максимальный объем производства в планируемом периоде (шт). В качестве планируемого периода принимается неделя; *F* – фонд времени работы рабочего места, час.

Затраты на содержание и эксплуатацию оборудования определяются из расчета на 1 час работы оборудования с учетом стоимости и производительности оборудования:

,

где  - расчетная себестоимость одного машино-часа работы оборудования на *i*-й технологической операции (для 1 ПК стоимость электроэнергии на час работы усредним до 2р.), р/м-ч;  - количество машино-часов, затрачиваемых на выполнение *i*-й технологической операции, м-ч



• Трудоемкость выполнения работ Tи в месяц = 285 чел.-час.,

• Часовая ставка инженера Ди = 800 руб.

• Процент дополнительной заработной платы 12%.

• Страховые взносы (замена ЕСН с 2010 года, в т.ч.: ПФР – 26%, ФСС – 2,9%, ФФОМС – 2,1%, ТФОМС – 3%) = 34%.

• Прочие прямые расходы 5%.

• Процент накладных расходов НР = 20%.

В статье “Основная заработная плата” показывается сумма заработной платы сотрудников, непосредственно занятых разработкой проекта, а также заработная плата работников нештатного состава, привлеченных к ее выполнению. Основная заработная плата исполнителей рассчитывается по формуле:

Сзо = Tи\* Ди = 228000 руб.

Статья “Дополнительная заработная плата” включает выплаты, которые в соответствии с законодательством о труде или коллективным договором, осуществляется за неотработанное время. К таким выплатам относятся оплата отпусков, время выполнения общественных обязанностей и другие. Дополнительная заработная плата рассчитывается по формуле:

Сдз = Сзо\*0,12 = 27360 руб.

Статья “Страховые взносы” включает сумму отчислений от основной и дополнительной заработной платы, т.е. отчисления в фонды социального страхования, медицинского страхования, фонд занятости и пенсионный фонд. Отчисления осуществляются по законодательно установленной норме. Отчисления на социальные нужды рассчитываются по формуле:

Ссн = (Сзо+ Сдз)\*0,34 = 86822,4 руб.

Статья “Прочие прямые расходы” включает расходы на получение специальной научно-технической информации, платежи за использование средств связи, коммуникации и др. Прочие прямые расходы рассчитываются по формуле:

Сп.пр = (См+Сзо+ Сдз+ Ссн+ Спл)\*0,05 = 17134,92 руб.

Статья “Накладные расходы” включает расходы на управление и хозяйственное обслуживание, которые в равной степени относятся ко всем выполняемым работам. Накладные расходы рассчитываются по формуле:

Снр = (Сзо+ Сдз)\*НР = 51072 руб.

На основании полученных данных по отдельным статьям затрат составляется калькуляция себестоимости разработки в целом (таблица 3.5).

Таблица 2

|  |  |  |
| --- | --- | --- |
| **№** | **Статья затрат** | **Сумма, руб.** |
| 1 | Материалы | *0* |
| 2 | Специальное оборудование для научных и экспериментальных работ | *23000* |
| 3 | Аренда помещения | *24000* |
| 4 | Основная заработная плата | *228000* |
| 5 | Дополнительная заработная плата | *27360* |
| 6 | Отчисления на социальные нужды | *86822,4* |
| 7 | Прочие прямые расходы | *17134,92* |
| 8 | Накладные расходы | *51072* |
| 9 | Расходы на электроэнергию | *516* |
| Итого себестоимость | | *457905,32* |

Вывод: данные, представленные в таблице 3.5, показывают, что итоговая себестоимость разработки равна *457905,32 руб*.

4.3.2 Организационный план проекта

Таблица 3

Организационный план работ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование работ | Ответственный исполнитель | Ожидаемые сроки выполнения |
| п/п | (мероприятий) |
| 1 | Формулирование целей и требований к проекту | Мейстельман В.А | 30 |
| 2 | Изучение концептуальной, логической и физической моделей существующих систем | Мейстельман В.А | 50 |
|  |
| 3 | Анализ требований к программному комплексу | Мейстельман В.А | 20 |
| 4 | Разработка технического задания | Мейстельман В.А | 15 |
| 5 | Разработка концептуальных моделей | Мейстельман В.А | 50 |
| 6 | Разработка системы связей с внешними источниками | Мейстельман В.А | 20 |
| 7 | Разработка графического интерфейса | Мейстельман В.А | 40 |
| 8 | Тестирование программного продукта | Мейстельман В.А | 50 |
| 9 | Разработка документации | Мейстельман В.А | 10 |
|  | Всего |  | 285 |

**Прогноз финансовых показателей проекта**

Капитальные вложения в научно-техническую продукцию определяются прямым счетом на основе планируемых (фактических) затрат на их создание. Величина этих затрат принимается равной себестоимости разработки.

Капитальные вложения в основные средства определяются единовременными затратами на приобретение дополнительных производственных и административных помещений, технологического оборудования, транспортных средств и иных основных средств, необходимых для организации производства и сбыта новой продукции.

Капитальные вложения в производственные и административные помещения определяются по формуле

,

где *Ч* – число дополнительных рабочих мест, создаваемых для организации производства новой продукции, определяемое как численность основных, вспомогательных рабочих, специалистов и административно-управленческого персонала, задействованного в проекте; *П* – усредненная площадь одного рабочего места с учетом потребности в производственных площадях, занимаемых оборудованием, кв.м; *Ц* – рыночная стоимость одного квадратного метра производственных (офисных) помещений, р/кв.м..

Возьмем помещение 12 кв.м.



Капитальные вложения в технологическое оборудование, транспортные средства, передаточные устройства определяются по формуле

,

где  - количество единиц оборудования *i*-го вида, необходимых для организации производства новой продукции в планируемых объемах, шт.;  - цена единицы оборудования *i*-го вида, руб; *m* – количество видов оборудования, транспортных средств, передаточных устройств и т.п.; *H* – доля затрат на транспортировку, монтаж и наладку от стоимости оборудования, передаточных устройств примем равной 4%.



Основные результаты расчетов капитальных вложений во внеоборотные активы сводятся в табл. 3.7, в которой указываются также нормативные сроки использования каждого из активов.

Таблица 4

Капитальные вложения во внеоборотные активы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п  п | Наименование актива | Затраты на приобретение, тыс.р. | Нормативный срок использования, лет | Годовая амортизация, тыс.р. |
| 1. | Основные средства | 23 | - | 2,3 |
| в том числе: |
| 1.1. | Оборудование | 23 | 10 | 2,3 |
| 1.2 | Аренда помещения | 12 | - | - |
| ВСЕГО | | 35 | - | 2,3 |

При расчете величины текущих производственно-сбытовых издержек, связанных с реализацией проекта, в их составе целесообразно выделить:

* переменные издержки, величина которых будет меняться с изменением объемов производства и продаж. В состав переменных издержек включаются затраты на расходные материалы, заработная плата персонала с начислениями. Переменные издержки рассчитываются прямым счетом на единицу продукции, а затем пересчитываются на установленные объемы производства в планируемом периоде;
* постоянные издержки, величина которых является неизменной при изменении объемов производства и продаж. Величина этих издержек рассчитывается на месяц . В состав этих издержек включаются амортизация внеоборотных активов, вовлеченных в проект (см. табл. 3.8), затраты на оплату арендуемых активов (арендная плата за помещения, оборудование и т.п.), административные и прочие общие расходы предприятия, относимые на проект.

Таблица 3.8.

Производственно-сбытовые издержки

Планируемый период – 2 месяца.

Объем производства и продаж в планируемом периоде – 1 шт.

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование показателей | Значение показателя на единицу продукции, р. |
| 1. | Переменные издержки | 342698,4 |
| 1.1. | Сырье и материалы | 0 |
| 1.2. | Заработная плата производственных рабочих | 255360 |
| 1.3. | Отчисления на социальные нужды | 86822,4 |
| 1.4. | Затраты на содержание и эксплуатацию производственного оборудования | 516 |
| 2. | Постоянные издержки | 24767 |
| 2.1. | Амортизация внеоборотных активов | 2300 |
| 2.2. | Арендная плата | 24000 |
| Производственно-сбытовые издержки,  Всего | | 367465,4 |

Величина начальных инвестиций в оборотные средства определяется на основе расчета текущих производственно-сбытовых издержек, связанных с производством и сбытом новой продукции (см. табл. 4.8), планируемой длительности производственно-сбытового цикла продукции, а также ритмичности производства новой продукции.

Расчет величины начальных инвестиций в оборотные средства проведем по следующей формуле:

,

где  - переменные издержки на производство и реализацию единицы продукции, тыс.р/шт.;  - планируемый объем производства продукции, шт./мес.;  - величина постоянных издержек при производстве и реализации продукции в расчете на 2 месяц, тыс. р;  - длительность производственно-сбытового цикла. Показатели  и  берутся из данных табл. 5.



Результаты расчета потребности в начальных инвестициях представляются в табличной форме (табл. 3.9).

Потребность в начальных инвестициях равна сумме капитальных вложениям во внеоборотные и оборотные активы. Результаты расчета потребности в начальных инвестициях представляются в табличной форме (табл. 6).

Таблица 6

Потребность в начальных инвестициях

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование показателей | Сумма, тыс. руб. |
| 1.1 | Основные средства  в том числе: | 47 |
| 1.1.1 | * здания и сооружения | 24 |
| 1.1.2 | * оборудование | 23 |
| 1.1.3 | * прочие основные средства | - |
| 1.3 | Прочие внеоборотные активы | - |
| 2 | Оборотные активы | 367,495 |
| 2.1 | Запасы | - |
| 2.2 | Прочие оборотные активы | - |
|  | Потребность в начальных инвестициях, всего | 414,495 |

1. Экономическая эффективность

**Основные показатели**

1. Себестоимость = *457905,32 руб.*
2. Амортизация

Для расчета амортизации можно применить способ списания стоимости пропорционально к объему произведенной продукции. При таком методе вначале следует вычислить амортизацию на единицу продукции при помощи формулы:

А = С / В,

где А – это сумма амортизации, применяемая на единицу продукции, С – это первоначальная стоимость объекта основных средств, а В – планируемый объем производства продукции.

1. A = *329 000* / *100* = *3290 руб*.

Тогда общая амортизация вычисляется по формуле:

1. Aобщ = A \* 25,
2. Aобщ = *3290 \* 25* = *82250* руб.

Период окупаемости вычисляется по формуле:

PBP = I / (π + Aобщ),

где I – инвестиции (эквивалентны себестоимости), π – доход за один период,

1. Aобщ – амортизация.

Примем стоимость одного экземпляра *3500 руб*., планируемое количество проданных копий – *50* в год:

1. PBP = *93дня.*

Показатель ROI вычисляется по формуле:

1. ROI =  *578%*

**Вывод**

Таблица 3.10

|  |  |
| --- | --- |
| Параметр | Значение |
| Себестоимость (руб.) | *457905* |
| Срок окупаемости (дней) | *93* |
| Коэффициент качества | *2,1* |
| ROI (%) | *578 %* |

Исходя из полученных результатов, можно сделать вывод, что разработка проекта является рентабельной.

**5 ОХРАНА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ**

**5.0 Введение**

Мною, Мейстельман Виталием Александровичем, в процессе выполнения дипломного проекта разработана программа для ЭВМ «Система для автоматической торговли на фондовой бирже». Этот результат научно-технической деятельности входит в перечень охраняемых объектов интеллектуальной собственности Гражданского Кодекса РФ. Программа разработана по заданию работодателя Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования “Санкт-Петербургский государственный электротехнический университет “ЛЭТИ” им. В.И.Ульянова (Ленина)” (СПбГЭТУ)

**5.1 Интеллектуальная собственность**

Согласно определению интеллектуальной собственности, принятому в российском законодательстве, а также на основании определения Стокгольмской конференции от 14 июля 1967 г., компьютерные программы относятся к объектам интеллектуальной собственности. Компьютерным программам предоставляется охрана нормами авторского права как литературным произведениям в соответствии с Бернской конвенцией.

В Российской Федерации вопросы предоставления правовой охраны программам регулируются Гражданским кодексом РФ, Часть 4 (ГК РФ Ч.4).

**5.2 Программа для ЭВМ**

Под **программой для ЭВМ** понимается *"... представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств в целях получения определенного результата"*. Кроме того, в понятие программы для ЭВМ входят *"...подготовительные материалы, полученные в ходе разработки программы для ЭВМ, и порождаемые ею аудиовизуальные отображения"* [2, ст. 1261].

С точки зрения программистов и пользователей программа для ЭВМ представляет собой детализацию алгоритма решения какой-либо задачи и выражена в форме определенной последовательности предписаний, обеспечивающих выполнение компьютером преобразования исходных данных в искомый результат.

Можно выделить следующие объективные формы представления программы для ЭВМ:

* **исходная программа** (или исходный текст) - последовательность предписаний на алгоритмическом языке высокого уровня, предназначенных для автоматизированного перевода этих предписаний в последовательность команд в объектном коде;
* **рабочая программа** (или объектный код) - последовательность машинных команд, т. е. команд, представленных на языке, понятном ЭВМ;
* **программа, временно введенная в память ЭВМ,** - совокупность физических состояний элементов памяти запоминающего устройства ЭВМ (ОЗУ), сохраняющихся до прекращения подачи электропитания к ЭВМ;
* **программа, постоянно хранимая в памяти ЭВМ,** - представленная на языке машины команда (или серия команд), выполненная в виде физических особенностей участка интегральной схемы, сохраняющихся независимо от подачи электропитания [4].

Исходная и рабочая программы представляются в электронном виде.

Правовая охрана программ для ЭВМ распространяется только в отношении формы их выражения и *«… не распространяется на идеи, концепции, принципы, методы, процессы, системы, способы, решения технических, организационных или иных задач, открытия, факты, языки программирования»* [2, ст.1259, п. 5].

**5.3 Авторское право на программу для ЭВМ**

Предпосылкой охраноспособности программы для ЭВМ и базы данных является их творческий характер, т. е. они должны быть продуктом личного творчества автора. Творческий характер деятельности автора предполагается до тех пор, пока не доказано обратное [2, ст. 1257].

Момент возникновения авторского права является важнейшим юридическим фактом, который устанавливается в силу создания программы для ЭВМ. *"Для возникновения, осуществления и защиты авторских прав не требуется регистрация произведения или соблюдение каких-либо иных формальностей"* [2, ст.1259, п.4].

Закон устанавливает, что обнародование программы не является обязательным условием для возникновения прав на нее: *“Авторские права распространяется как на обнародованные, так и на необнародованные произведения, выраженные в какой-либо объективной форме …”* [2, ст. 1259, п. 3].

Таким образом, только сам факт создания программы, зафиксированной в объективной форме, является основанием возникновения авторского права на нее.

Каждая составляющая понятия использования программы для ЭВМ имеет конкретное содержание, которое также определено законом:

* **воспроизведение** - "... изготовление одного или более экземпляров произведения или его части в любой материальной форме, … в том числе запись в память ЭВМ" [2, ст. 1270, п. 2, п.п.1];
* **распространение** – предоставление доступа к произведению "... путем продажи или иного отчуждения его оригинала или экземпляров" [2, ст. 1270, п. 2, п.п.2].
* **публичный показ -** "... любая демонстрация оригинала или экземпляров произведения непосредственно … … либо с помощью технических средств в месте, открытом для свободного посещения, или в месте, где присутствует значительное число лиц … " [2, ст. 1270, п. 2, п.п.3];

В целях оповещения о своих правах правообладатель "... вправе использовать знак охраны авторского права, который помещается на каждом экземпляре произведения и состоит из следующих элементов: латинской буквы С в окружности; имени или наименования правообладателя; года первого опубликования произведения" [2, ст. 1271].

Исключительные права на программу переходят по наследству в установленном законом порядке, и их можно реализовать в течение срока действия авторского права.

Передача прав на материальный носитель не влечет за собой передачи каких-либо прав на программу для ЭВМ [2, ст.1227]. Иными словами, передача носителя информации (например, диска) с зафиксированной на нем программой третьему лицу не означает передачи каких-либо прав на эту программу.

**5.4 Правообладание**

Если человек (или группа людей) самостоятельно, по личной инициативе создал программу для ЭВМ, то он является одновременно и автором, и правообладателем созданного произведения, что позволяет ему по собственному усмотрению использовать эту программу (или базу данных) в личных целях, продавать, раздавать бесплатно, разрешать тиражировать и распространять или иным образом распоряжаться своими исключительными правами.

Кроме личных (неимущественных) прав автору “служебной” программы для ЭВМ (базы данных) принадлежит право на вознаграждение при условии использования работодателем созданных произведений или передачи исключительного права другому лицу. Размер и порядок выплаты этого определяется договором [2, ст. 1295, п. 2,].

**5.1 Нарушение прав на программу для ЭВМ и базу данных**

Специфика программ для ЭВМ такова, что они очень уязвимы в смысле их незаконного использования (прежде всего, путем копирования и распространения копий). Незаконно изготовленные (скопированные) или используемые экземпляры программы для ЭВМ или базы данных называются контрафактными, а несанкционированное использование чужих программ или баз данных путем опубликования (выпуска в свет), воспроизведения (полного или частичного), распространения, иного использования считается нарушением исключительных прав на программы для ЭВМ или базы данных, т. е. нарушением авторского права.

**5.2 Право на официальную регистрацию**

В ст. 1262 ГК РФ Ч.4 [2] закреплено право автора или иного правообладателя на государственную регистрацию программы для ЭВМ или базы данных: *"Правообладатель в течение срока действия исключительного права на программу для ЭВМ может по своему желанию зарегистрировать такую программу для ЭВМ или такую базу данных в федеральном органе исполнительной власти по интеллектуальной собственности "*. Исключение составляют программы для ЭВМ и базы данных, в которых содержатся сведения, составляющие государственную тайну.

Предусмотренная регистрация не является правообразующей и носит факультативный характер, т. е. с ней не связано возникновение прав на программу для ЭВМ, однако такая процедура представляется полезной по следующим соображениям.

1. Она является **официальным уведомлением общественности** о наличии у правообладателей прав в отношении рассматриваемых объектов.
2. Государственная регистрация **содействует защите прав** в случае возникновения конфликтных ситуаций при нарушении прав или установлении приоритета.

**5.7 Процедура официальной регистрации**

Процедура официальной регистрации программ для ЭВМ и баз данных в целом определена ст. 1262 ГК РФ Ч.4 и включает подачу заявки в федеральный орган исполнительной власти по интеллектуальной собственности (Роспатент), проверку поданных документов и собственно регистрацию. После поступления заявки на регистрацию в Роспатент проверяется наличие необходимых документов и их соответствие установленным требованиям. При положительном результате проверки сведения о программе для ЭВМ или базе данных вносятся, соответственно, в Реестр программ для ЭВМ или Реестр баз данных под уникальным регистрационным номером и выдается заявителю (здесь заявителем называют правообладателя, подавшего заявку на регистрацию программы или базы данных в Роспатент) свидетельство о государственной регистрации установленной формы, в котором указаны регистрационный номер объекта по Реестру, название программы или базы данных, имя или наименование правообладателя, фамилии авторов и дата регистрации. Сведения о зарегистрированных программах для ЭВМ и базах данных публикуются в официальном бюллетене Роспатента.

**5.8 Заявка на официальную регистрацию**

Состав заявки на официальную регистрацию программы для ЭВМ или базы данных (далее - Заявка) определен п. 2 ст. 1262 ГК РФ Ч.1, а также в Правилах составления, подачи и рассмотрения заявок на официальную регистрацию программ для электронных вычислительных машин и баз данных (далее - Правила) [3] .

Заявка должна относиться к одной программе или одной базе данных. При этом *"Программа для ЭВМ, состоящая из нескольких программ для ЭВМ (программный комплекс), которые не могут быть использованы самостоятельно, регистрируется в целом (без регистрации каждой входящей в нее (него) программы для ЭВМ)"* [3, п. 5]. Заявка должна содержать следующие документы:

* заявление о государственной регистрации;
* депонируемые материалы, идентифицирующие программу для ЭВМ, включая реферат;
* документ, подтверждающий уплату государственной пошлины в установленном размере или основание для освобождения от уплаты государственной пошлины или уменьшения его размера.

В Правилах подробно описаны требования, предъявляемые к документам заявки.

Заявление на официальную регистрацию представляется отпечатанным на типографском бланке или в виде компьютерной распечатки согласно образцам, приведенным в приложениях к Правилам (формы РП и РП/ДОП). Заявление представлено в ниже.

В объем депонируемых материалов включается титульный лист, при оформлении которого следует руководствоваться п. 18 Правил, а также ГОСТ 19103-77 и ГОСТ 19104-78.

**ЛИЦЕНЗИОННЫЙ ДОГОВОР**

**НА ИСПОЛЬЗОВАНИЕ ПРОГРАММЫ ДЛЯ ЭВМ**

**Стороны в Договоре:**

Гражданин **Мейстельман Виталий Александрович, проживающий по адресу: 190000, г. Санкт-Петербург, ул. Бадаева, д. 1, кв. 13,** именуемый в дальнейшем "ЛИЦЕНЗИАР", с одной стороны, и **Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования “Санкт-Петербургский государственный электротехнический университет “ЛЭТИ” им. В.И.Ульянова (Ленина)” (СПбГЭТУ),** именуемый в дальнейшем "ЛИЦЕНЗИАТ", в лице **проректора по научной работе Шестопалова М.Ю**., действующего на основании Доверенности, с другой стороны,

**принимая во внимание:**

1. что Лицензиар является автором и правообладателем программы для ЭВМ **"Система для автоматической торговли на фондовой бирже ";**
2. Лицензиат желает получить на условиях настоящего Договора лицензию на использование упомянутой программы для ЭВМ с целью проведения научных исследований в области медицинского приборостроения;
3. Лицензиар готов предоставить Лицензиату такую лицензию,

договорились о следующем.

**1. Термины и их определения**

1. "ПРОГРАММА ДЛЯ ЭВМ (ПрЭВМ)" - программное обеспечение

**"Система для автоматической торговли на фондовой бирже"**.

1. "ДОКУМЕНТАЦИЯ" - комплект документов, передаваемых Лицензиаром Лицензиату, включающий руководство пользователя по применению и обслуживанию программы для ЭВМ.
2. "ПРОИЗВОДСТВЕННАЯ ПЛОЩАДКА" - **научные лаборатории и кафедры** Лицензиата.
3. "РАБОЧЕЕ МЕСТО" - конкретная ЭВМ, на которой используется Программа для ЭВМ.

**2. Предмет Договора**

1. Лицензиар предоставляет Лицензиату на срок действия настоящего Договора и за вознаграждение, уплачиваемое Лицензиатом, неисключительную лицензию на использование ПрЭВМ. При этом Лицензиату предоставляется право на установку ПрЭВМ не более чем на 10 (десяти) Рабочих местах.
2. Лицензиар передает Лицензиату Документацию к ПрЭВМ.
3. Предоставленное Лицензиату в рамках настоящего Договора право ограничено Производственной площадкой.
4. Лицензиар осуществляет авторский контроль за соблюдением объемов использования ПрЭВМ по настоящему Договору, при этом Лицензиат обеспечивает возможность такого контроля.
5. Лицензиар сохраняет за собой право самому использовать ПрЭВМ и предоставлять неисключительные лицензии на право ее использования третьим лицам.

**3. Обеспечение Договора**

1. Лицензиар передает Лицензиату ПрЭВМ в объеме и виде, достаточном для ее использования, и Документацию в течение 15 (пятнадцати) дней со дня подписания настоящего Договора. ПрЭВМ передается Лицензиату в виде в количестве 5 (пяти) штук, содержащих ПрЭВМ. По факту передачи ПрЭВМ и Документации составляется акт сдачи-приемки с перечнем переданных материалов, подписываемый обеими Сторонами.
2. Если Лицензиат установит неполноту или неправильность полученных ПрЭВМ или Документации, то Лицензиар в течение 15 (пятнадцати) дней после сообщения ему об этом Лицензиатом обязан передать недостающие материалы или устранить недостатки ранее переданных ПрЭВМ и Документации.
3. Для оказания помощи в освоении ПрЭВМ Лицензиар по просьбе Лицензиата оказывает консультации пользователям ПрЭВМ.
4. Для целей использования ПрЭВМ в объеме, предусмотренном п. 2.1 настоящего Договора, Лицензиат может изготавливать в необходимом ему количестве копии ПрЭВМ и копии Документации.

**4. Усовершенствования**

1. Лицензиар обязуется незамедлительно информировать Лицензиата о всех произведенных им усовершенствованиях ПрЭВМ и, при желании Лицензиата, передать ему в согласованные сроки новые варианты ПрЭВМ. В отношении новых вариантов ПрЭВМ, переданных Лицензиаром Лицензиату, распространяются все условия настоящего Договора.
2. Лицензиат обязуется предоставлять Лицензиару информацию об использовании ПрЭВМ, которая могла бы быть полезной для усовершенствования ПрЭВМ.

**5. Платежи**

1. За предоставление прав, предусмотренных настоящим Договором, Лицензиат выплачивает Лицензиару единовременное вознаграждение в размере 50000(двадцать тысяч) рублей.
2. Вознаграждение, предусмотренное п. 5.1 настоящего Договора, выплачивается Лицензиатом в течение 30 (тридцати) дней, следующих после подписания акта приемки-сдачи.

**6. Реклама**

1. Лицензиат обязуется при опубликовании результатов исследований, полученных с использованием ПрЭВМ, сообщать в рекламных целях, что исследования производились с использованием ПрЭВМ Лицензиара с указанием авторского права Лицензиара.

**7. Защита передаваемых прав**

1. Лицензиат обязуется не вносить самовольно каких-либо изменений в ПрЭВМ и Документацию и не дополнять их какими-либо комментариями. Подобные изменения или дополнения возможны только с согласия Лицензиара.
2. Лицензиат обязуется предпринимать все необходимые меры для предотвращения несанкционированного копирования ПрЭВМ и Документации третьими лицами, а также несанкционированной передачи ПрЭВМ и Документации работниками Лицензиата третьим лицам.
3. Если Лицензиату станет известно о противоправном использовании ПрЭВМ третьими лицами, то он незамедлительно сообщит об этом Лицензиару.

**8. Ответственность Сторон и разрешение споров**

1. За невыполнение или ненадлежащее выполнение обязательств по настоящему Договору Стороны несут имущественную ответственность в соответствии с действующим законодательством.
2. Стороны освобождаются от ответственности за неисполнение или ненадлежащее исполнение обязательств, принятых по настоящему Договору, если неисполнение явилось следствием обстоятельств непреодолимой силы (форс-мажор).
3. Сторона, нарушившая свои обязательства по настоящему Договору, освобождается от ответственности за неисполнение или ненадлежащее исполнение этих обязательств, если это нарушение было вызвано причинами, за которые отвечает другая Сторона.
4. В случае возникновения споров между Лицензиаром и Лицензиатом по вопросам, предусмотренным настоящим Договором, Стороны примут все меры к разрешению их путем переговоров между собой. В случае невозможности разрешения указанных споров путем переговоров они будут разрешаться в порядке, предусмотренном действующим законодательством.

**9. Срок действия Договора и условия его расторжения**

1. Настоящий Договор заключен на срок 2 года и вступает в силу с даты его подписания обеими Сторонами.
2. По истечении срока действия настоящего Договора Лицензиат вправе использовать ПрЭВМ, включая усовершенствованные варианты, на Производственной площадке на любом количестве Рабочих мест. При этом обязательства Лицензиата, предусмотренные пп. 7.1 и 7.2 настоящего Договора, сохраняются бессрочно.
3. Действие настоящего Договора по обоюдному согласию Сторон может быть досрочно прекращено, но не ранее чем через три месяца после предложения об этом одной из Сторон. При этом Лицензиат не освобождается от обязательств по платежам, возникшим до расторжения настоящего Договора.
4. Настоящий Договор может быть досрочно расторгнут в одностороннем порядке со стороны Лицензиара из-за невыполнения Лицензиатом своих обязательств по пп. 7.1 или 7.2. В этом случае Лицензиат лишается права дальнейшего использования ПрЭВМ в любой форме и обязан вернуть ее Лицензиару.
5. Если Лицензиат откажется от дальнейшего использования ПрЭВМ, то он уничтожит все имеющиеся у него копии ПрЭВМ.

**10. Заключительные положения**

1. Все изменения и дополнения к настоящему Договору действительны только в тех случаях, если они совершены в письменной форме и подписаны обеими Сторонами.
2. Стороны не имеют права передавать свои права и обязательства по настоящему Договору третьим лицам без письменного согласия на то другой Стороны.
3. Во всем остальном, что не предусмотрено условиями настоящего Договора, будут применяться нормы законодательства Российской Федерации.

**11. Адреса Сторон**

1. ЛИЦЕНЗИАР: **Мейстельман Виталий Александрович, проживающий по адресу: 190000, г. Санкт-Петербург, ул. Бадаевава, д. 1, кв. 13**
2. ЛИЦЕНЗИАТ: **СПбГЭТУ, адрес: 197376, Санкт-Петербург, ул. Проф. Попова, д. 5.**

Настоящий Договор составлен в двух экземплярах для каждой из Сторон и подписан "\_\_\_"\_\_\_\_\_\_\_\_\_\_ 200\_\_ г. в г. Санкт-Петербурге.

|  |  |
| --- | --- |
| **ЛИЦЕНЗИАР:** | **От ЛИЦЕНЗИАТА:** |
| \_\_\_\_\_\_\_\_\_\_\_ Мейстельман В.А. | Проректор по научной работе СПбГЭТУ  \_\_\_\_\_\_\_\_\_\_\_\_ М.Ю. Шестопалов |

1. документ, подтверждающий уплату государственной пошлины в установленном размере или основание для освобождения от уплаты государственной пошлины или уменьшения его размера.

Далее представлены документы, необходимые для официальной регистрации

**ПРОГРАММА ДЛЯ ЭВМ**

**Система для автоматической торговли фондовой бирже**

Исходный текст программы

Всего 57 листов

Правообладатель: Мейстельман Виталий Александрович

Автор: \_Мейстельман Виталий Александрович

(ф., и., о.)

(C) Мейстельман В.А., 2012

Санкт-Петербург

2012

**СОСТАВ ПРОГРАММНОГО КОМПЛЕКСА**

Trade.DLL – библиотека фреймворка.

Help.docx – файл справки.

**Р Е Ф Е Р А Т**

Автор: Мейстельман Виталий Александрович

Правообладатель: Мейстельман Виталий Александрович

Программа для ЭВМ: Система для автоматической торговли на фондовой бирже

Аннотация: Программный комплекс предназначен для автоматизации торговых стратегий. Интегрируется с программными комплексами для торговли на фондовой бирже и расширяет их функциональность. Так же основным преимуществом является универсальность данного средства и легкость настройки под конкретный торговый терминал брокера. Предоставляет библиотеки для реализации торговых стратегий, графический интерфейс пользователя.

Тип ЭВМ: IBM PC/AT и совместимые с ней

ОС: Windows XP/Vista/7

Язык программирования: С# (NET Framework 4.0)

Объем: 307 Кбайт

Выводы

В результате работы над дипломным проектом:

* Изучены стратегии торговли на бирже и имеющиеся на российском рынке решения для автоматизации торговли
* Разработано приложение-робот для торговли на бирже по выбранным торговым стратегиям
* Приложение протестировано.

Созданная система реализует различные стратегии торговли на фондовой бирже. Основной задачей при разработке являлось создание подходящей платформы, предоставляющей весь необходимый для робота базовый функционал и удобной для реализации различных стратегий.

Созданная система взаимодействует с информационно-торговой системой Quik для получения сведений о текущем состоянии на бирже (котировках, заключенных сделках и отправленных заявках) и отправки торговых приказов. Автором реализована работа всех необходимых протоколов взаимодействия с системой Quik. Стратегия формализуется трейдером-аналитиком и представляется в виде алгоритма действий, а затем описывается на языке программирования C# (платформа предоставляет C# API). Стратегия – это C#-класс, подписывающийся и реагирующий на различные события платформы.

СПИСОК ЛИТЕРАТУРЫ

* + 1. Технико-экономическое обоснование дипломных проектов. Под ред. Проф. Беклкшова В.К., М. Высшая школа, 1999. – 65 с.
    2. С.В. Маклаков. CASE – средства разработки информационных систем.- М.: Диалог-Мифи, 2000. – 120 c.
    3. С.А. Орлов. Технологии разработки программного обеспечения. СПб.: Питер, 2002. – 463 c.
    4. Д. Н. Колисниченко - Самоучитель PHP 5.
    5. [http://ru.wikipedia.org](http://ru.wikipedia.org/) – русскоязычная свободная энциклопедия Википедия.
    6. C# 3.0. Справочник. Джозеф Албахари, Бен Албахари. БХВ-Петербург.
    7. Технический анализ фьючерсных рынков: Теория и практика. Джон Дж. Мерфи. Евро. 2008 г.

ПРИЛОЖЕНИЕ

Листинг коннектора для работы с терминалом:

#region Used Namespaces

// System Namespases

using System;

using System.Runtime.InteropServices;

using System.Collections.Generic;

using System.Text;

#endregion

namespace TRANS2QUIKSHARP

{

/// <summary>

/// Класс предоставляет C# описание функций библиотеки TRANS2QUIK.DLL

/// </summary>

public static class QUIKAPI

{

#region Member

/// <summary>

/// Last Executed Function Name

/// </summary>

private static System.String m\_LEF = string.Empty;

/// <summary>

/// Extended Error Code

/// </summary>

private static System.Int32 m\_EEC = 0;

/// <summary>

/// Error Message

/// </summary>

private static System.String m\_ERM = string.Empty;

/// <summary>

/// Error Message Size

/// </summary>

private static System.UInt32 m\_EMS = 0;

/// <summary>

/// Result Message

/// </summary>

private static System.String m\_RSM = string.Empty;

/// <summary>

/// Reply Code

/// </summary>

private static System.Int32 m\_RPC = 0;

/// <summary>

/// TransactionID

/// </summary>

private static System.UInt32 m\_TID = 0;

/// <summary>

/// Result Message Size

/// </summary>

private static System.UInt32 m\_RMS = 0;

#endregion

#region Public Properties

/// <summary>

/// <see cref="String"/>Возвращает название функции (метода) библиотеки выполнявшейся последней (Last Executed Function Name)

/// </summary>

public static String LEF

{

get { return m\_LEF; }

}

/// <summary>

/// <see cref="Int32"/>Возвращает Расширенный код ошибки (Extended Error Code)

/// </summary>

public static Int32 EEC

{

get { return m\_EEC; }

}

/// <summary>

/// Возвращает сообщение об ошибке (Error Message)

/// </summary>

public static String ERM

{

get { return m\_ERM; }

}

/// <summary>

/// <see cref="String"/>Возвращает сообщение описывающее результат выполнения некоторых функций библиотеки (Result Message)

/// </summary>

public static String RSM

{

get { return m\_RSM; }

}

/// <summary>

/// <see cref="Int32"/>Возвращает код результа метода обратного вызова обработки транзакций (Reply Code)

/// </summary>

public static System.Int32 RPC

{

get { return m\_RPC; }

}

/// <summary>

/// <see cref="UInt32"/>Взвращает идентификатор транзакции (Transaction ID)

/// </summary>

public static System.UInt32 TID

{

get { return m\_TID; }

}

#endregion

#region Prototype Method

//-----------------------------------------------------------------

//long TRANS2QUIK\_API \_\_stdcall TRANS2QUIK\_CONNECT (

//LPSTR lpstConnectionParamsString,

//long\* pnExtendedErrorCode,

//LPSTR lpstrErrorMessage,

//DWORD dwErrorMessageSize);

//-----------------------------------------------------------------

/// <summary>

/// Функция для установления связи библиотеки TRANS2QUIK.DLL с терминалом QUIK

/// </summary>

/// <param name="connectionParamsString">Полный путь к каталогу, в котором установлен INFO.EXE, с которым устанавливаем соединениею. Тип <b>String</b></param>

/// <param type="String"></param>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="errorMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

/// <param name="errorMessageSize">Содержит длину строки, на которую ссылается указатель errorMessage</param>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_SUCCESS</term>

/// <description>соединение установлено успешно</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_QUIK\_TERMINAL\_NOT\_FOUND</term>

/// <description>в указанном каталоге либо отсутствует INFO.EXE, либо у него не запущен сервис обработки

/// внешних подключений, в extendedErrorCode в этом случае передается 0</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_VERSION\_NOT\_SUPPORTED</term>

/// <description>используемая версия Trans2QUIK.DLL указанным INFO.EXE не поддерживается,

/// в extendedErrorCode в этом случае передается 0</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_ALREADY\_CONNECTED\_TO\_QUIK</term>

/// <description>соединение уже установлено, в extendedErrorCode в этом случае передается 0</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_FAILED</term>

/// <description>произошла ошибка при установлении соединения, в extendedErrorCode в этом случае

/// передается дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

///

[DllImport("TRANS2QUIK.DLL", EntryPoint = "\_TRANS2QUIK\_CONNECT@16", CallingConvention = CallingConvention.StdCall)]

public static extern System.Int32 CONNECT(

System.String connectionParamsString,

ref System.Int32 extendedErrorCode,

System.String errorMessage,

System.UInt32 errorMessageSize);

// -----------------------------------------------------------------

// long TRANS2QUIK\_API \_\_stdcall TRANS2QUIK\_IS\_DLL\_CONNECTED (

// long\* pnExtendedErrorCode,

// LPSTR lpstrErrorMessage,

// DWORD dwErrorMessageSize);

// -----------------------------------------------------------------

/// <summary>

/// Функция для проверки соединения библиотеки TRANS2QUIK.DLL с терминалом QUIK

/// </summary>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="errorMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

/// <param name="errorMessageSize">Содержит длину строки, на которую ссылается указатель errorMessage</param>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_DLL\_CONNECTED</term>

/// <description>соединение библиотеки TRANS2QUIK.DLL с терминалом QUIK установлено</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_NOT\_CONNECTED</term>

/// <description>не установлена связь библиотеки TRANS2QUIK.DLL с терминалом QUIK</description>

/// </item>

/// </list>

/// </returns>

///

[DllImport("TRANS2QUIK.DLL", EntryPoint = "\_TRANS2QUIK\_IS\_DLL\_CONNECTED@12", CallingConvention = CallingConvention.StdCall)]

public static extern System.Int32 IS\_DLL\_CONNECTED(

ref System.Int32 extendedErrorCode,

System.String errorMessage,

System.UInt32 errorMessageSize);

// -----------------------------------------------------------------

// long TRANS2QUIK\_API \_\_stdcall TRANS2QUIK\_DISCONNECT (

// long\* pnExtendedErrorCode,

// LPSTR lpstrErrorMessage,

// DWORD dwErrorMessageSize);

// -----------------------------------------------------------------

/// <summary>

/// Функция для разрыва соединения библиотеки TRANS2QUIK.DLL с терминалом QUIK

/// </summary>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="errorMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

/// <param name="errorMessageSize">Содержит длину строки, на которую ссылается указатель errorMessage</param>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_SUCCESS</term>

/// <description>соединение библиотеки TRANS2QUIK.DLL с терминалом QUIK разорвано успешно</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_FAILED</term>

/// <description>произошла ошибка при разрыве соединения, в extendedErrorCode в этом случае

/// передается дополнительный код ошибки</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_NOT\_CONNECTED</term>

/// <description>попытка разорвать соединение при не установленной связи. В этом случае в

/// extendedErrorCode может передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

///

[DllImport("TRANS2QUIK.DLL", EntryPoint = "\_TRANS2QUIK\_DISCONNECT@12", CallingConvention = CallingConvention.StdCall)]

public static extern System.Int32 DISCONNECT(

ref System.Int32 extendedErrorCode,

System.String errorMessage,

System.UInt32 errorMessageSize);

// -----------------------------------------------------------------

// long TRANS2QUIK\_API \_\_stdcall TRANS2QUIK\_IS\_QUIK\_CONNECTED (

// long\* pnExtendedErrorCode,

// LPSTR lpstrErrorMessage,

// DWORD dwErrorMessageSize);

// -----------------------------------------------------------------

/// <summary>

/// Функция для проверки соединения терминала QUIK с сервером QUIK

/// </summary>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="errorMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

/// <param name="errorMessageSize">Содержит длину строки, на которую ссылается указатель errorMessage</param>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_QUIK\_CONNECTED</term>

/// <description>соединение установлено</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_QUIK\_NOT\_CONNECTED</term>

/// <description>соединение не установлено</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_NOT\_CONNECTED</term>

/// <description>не установлена связь библиотеки TRANS2QUIK.DLL с терминалом QUIK. В этом случае

/// проверить наличие или отсутствие связи терминала QUIK с сервером невозможно </description>

/// </item>

/// </list>

/// </returns>

///

[DllImport("TRANS2QUIK.DLL", EntryPoint = "\_TRANS2QUIK\_IS\_QUIK\_CONNECTED@12", CallingConvention = CallingConvention.StdCall)]

public static extern System.Int32 IS\_QUIK\_CONNECTED(

ref System.Int32 extendedErrorCode,

System.String errorMessage,

System.UInt32 errorMessageSize);

// -----------------------------------------------------------------

// long TRANS2QUIK\_API \_\_stdcall TRANS2QUIK\_SEND\_SYNC\_TRANSACTION (

// LPSTR lpstTransactionString,

// long\* pnReplyCode,

// PDWORD pdwTransId,

// double\* pdOrderNum,

// LPSTR lpstrResultMessage,

// DWORD dwResultMessageSize,

// long\* pnExtendedErrorCode,

// LPSTR lpstErrorMessage,

// DWORD dwErrorMessageSize);

// -----------------------------------------------------------------

/// <summary>

/// Функция для отправки синхронной транзакции

/// </summary>

/// <param name="transactionString">Строка с описанием транзакции. Формат строки тот же самый, что и при

/// отправке транзакций через файл</param>

/// <param name="replyCode">Получает статус выполнения транзакции. Значения статусов те же самые, что и

/// при подаче заявок через файл</param>

/// <param name="transactionID">Получает значение TransID транзакции, указанной пользователем.</param>

/// <param name="orderNumber">В случае успеха получает номер заявки в торговой системе</param>

/// <param name="resultMessage">В случае успеха содержит сообщение торговой системы</param>

/// <param name="resultMessageSize">Содержит длину строки, на которую ссылается указатель resultMessage</param>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="errorMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

/// <param name="errorMessageSize">Содержит длину строки, на которую ссылается указатель errorMessage</param>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_SUCCESS</term>

/// <description>транзакция отправлена успешно</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_WRONG\_SYNTAX</term>

/// <description>строка транзакции заполнена неверно</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между библиотекой TRANS2QUIK.DLL и терминалом QUIK</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_QUIK\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между терминалом QUIK и сервером</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_FAILED</term>

/// <description>транзакцию отправить не удалось. В переменную extendedErrorCode в этом случае может

/// передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

///

[DllImport("TRANS2QUIK.DLL", EntryPoint = "\_TRANS2QUIK\_SEND\_SYNC\_TRANSACTION@36", CallingConvention = CallingConvention.StdCall)]

public static extern System.Int32 SEND\_SYNC\_TRANSACTION(

System.String transactionString,

ref System.Int32 replyCode,

ref System.UInt32 transactionID,

ref System.Double orderNumber,

System.String resultMessage,

System.UInt32 resultMessageSize,

ref System.Int32 extendedErrorCode,

System.String errorMessage,

System.UInt32 errorMessageSize);

// -----------------------------------------------------------------

// long TRANS2QUIK\_API \_\_stdcall TRANS2QUIK\_SEND\_ASYNC\_TRANSACTION (

// LPSTR lpstTransactionString,

// long\* pnExtendedErrorCode,

// LPSTR lpstErrorMessage,

// DWORD dwErrorMessageSize);

// -----------------------------------------------------------------

/// <summary>

/// Функция для отправки асинхронной транзакции

/// </summary>

/// <param name="transactionString">Строка с описанием транзакции. Формат строки такой же, что и при

/// отправке транзакций через файл</param>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="errorMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

/// <param name="errorMessageSize">Содержит длину строки, на которую ссылается указатель errorMessage</param>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_SUCCESS</term>

/// <description>транзакция отправлена успешно</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_WRONG\_SYNTAX</term>

/// <description>строка транзакции заполнена неверно</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между библиотекой TRANS2QUIK.DLL и терминалом QUIK</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_QUIK\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между терминалом квик и сервером</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_FAILED</term>

/// <description>транзакцию отправить не удалось. В переменную extendedErrorCode в этом случае может

/// передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

///

[DllImport("TRANS2QUIK.DLL", EntryPoint = "\_TRANS2QUIK\_SEND\_ASYNC\_TRANSACTION@16", CallingConvention = CallingConvention.StdCall)]

public static extern System.Int32 SEND\_ASYNC\_TRANSACTION(

System.String transactionString,

ref System.Int32 extendedErrorCode,

System.String errorMessage,

System.UInt32 errorMessageSize);

// -----------------------------------------------------------------

// void TRANS2QUIK\_CONNECTION\_STATUS\_CALLBACK (

// long nConnectionEvent,

// long nExtendedErrorCode,

// LPSTR lpstrInfoMessage)

// -----------------------------------------------------------------

/// <summary>

/// Описывает прототип функции обратного вызова для контроля за

/// состоянием соединения между TRANS2QUIK.DLL и используемым терминалом

/// QUIK и между используемым терминалом QUIK и сервером

/// </summary>

/// <param name="connectionEvent">Число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_QUIK\_CONNECTED</term>

/// <description>соединение между терминалом QUIK и сервером установлено</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_QUIK\_DISCONNECTED</term>

/// <description>соединение между терминалом QUIK и сервером разорвано</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_CONNECTED</term>

/// <description>соединение между DLL и используемым терминалом QUIK установлено</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_DISCONNECTED</term>

/// <description>соединение между DLL и используемым терминалом QUIK разорвано</description>

/// </item>

/// </list>

/// </param>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="infoMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

///

public delegate void CONNECTION\_STATUS\_CALLBACK(

System.Int32 connectionEvent,

System.Int32 extendedErrorCode,

System.String infoMessage);

// -----------------------------------------------------------------

// long TRANS2QUIK\_API \_\_stdcall TRANS2QUIK\_SET\_CONNECTION\_STATUS\_CALLBACK (

// TRANS2QUIK\_CONNECTION\_STATUS\_CALLBACK pfConnectionStatusCallback,

// long\* pnExtendedErrorCode,

// LPSTR lpstrErrorMessage,

// DWORD dwErrorMessageSize);

// -----------------------------------------------------------------

/// <summary>

/// Функция устанавливает функцию обратного вызова ConnectionStatusCallBack

/// </summary>

/// <param name="connectionStatusCallBack">Ссылается на функцию, которая будет обрабатывать информацию

/// о состоянии связи библиотеки TRANS2QUIK.DLL с терминалом QUIK или терминала QUIK с сервером</param>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="errorMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

/// <param name="errorMessageSize">Содержит длину строки, на которую ссылается указатель errorMessage</param>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_SUCCESS</term>

/// <description>функция обратного вызова установлена</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_FAILED</term>

/// <description>функцию обратного вызова установить не удалось. В этом случае в переменную

/// extendedErrorCode может передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

///

[DllImport("TRANS2QUIK.DLL", EntryPoint = "\_TRANS2QUIK\_SET\_CONNECTION\_STATUS\_CALLBACK@16", CallingConvention = CallingConvention.StdCall)]

public static extern System.Int32 SET\_CONNECTION\_STATUS\_CALLBACK(

CONNECTION\_STATUS\_CALLBACK connectionStatusCallBack,

System.Int32 extendedErrorCode,

System.String errorMessage,

System.UInt32 errorMessageSize);

// -----------------------------------------------------------------

// void TRANS2QUIK\_TRANSACTION\_REPLY\_CALLBACK(

// long nTransactionResult,

// long nTransactionExtendedErrorCode,

// long nTransactionReplyCode,

// DWORD dwTransId,

// double dOrderNum,

// LPSTR lpstrTransactionReplyMessage)

// -----------------------------------------------------------------

/// <summary>

/// Описывает прототип функции обратного вызова для обработки полученной информации об отправленной

/// транзакции

/// </summary>

/// <param name="transactionResult">

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_SUCCESS</term>

/// <description>транзакция передана успешно</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_WRONG\_SYNTAX</term>

/// <description>строка транзакции заполнена неверно</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_DLL\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между библиотекой TRANS2QUIK.DLL и терминалом QUIK</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_QUIK\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между терминалом квик и сервером</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_FAILED</term>

/// <description>транзакцию передать не удалось. В этом случае в переменную extendedErrorCode может

/// передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </param>

/// <param name="transactionExtendedErrorCode">В случае возникновения проблемы при выходе из функции

/// обратного вызова в переменную может быть помещен расширенный код ошибки</param>

/// <param name="transactionReplyCode">Cтатус выполнения транзакции. Значения

/// статусов те же самые, что и при подаче заявок через файл</param>

/// <param name="transactionID">Содержимое параметра TransId, который получила зарегестрированная транзакция</param>

/// <param name="orderNumber">Номер транзакции, присвоенный торговой системой</param>

/// <param name="transactionReplyMessage">Сообщение от торговой системы или сервера QUIK</param>

///

public delegate void TRANSACTION\_REPLY\_CALLBACK(

System.Int32 transactionResult,

System.Int32 transactionExtendedErrorCode,

System.Int32 transactionReplyCode,

System.UInt32 transactionID,

System.Double orderNumber,

System.String transactionReplyMessage);

// -----------------------------------------------------------------

// long TRANS2QUIK\_API \_\_stdcall TRANS2QUIK\_SET\_TRANSACTIONS\_REPLY\_CALLBACK (

// TRANS2QUIK\_TRANSACTION\_REPLY\_CALLBACK pfTransactionReplyCallback,

// long\* pnExtendedErrorCode,

// LPSTR lpstrErrorMessage,

// DWORD dwErrorMessageSize);

// -----------------------------------------------------------------

/// <summary>

/// Функция устанавливает функцию обратного вызова TransactionReplyCallBack

/// </summary>

/// <param name="transactionReplyCallBack">Функция, которая будет обрабатывать информацию об отправленной транзакции</param>

/// <param name="extendedErrorCode">В случае возникновения ошибки может содержать расширенный код ошибки</param>

/// <param name="errorMessage">В случае возникновения ошибки может получать сообщение о возникшей ошибке</param>

/// <param name="errorMessageSize">Содержит длину строки, на которую ссылается указатель errorMessage</param>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>TRANS2QUIK\_SUCCESS</term>

/// <description>функция обратного вызова установлена</description>

/// </item>

/// <item>

/// <term>TRANS2QUIK\_FAILED</term>

/// <description>функцию обратного вызова установить не удалось. В этом случае в переменную

/// extendedErrorCode может передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

///

[DllImport("TRANS2QUIK.DLL", EntryPoint = "\_TRANS2QUIK\_SET\_TRANSACTIONS\_REPLY\_CALLBACK@16", CallingConvention = CallingConvention.StdCall)]

public static extern System.Int32 SET\_TRANSACTIONS\_REPLY\_CALLBACK(

TRANSACTION\_REPLY\_CALLBACK transactionReplyCallBack,

System.Int32 extendedErrorCode,

System.String errorMessage,

System.UInt32 errorMessageSize);

#endregion

#region Public Method

/// <summary>

/// Устанавливает связь между библиотекой TRANS2QUIK.DLL с терминалом QUIK

/// </summary>

/// <param name="thePathToQUIK">Полный путь к терминалу QUIK</param>

/// <returns>Возвращает значение типа <b>eQUIKResult</b> характеризующее результат. Может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>SUCCESS</term>

/// <description>соединение установлено успешно</description>

/// </item>

/// <item>

/// <term>QUIK\_TERMINAL\_NOT\_FOUND</term>

/// <description>в указанном каталоге либо отсутствует INFO.EXE, либо у него не запущен сервис обработки

/// внешних подключений, в extendedErrorCode в этом случае передается 0</description>

/// </item>

/// <item>

/// <term>DLL\_VERSION\_NOT\_SUPPORTED</term>

/// <description>используемая версия Trans2QUIK.DLL указанным INFO.EXE не поддерживается,

/// в extendedErrorCode в этом случае передается 0</description>

/// </item>

/// <item>

/// <term>ALREADY\_CONNECTED\_TO\_QUIK</term>

/// <description>соединение уже установлено, в extendedErrorCode в этом случае передается 0</description>

/// </item>

/// <item>

/// <term>FAILED</term>

/// <description>произошла ошибка при установлении соединения, в [EEC] в этом случае

/// передается дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

///

public static eQUIKResult Connect(string thePathToQUIK)

{

// Locals

eQUIKResult result = eQUIKResult.UNKNOWN;

//

m\_LEF = "TRANS2QUIKSHARP.QUIKAPI.CONNECT";

result = (eQUIKResult)CONNECT(thePathToQUIK, ref m\_EEC, m\_ERM, m\_EMS);

//

return result;

}

/// <summary>

/// Разрывает связь между библиотекой TRANS2QUIK.DLL с терминалом QUIK

/// </summary>

/// <returns>Возвращает значение типа <b>eQUIKResult</b> характеризующее результат. Может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>SUCCESS</term>

/// <description>соединение библиотеки TRANS2QUIK.DLL с терминалом QUIK разорвано успешно</description>

/// </item>

/// <item>

/// <term>DLL\_NOT\_CONNECTED</term>

/// <description>попытка разорвать соединение при не установленной связи. В этом случае в

/// [EEC] может передаваться дополнительный код ошибки</description>

/// </item>

/// <item>

/// <term>FAILED</term>

/// <description>произошла ошибка при разрыве соединения. В этом случае в

/// [EEC] передается дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

///

public static eQUIKResult Disconnect()

{

// Locals

eQUIKResult result = eQUIKResult.UNKNOWN;

//

m\_LEF = "TRANS2QUIKSHARP.QUIKAPI.DISCONNECT";

result = (eQUIKResult)DISCONNECT(ref m\_EEC, m\_ERM, m\_EMS);

//

return result;

}

/// <summary>

/// Проверят соединение библиотеки TRANS2QUIK.DLL с терминалом QUIK

/// </summary>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term><see cref="TRANS2QUIKSHARP.eQUIKResult"/>.DLL\_CONNECTED</term>

/// <description>соединение библиотеки TRANS2QUIK.DLL с терминалом QUIK установлено</description>

/// </item>

/// <item>

/// <term><see cref="TRANS2QUIKSHARP.eQUIKResult"/>.DLL\_NOT\_CONNECTED</term>

/// <description>соединение библиотеки TRANS2QUIK.DLL с терминалом QUIK не установлено</description>

/// </item>

/// </list>

/// </returns>

public static eQUIKResult TestConnectionDLL()

{

// Locals

eQUIKResult result = eQUIKResult.UNKNOWN;

//

m\_LEF = "TRANS2QUIKSHARP.QUIKAPI.IS\_DLL\_CONNECTED";

result = (eQUIKResult)IS\_DLL\_CONNECTED(ref m\_EEC, m\_ERM, m\_EMS);

//

return result;

}

/// <summary>

/// Проверяет соединение терминала QUIK с сервером QUIK

/// </summary>

/// <returns>Возвращаемое число может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term><see cref="TRANS2QUIKSHARP.eQUIKResult"/>.QUIK\_CONNECTED</term>

/// <description>соединение терминала QUIK с сервером QUIK установлено</description>

/// </item>

/// <item>

/// <term><see cref="TRANS2QUIKSHARP.eQUIKResult"/>.QUIK\_NOT\_CONNECTED</term>

/// <description>соединение терминала QUIK с сервером QUIK не установлено</description>

/// </item>

/// <item>

/// <term><see cref="TRANS2QUIKSHARP.eQUIKResult"/>.DLL\_NOT\_CONNECTED</term>

/// <description>не установлена связь библиотеки TRANS2QUIK.DLL с терминалом QUIK. В этом случае

/// проверить наличие или отсутствие связи терминала QUIK с сервером невозможно </description>

/// </item>

/// </list>

/// </returns>

public static eQUIKResult TestConnectionQUIK()

{

// Locals

eQUIKResult result = eQUIKResult.UNKNOWN;

//

m\_LEF = "TRANS2QUIKSHARP.QUIKAPI.IS\_QUIK\_CONNECTED";

result = (eQUIKResult)IS\_QUIK\_CONNECTED(ref m\_EEC, m\_ERM, m\_EMS);

//

return result;

}

/// <summary>

/// Отправляет транзакции в синхронном режиме

/// </summary>

/// <param name="theTransactionDescription">Строка с описанием транзакции</param>

/// <param name="theOrderNumber">Если результат выполнения функции <b>SUCCESS</b>содержит номер заявки в торговой системе</param>

/// <returns>Возвращает значение типа <b>eQUIKResult</b> характеризующее результат. Может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>SUCCESS</term>

/// <description>транзакция отправлена успешно</description>

/// </item>

/// <item>

/// <term>WRONG\_SYNTAX</term>

/// <description>строка транзакции заполнена неверно</description>

/// </item>

/// <item>

/// <term>DLL\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между библиотекой TRANS2QUIK.DLL и терминалом QUIK</description>

/// </item>

/// <item>

/// <term>QUIK\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между терминалом QUIK и сервером</description>

/// </item>

/// <item>

/// <term>FAILED</term>

/// <description>транзакцию отправить не удалось. В переменную extendedErrorCode в этом случае может

/// передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

public static eQUIKResult SendSyncTransaction(string theTransactionDescription, double theOrderNumber)

{

// Locals

eQUIKResult result = eQUIKResult.UNKNOWN;

//

m\_LEF = "TRANS2QUIKSHARP.QUIKAPI.SEND\_SYNC\_TRANSACTION";

result = (eQUIKResult)SEND\_SYNC\_TRANSACTION(theTransactionDescription, ref m\_RPC, ref m\_TID,

ref theOrderNumber, m\_RSM, m\_RMS, ref m\_EEC, m\_ERM, m\_EMS);

//

return result;

}

/// <summary>

/// Отправляет транзакции в асинхронном режиме

/// </summary>

/// <param name="theTransactionDescription">Строка с описанием транзакции</param>

///

/// <returns>Возвращает значение типа <b>eQUIKResult</b> характеризующее результат. Может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>SUCCESS</term>

/// <description>транзакция отправлена успешно</description>

/// </item>

/// <item>

/// <term>WRONG\_SYNTAX</term>

/// <description>строка транзакции заполнена неверно</description>

/// </item>

/// <item>

/// <term>DLL\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между библиотекой TRANS2QUIK.DLL и терминалом QUIK</description>

/// </item>

/// <item>

/// <term>QUIK\_NOT\_CONNECTED</term>

/// <description>отсутствует соединение между терминалом квик и сервером</description>

/// </item>

/// <item>

/// <term>FAILED</term>

/// <description>транзакцию отправить не удалось. В переменную extendedErrorCode в этом случае может

/// передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

public static eQUIKResult SendAsyncTransaction(string theTransactionDescription)

{

// Locals

eQUIKResult result = eQUIKResult.UNKNOWN;

//

m\_LEF = "TRANS2QUIKSHARP.QUIKAPI.SEND\_ASYNC\_TRANSACTION";

result = (eQUIKResult)SEND\_ASYNC\_TRANSACTION(theTransactionDescription, ref m\_EEC, m\_ERM, m\_EMS);

//

return result;

}

/// <summary>

/// Устанавливает функцию обратного вызова ConnectionStatusCallBack

/// </summary>

/// <param name="theConnectionStatusCallBack">Функция, которая будет обрабатывать информацию

/// о состоянии связи библиотеки TRANS2QUIK.DLL с терминалом QUIK или терминала QUIK с сервером</param>

/// <returns>Возвращает значение типа <b>eQUIKResult</b> характеризующее результат. Может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>SUCCESS</term>

/// <description>функция обратного вызова установлена</description>

/// </item>

/// <item>

/// <term>FAILED</term>

/// <description>функцию обратного вызова установить не удалось. В этом случае в переменную

/// extendedErrorCode может передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

public static eQUIKResult SetConnectionStatusDelegate(CONNECTION\_STATUS\_CALLBACK theConnectionStatusCallBack)

{

// Locals

eQUIKResult result = eQUIKResult.UNKNOWN;

//

m\_LEF = "TRANS2QUIKSHARP.QUIKAPI.SET\_CONNECTION\_STATUS\_CALLBACK";

result = (eQUIKResult)SET\_CONNECTION\_STATUS\_CALLBACK(theConnectionStatusCallBack, m\_EEC, m\_ERM, m\_EMS);

//

return result;

}

/// <summary>

/// Устанавливает функцию обратного вызова TransactionReplyCallBack

/// </summary>

/// <param name="theTransactionReplyCallBack">Функция, которая будет обрабатывать информацию об отправленной транзакции</param>

/// <returns>Возвращает значение типа <b>eQUIKResult</b> характеризующее результат. Может принимать следующие значения:

/// <list type ="table">

/// <item>

/// <term>SUCCESS</term>

/// <description>функция обратного вызова установлена</description>

/// </item>

/// <item>

/// <term>FAILED</term>

/// <description>функцию обратного вызова установить не удалось. В этом случае в переменную

/// extendedErrorCode может передаваться дополнительный код ошибки</description>

/// </item>

/// </list>

/// </returns>

public static eQUIKResult SetTransactionReplyDelegate(TRANSACTION\_REPLY\_CALLBACK theTransactionReplyCallBack)

{

// Locals

eQUIKResult result = eQUIKResult.UNKNOWN;

//

m\_LEF = "TRANS2QUIKSHARP.QUIKAPI.SET\_TRANSACTIONS\_REPLY\_CALLBACK";

result = (eQUIKResult)SET\_TRANSACTIONS\_REPLY\_CALLBACK(theTransactionReplyCallBack, m\_EEC, m\_ERM, m\_EMS);

//

return result;

}

#endregion

}

}

**Листинг модуля для работы с форматом XlTable:**

using System.IO;

using System.Text;

namespace XlDde

{

public class XlTable

{

public enum BlockType

{

Table = 0x10,

Float = 0x01,

String = 0x02,

Bool = 0x03,

Error = 0x04,

Blank = 0x05,

Int = 0x06,

Skip = 0x07,

Unknown = 0x10000,

Bad = 0x10001

}

const int codepage = 1251;

const int wsize = 2;

const int fsize = 8;

const int hsize = wsize \* 2;

byte[] data;

MemoryStream ms;

BinaryReader br;

int blocksize;

public int Rows { get; protected set; }

public int Columns { get; protected set; }

public BlockType ValueType { get; protected set; }

public double FloatValue { get; protected set; }

public string StringValue { get; protected set; }

public ushort WValue { get; protected set; }

public XlTable(byte[] data)

{

this.data = data;

ms = new MemoryStream(data);

br = new BinaryReader(ms, Encoding.ASCII);

if(data.Length < wsize \* 4 || (BlockType)br.ReadUInt16() != BlockType.Table)

SetBadDataStatus();

ms.Seek(wsize, SeekOrigin.Current);

Rows = br.ReadUInt16();

Columns = br.ReadUInt16();

ValueType = BlockType.Unknown;

}

void SetBadDataStatus()

{

ValueType = BlockType.Bad;

blocksize = 1;

}

public void ReadValue()

{

if(ValueType == BlockType.Unknown)

{

if(ms.Position + hsize > data.Length)

SetBadDataStatus();

else

{

ValueType = (BlockType)br.ReadUInt16();

blocksize = br.ReadUInt16();

if(ms.Position + blocksize > data.Length)

SetBadDataStatus();

}

}

if(blocksize > 0)

switch(ValueType)

{

case BlockType.Float:

blocksize -= fsize;

if(blocksize >= 0)

FloatValue = br.ReadDouble();

else

SetBadDataStatus();

break;

case BlockType.String:

int strlen = ms.ReadByte();

blocksize -= strlen + 1;

if(blocksize >= 0)

{

StringValue = Encoding.GetEncoding(codepage).GetString(data, (int)ms.Position, strlen);

br.BaseStream.Seek(strlen, SeekOrigin.Current);

}

else

SetBadDataStatus();

break;

case BlockType.Bool:

case BlockType.Error:

case BlockType.Blank:

case BlockType.Int:

case BlockType.Skip:

blocksize -= wsize;

if(blocksize >= 0)

WValue = br.ReadUInt16();

else

SetBadDataStatus();

break;

default:

SetBadDataStatus();

break;

}

else

{

ValueType = BlockType.Unknown;

ReadValue();

}

}

}

}

**Листинг модуля XlDdeServer:**

using System;

using System.Collections.Generic;

using NDde.Server;

namespace XlDde

{

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* XlDdeChannel class \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

abstract class XlDdeChannel

{

public virtual bool IsConnected { get; set; }

public DateTime DataReceived { get; protected set; }

public bool IsError { get; protected set; }

public void ResetError() { IsError = false; }

public void PutDdeData(byte[] data)

{

DataReceived = DateTime.Now;

ProcessTable(new XlTable(data));

}

protected abstract void ProcessTable(XlTable xt);

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* XlDdeServer class \*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

class XlDdeServer : DdeServer

{

Dictionary<string, XlDdeChannel> channels;

// --------------------------------------------------------------

public XlDdeServer(string service)

: base(service)

{

channels = new Dictionary<string, XlDdeChannel>();

}

// --------------------------------------------------------------

public void AddChannel(string topic, XlDdeChannel channel)

{

channels.Add(topic, channel);

}

// --------------------------------------------------------------

protected override bool OnBeforeConnect(string topic)

{

return channels.ContainsKey(topic);

}

// --------------------------------------------------------------

protected override void OnAfterConnect(DdeConversation c)

{

XlDdeChannel channel = channels[c.Topic];

c.Tag = channel;

channel.IsConnected = true;

}

// --------------------------------------------------------------

protected override void OnDisconnect(DdeConversation c)

{

((XlDdeChannel)c.Tag).IsConnected = false;

}

// --------------------------------------------------------------

protected override PokeResult OnPoke(DdeConversation c, string item, byte[] data, int format)

{

//if(format != xlTableFormat)

// return PokeResult.NotProcessed;

((XlDdeChannel)c.Tag).PutDdeData(data);

return PokeResult.Processed;

}

// --------------------------------------------------------------

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}