Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ _____
подпись инициалы, фамилия
« _____» _____ 20 ___ г.

ДИПЛОМНЫЙ ПРОЕКТ

230101.65 «Вычислительные машины, комплексы, системы и сети»
код и наименование специальности

Разработка ПО информационно-развлекательного терминала на базе ОС Windows
тема

Пояснительная записка

Руководитель _____ доцент, к.т.н. Д.А. Швец
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ ЗКИ 10-01 780702219 К.В. Дрючин
подпись, дата номер группы номер зачетной книжки инициалы, фамилия

Нормоконтролер _____ доцент, к.т.н. В.И. Иванов
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный уровень развития технологий сделал неразрывными большинство сфер нашей жизни и телекоммуникации. Компьютеризация способствует переведению на автоматический уровень многих повседневных задач, облегчает доступ к информации, что в наш информационный век является наиболее значимо, а так же позволяет быстро и интересно организовать досуг. Как следствие этого всем видам бизнеса – от малого до крупного, приходится подстраиваться под текущие тенденции и отвечать всем современным требованиям. Сегодня каждая компания старается обеспечить своим клиентам самые комфортные условия, например: в кафе и ресторанах присутствуют Wi-Fi точки доступа, обеспечивающие быстрый и бесплатный доступ к сети интернет, в автомойках и библиотеках можно сесть за ПК и получить доступ как к сети интернет, так и к некоторым играм.

В процессе обслуживания IT парка автомоечных комплексов «25 часов» было выявлено несколько проблем связанных с администрированием терминалов, предоставляющих клиентам доступ в сеть интернет, а именно:

1. Различные операционные системы, установленные на клиентские терминалы: от Windows XP Professional до Windows 7 Professional. Основная масса Windows 7 Starter, возможности которой очень ограничены с позиции системного администрирования.

2. Отсутствие какого-то ни было удаленного управления клиентскими терминалами, вследствие чего, например, приходится выезжать на обслуживаемую точку для обновления информационно-рекламных изображений. Наличие у клиентов легкого доступа к некоторым настройкам операционной системы, что приводило как к появлению

нежелательного ПО на рабочей станции, так и некорректной работе некоторых приложений.

Попытки найти свободно распространяемое программное обеспечение, которое решало бы все эти проблемы на платформе ОС Windows не увенчались успехом. Платное ПО либо не решало всех поставленных задач, либо создавало новые проблемы, решение некоторых из которых не представляется возможным без вливания еще больших средств. Переход на ОС семейства Linux был отклонен не только из-за некорректной поддержки некоторого оборудования, но и в связи с отсутствием некоторого ПО, необходимого для работы автомоечного комплекса, как например ЕГАИС.

Поэтому руководством была поставлена задача внедрить программное решение, которое будет достигать следующие цели:

1. Упростить администрирование ПК в клиентском зале
2. Свести к минимуму финансовые затраты при достижении поставленной задачи

1 Техническое задание

Разработать ПО для информационно-развлекательного терминала на базе ОС Windows. Программный комплекс должен включать в себя 3 различных модуля:

1.1 Клиентский модуль

Функционал доступный пользователю.

1) Запуск браузера
2) Запуск игр
3) Отображение текущего времени
4) Копирование при загрузке ПК рекламных изображений с указанного сервера

Функционал доступный системному администратору.

1) Открытие «Панели управления»
2) Запуск настроек экрана
3) Кнопка «Свернуть»
4) Запуск «COMODO»
5) Запуск «Explorer»
6) Вызов командной строки от имени администратора
7) Запуск «Диспетчера задач»
8) Выполнить выход из системы
9) Выполнить перезагрузку
10) Выполнение удаленных команд от имени администратора

1.2 Модуль «Updater»

1) Закрытие клиентской части
2) Удаление клиентской части
3) Копирование обновленной клиентской части
4) Запуск обновленной клиентской части

1.3 Серверный модуль

1) Поиск клиентских ПК.
2) Возможность ручного ввода сканируемого диапазона IP-адресов
3) Редактирование полученного списка IP-адресов, на которых запущен клиентский модуль
4) Сохранение и загрузка списков
5) Возможность ручного ввода порта, по которому будет идти обмен данными
6) Отображение IP-адреса ПК, с которого клиентский модуль
7) Команды для клиентского модуля.
8) Возможность быстрого ввода команд из заранее подготовленного списка.
9) Возможность ручного ввода команд

Лог действий.

1) Отображение как совершенных действий, так и ответов ПК с установленными клиентскими модулями
2) Возможность очистки списка событий
3) Сохранение списка событий

2 Анализ технического задания

2.1 Особенности

При разработке ПО необходимо учесть следующие особенности, присутствующие в работе на сети автомоечных комплексов «25 часов»:

1) Отсутствие выделенного сервера для одновременного управления всеми точками

2) Возможность управлять клиентскими ПК с любого ПК в зале ожидания клиентов

3) Защищенность передаваемых данных по сети при помощи шифрования

4) Возможность редактирования настроек клиентского модуля через *.cfg файл

5) Клиентский модуль должен ограничивать количество одновременно запускаемых приложений

6) Клиентский модуль должен ограничивать запускаемые приложения согласно подготовленному списку

7) Клиентский модуль должен закрывать браузер при простое ПК более 2-х минут

8) Предоставление клиентским модулем функций администратора после ввода пароля

Взаимодействие с ПО представлено на UML диаграмме, на рисунке 1.

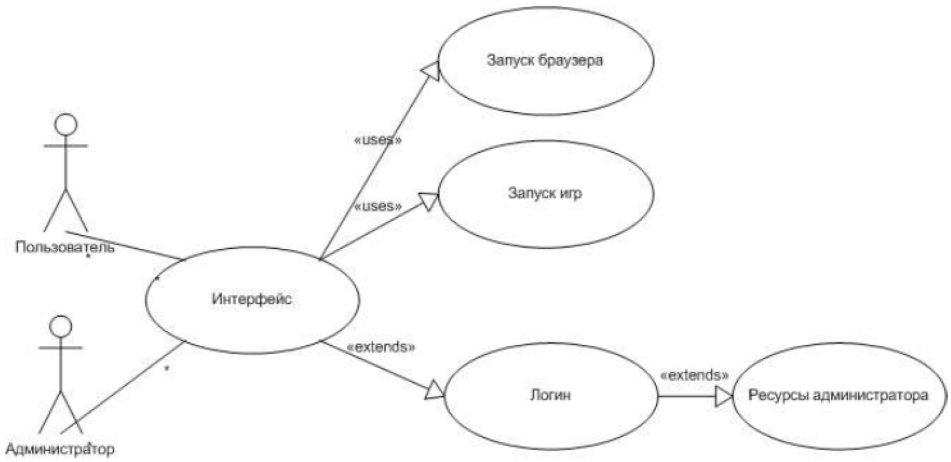


Рисунок 1 – Взаимодействие с ПО

Общая структурная схема работы всех модулей изображена на рисунке 2.

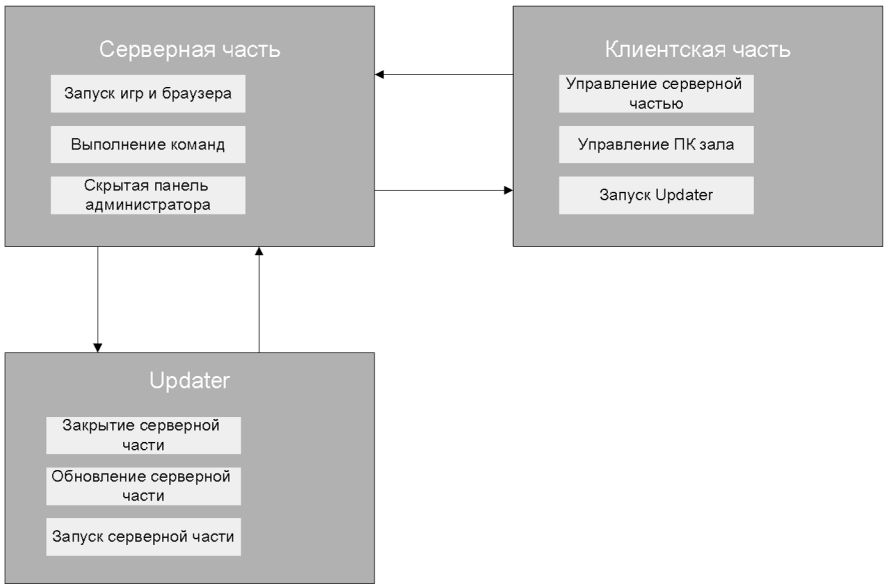


Рисунок 2 – Структурная схема ПО

2.2 Альтернативные решения

Перед созданием собственного ПО информационно-развлекательного терминала на базе ОС Windows были изучены альтернативные варианты.

2.2.1 Тонкие клиенты на базе ОС Linux

Плюсы:

1) Настройка нового рабочего места требует минимально короткий промежуток времени

2) При отключении питания на клиентской машине сессия остается активной на сервере

3) Замена тонкого клиента не вызывает проблем и производится очень быстро

4) Отсутствие вирусных атак

5) Минимальное энергопотребление

6) Простота обслуживания

7) Обмен данными только с сервером

Минусы:

1) Необходимость приобретения отдельного сервера

2) При выходе из строя сети или сервера рабочие места полностью теряют свой функционал

3) Медленная буферизация и «фризы» роликов с медиа-порталов

4) Отсутствие необходимого ПО на семейство Linux

Приблизительные затраты на данную реализацию при покупке сервера: 50 000 руб.[6]

2.2.2 Тонкие клиенты на базе ОС Windows

Плюсы:

1) Настройка нового рабочего места требует минимально короткий промежуток времени

2) При отключении питания на клиентской машине сессия остается активной на сервере

3) Замена тонкого клиента не вызывает проблем и производится очень быстро

4) Защита от вирусных атак только сервера

5) Минимальное энергопотребление

6) Простота обслуживания

7) Обмен данными только с сервером

Минусы:

1) Необходимость приобретения отдельного сервера

2) Необходимость оплаты ОС и лицензий на подключение к Windows-серверу

3) При выходе из строя сети или сервера рабочие места полностью теряют свой функционал

4) Медленная буферизация и «фризы» роликов с медиа-порталов

5) Отсутствие необходимого ПО на семейство Linux

Приблизительные затраты на данную реализацию из расчета лицензий на WTWare и выделенный сервер:

110 клиентских ПК по 350 рублей – 38 500 руб.[7]

Microsoft Windows Server Standard 2012 R2 – 43 240 руб.[8]

Сервер – 50 000 руб.[6]

2.2.3 Использование групповых политик и централизованное управление с использованием контроллера домена и AD

Плюсы:

1) Полностью централизованное управление всем парком ПК

2) Все необходимое ПО корректно работает

3) Полноценное удаленное управление каждой рабочей станцией

Минусы:

1) Необходимость приобретения отдельного сервера

2) Необходимость приобретения лицензионных версий ОС Windows редакции Professional

3) Необходимость обновления аппаратного обеспечения некоторых рабочих станций

Приблизительные затраты на данную реализацию из расчета лицензий на Windows 10 Pro и выделенный сервер:

110 клиентских ПК по 9 817.57руб. – 1 079 933 руб.[9]

Microsoft Windows Server Standard 2012 R2 – 43 240 руб.[8]

Сервер – 50 000 руб.[6]

2.2.4 Использование прочего стороннего ПО

Платное ПО для управлением рабочим местом на примере LogMeIn RemotlyAnywhere:

Плюсы:

1) Легкий удаленный доступ к ПК через браузер

2) Простота обслуживания

Минусы:

1) Слишком большая цена программы

Бесплатное ПО для управления рабочим местом на примере IPGuard

Плюсы:

1) Автоочистка системы от несистемных процессов между сеансами

2) Автоматическое завершение процессов согласно подготовленному списку

Минусы:

1) Отсутствие возможности удаленного исполнения команд

2) Наличие большого количества ненужного функционала, как например «Ведение кассы»

Из вышеперечисленного несложно заметить, что все положительные стороны готовых решений сводятся на нет несколькими крайне серьезными недостатками:

1) Отсутствие возможности запуска некоторого ПО и его управлением

2) Высокая стоимость решения

Приблизительные затраты на данную реализацию из расчета лицензий на LogMeIn RemotlyAnywhere:

110 клиентских ПК по 99$ - приблизительно 707 850 руб.[10]

2.3 Выбор программных продуктов для решения задачи

В итоге для решения поставленной задачи было решено приступить к написанию собственного ПО, внедрение которого позволит достичь указанные цели. Например, в таблице 1 приведено сравнение стоимости различных альтернативных реализаций без учета времени, требуемого на их претворение в жизнь и без учета заработной платы сотрудника компании.

Таблица 1 – Сравнение стоимости альтернативных реализаций

Реализация	Общая стоимость, руб.
Тонкие клиенты на базе ОС Linux	50 000
Тонкие клиенты на базе ОС Windows	131 740

Окончание таблицы 1

Реализация	Общая стоимость, руб.
Использование групповых политик и централизованное управление с использованием контроллера домена и AD	1 173 173
Использование прочего стороннего ПО	707 850
Написание собственного ПО	0

Как система реализации была выбрана IDE Lazarus версии 1.6, так как она предоставляет широкие возможности для программирования приложений ОС Windows, а так же именно на ее стороне мои личные предпочтения. Lazarus - это IDE для создания графических и консольных приложений при помощи компилятора Free Pascal. Free Pascal - это компилятор языков Pascal и Object Pascal, работающий под Windows, Linux, Mac OS X, FreeBSD, и другими ОС. Преимущества Lazarus по сравнению с аналогичными программными продуктами:

1) Быстрота разработки приложения
2) Высокая производительность разработанного приложения
3) Низкие требования разработанного приложения к ресурсам компьютера
4) Наращиваемость за счет встраивания новых компонент и инструментов в среду Lazarus
5) Возможность реализации кроссплатформенности
6) Лицензия: GPL/LGPL

3 Разработка ПО

3.1 Выбор компонентов

Для клиентского модуля программы было решено использовать следующие компоненты:

3.1.1 Компонент BitBtn

Компонент Lazarus BitBtn по сути является компонентом Button, но на нем можно размещать битовый рисунок вместе с текстом этой кнопки. При размещении компонента BitBtn на форме и загрузки в него изображения он принимает следующий вид показанный на рисунке 3.

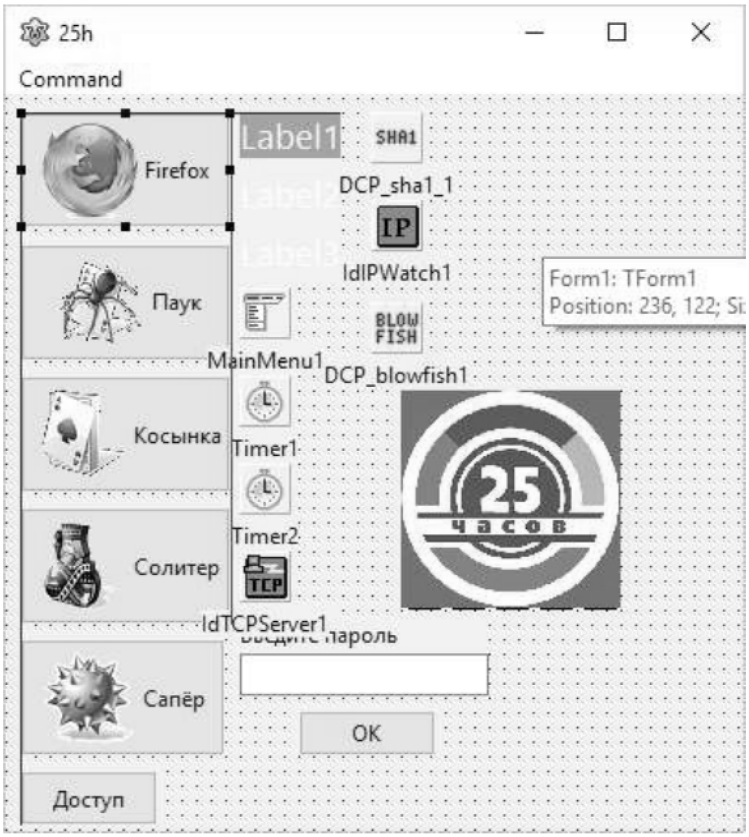


Рисунок 3 – Компонент BitBtn

Для загрузки такого изображения удобно использовать встроенный редактор Picture editor, открыть который можно в свойстве Glyph, нажатием на кнопку с тремя точками. В открывшемся редакторе с помощью кнопки Load можно загрузить изображение формата .bmp. Рисунок может содержать в себе до трех разных изображений. Какое изображение выведется на кнопку, зависит от следующих факторов: первое изображение будет отображаться когда кнопка не нажата; второе изображение будет отображается когда кнопка неактивна т.е свойство enabel равно False; и третье изображение будет отображается при нажатой кнопке (выполнен щелчок). Для того что бы сделать такое изображение необходимо найти 3 подходяших квадратных рисунка например размером 20x20 и с помошью Paint'a(или другого редактора изображений) разместить их на одном полотне размером 60x20 по порядку, затем сохраняем это изображение в формате bmp и загружаем в нашу кнопку. В данном случае компонент используется для запуска игр, входящих в состав ОС Windows, и браузера. Основные свойства представлены в таблице 2.

Таблица 2 – Свойства компонента BitBtn

Свойство	Описание свойства
Name	Имя компонента, используемое для доступа к этому компоненту
Glyph	Свойство, позволяющее привязать изображение к кнопке
Layout	Определяет к какому краю кнопки, прижимается изображение: blGlyphBottom — к нижнему краю; bIGlyphLeft — к левому краю; bIGlyphRight — к правому краю; bIGlyphTop — к верхнему краю.

Окончание таблицы 2

Свойство	Описание свойства
NumGlyphs	Определяет количество изображений.
Spacing	Определяет расстояние от изображения до надписи на кнопке(измеряется в пикселях)
Caption	Текст, выводимый на кнопке
Kind	Это свойство определяет несколько предопределенных видов, при выборе которого на кнопке отображается стандартное изображение
Marign	Определяет расстояние от края кнопки до изображения (измеряется в пикселях)

3.1.2 Компонент Timer

Компонент Lazarus Timer, выделенный на рисунке 4, генерирует последовательность событий timer, этот компонент является не визуальным, то есть во время работы программы он не отображается на форме. Тут он используется для обновления отображения времени и языка ввода.

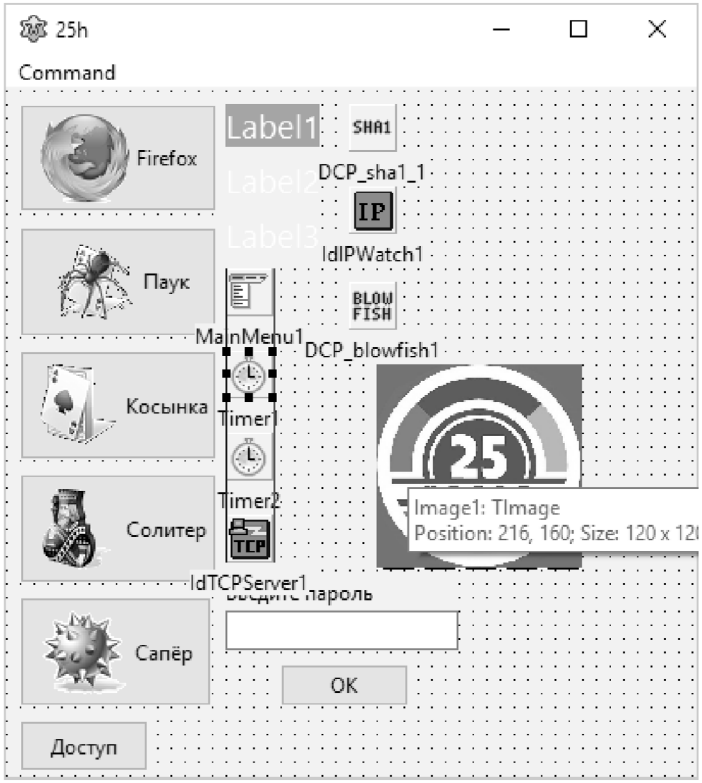


Рисунок 4– Компонент Timer

Timer имеет два необходимых свойства, позволяющие управлять им: Interval (интервал времени в миллисекундах), Enabled — доступность компонента. Так как по умолчанию свойству Enabled присвоено True, то в вашей программе, через промежуток времени установленный в свойстве Interval, сработает timer, т.е. выдаст событие OnTimer, и будут выполнены действия описанные в этом событии. Если компонент необходим, когда запланированные действия должны выполняться не при запуске программы, а в ходе её работы, то в этом случае свойство Enabled устанавливаем в False. Пока Timer1.Enabled равен True, компонент будет продолжать генерировать событие OnTimer через промежуток времени, установленный в свойстве Interval. Основные свойства представлены в таблице 3.

Таблица 3 – Свойства компонента Timer

Свойство	Описание свойства
Name	Имя компонента необходимое для доступу к компоненту и свойствам этого компонента
Interval	Период генерации событий OnTimer, в миллисекундах
Enabled	Разрешает(True) или запрещает(False) генерацию события OnTimer

3.1.3 Компонент Image

Компонент Lazarus Image, выделенный на рисунке 5, предназначен для отображение на форме графических изображений. По умолчанию выводит на поверхность формы изображения, представленные в bmp формате. После размещения на форме компонента Image, он принимает вид выделенной прямоугольной области.

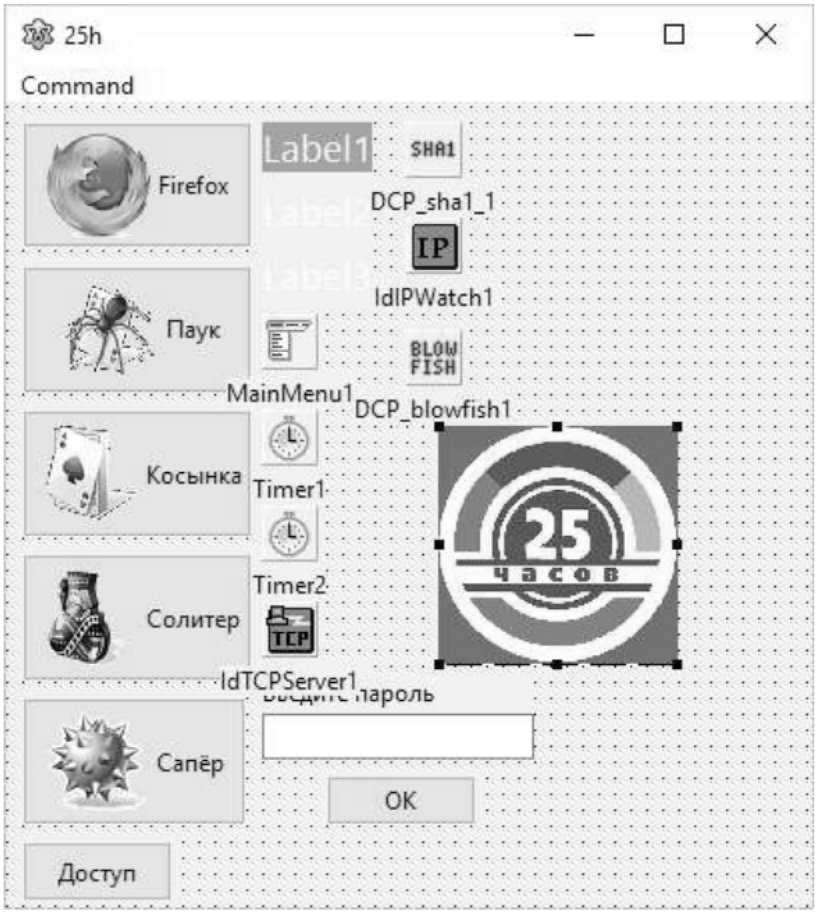


Рисунок 5 – Компонент Image

Соответственно именно в него загружается логотип компании как для рекламных, так и для дизайнерских целей. Основные свойства представлены в таблице 4.

Таблица 4 – Свойства компонента Image

Свойство	Описание свойства
Picture	Изображение, отображающееся в поле компонента
Width, Height	Размеры компонента. Если эти размеры меньше размера иллюстрации, а значение свойств Strech, AutoSize и Proportional равны False, то отображается часть изображения
Proportional	Позволяет автоматически масштабировать картинки без искажения. Для выполнения масштабирования, значение свойства AutoSize должно быть равным False
Strech	Позволяет автоматически масштабировать (сжимать или растягивать) изображение в соответствии с размером компонента Image. Если размер компонента не пропорционален размеру изображения, то изображение будет искажено
AutoSize	Позволяет автоматически изменять размер компонента в соответствии с размером изображения
Center	Позволяет определять расположение изображения в поле компонента Image по горизонтали, если ширина картинки меньше ширины компонента. Если свойства равно False, то изображение прижато к правой границе, если True то изображение располагается по центру
Canvas	Поверхность, позволяющая вывести графику
Transparent	Указывает прозрачный цвет фона изображения

3.1.4 MainMenu

Компонент **MainMenu,** выделенный на рисунке 6, является невизуальным, позволяет конструировать и создавать на форме полосу главного меню, а также сопутствующие выпадающие меню. Компонент MainMenu (рисунок 6) отображает на форме главное меню. При запуске приложения меню будет скрыто, т.к. по ТЗ только системные администраторы имеют к нему доступ. Основные свойства представлены в таблице 5.

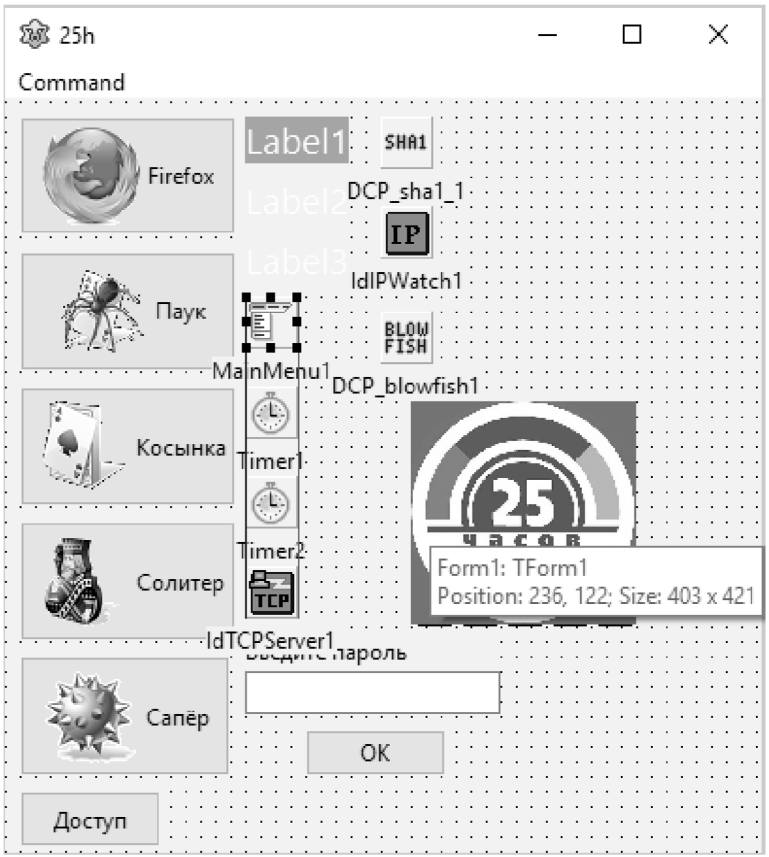


Рисунок 6 – Компонент MainMenu

Таблица 5 – Свойства компонента MainMenu

Свойство	Описание свойства
AutoHotKeys	Определяет, могут ли горячие клавиши элементов меню устанавливаться автоматически.
AutoMerge	Определяет, объединяются ли главные меню вспомогательных форм с главным меню основной формы.
Items	Список элементов меню типа TMenuEtem

3.1.5 IdTCPServer

Компонент, изображенный на рисунке 7, используется для серверной части сетевых приложений. Для его работы необходимо указать Ip-адрес, на котором он будет слушать указанный порт. Все действия выполняются на событии OnExecute, а обмен сообщениями происходит по принципу принял – ответил. Для нашей программы текущий Ip-адрес получаем из компонента IdIPWatch. Основные свойства компонента IdTCPServer представлены в таблице 6.

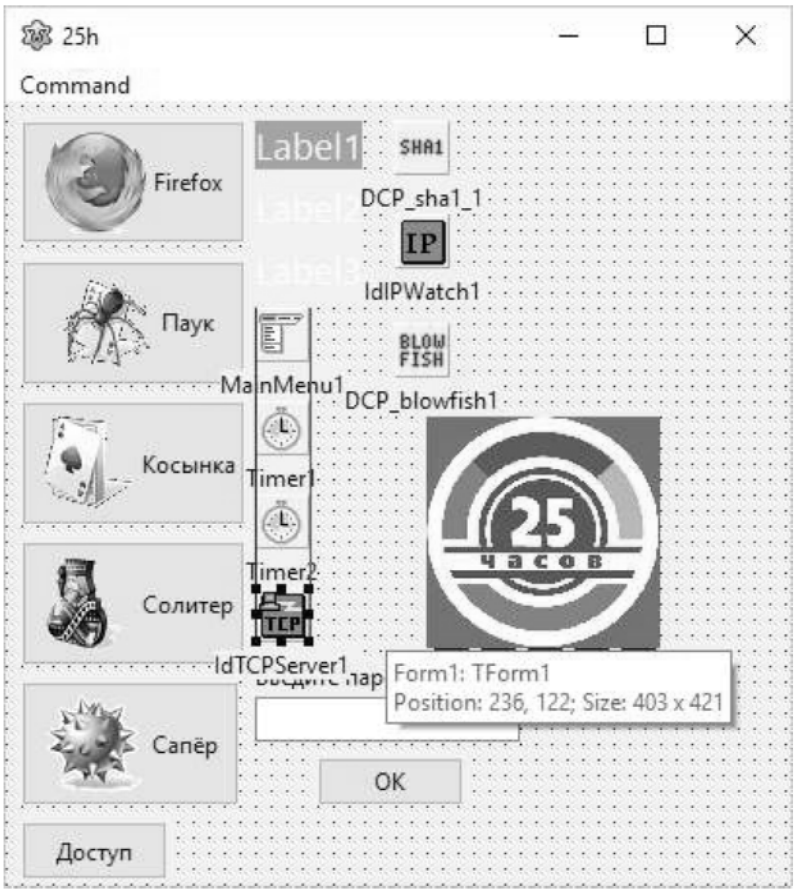


Рисунок 7 – Компонент IdTCPServer

Таблица 6 – Свойства компонента IdTCPServer

Свойство	Описание свойства
Active	Опередяляет включен ли компонент, может принимать занчении true, false.
IP Address	Указание IP-адреса на котором будет осуществляться работа компонента, для установления соединения
Port	Указание порта, который будет прослушиваться компонентом для установления соединения

3.1.6 DCP_blowfish и DCP_sha1

Эти два компонента не входят в стандартный комплект поставки IDE Lazarus, они устанавливаются отдельно, в составе пакета dcpcrypt. В данном случае использована версия 2.0.4.1. Было решено использовать уже готовые компоненты, т.к. это во-первых очень сильно упрощает и ускоряет процесс разработки приложения, а во-вторых дает возможность комбинировать различные методы шифрования и хеш.

Т.к. программа использует шифрование в том числе и для отправки шифрованных сообщений, т.е. важна скорость шифрования, а безопасностью в разумных пределах можно пренебречь, то были выбраны метод шифрования Blowfish и хеш Sha1.

Алгоритм Blowfish был разработан Брюсом Шнайером в 1994 г. Автор алгоритма предложил его в качестве замены стандарту DES. Blowfish оказался весьма удачным алгоритмом. Он широко реализован в различных шифровальных средствах - на сайте Брюса Шнайера (http://www.schneier.com) приведен список из примерно 150 продуктов, которые шифруют алгоритмом Blowfish. Алгоритм предназначен в основном для приложений, в которых ключ меняется нечасто, к тому же существует фаза начального рукопожатия, во время которой происходит аутентификация сторон и согласование общих параметров и секретов. Blowfish использует большое количество подключей. Эти ключи должны быть вычислены заранее, до начала любого шифрования или дешифрования данных. Данный алгоритм не запатентованный и свободно распространяемый[12].

Secure Hash Algorithm 1 — алгоритм криптографического хеширования. Описан в RFC 3174. Для входного сообщения произвольной

длины алгоритм генерирует 160-битное хеш-значение, называемое также дайджестом сообщения.

SHA1 используется во многих криптографических приложениях и протоколах. Также рекомендован в качестве основного для государственных учреждений в США. Принципы, положенные в основу SHA-1, аналогичны тем, которые использовались Рональдом Ривестом при проектировании MD4[11].

3.1.7 IdTCPServer и IdTCPClient

Общение серверной и клиентской части программы происходит через протокол TCP/IP. Для реализации были выбраны компоненты не входящие в стандартный состав IDE Lazarus, требуется установка пакета компонентов Indy. Мною была использована indy-10.2.0.3. Самой главной причиной выбора именно этого дополнительного пакета стало большое количество информации о работе с ним, причем как на Pascal, так и на C++ - он является универсальным. Все передаваемые сообщения предварительно шифруются при помощи компонентов, описанных выше.

3.2 Функции и процедуры

При написании данного проекта под ОС Windows нельзя было игнорировать WinAPI, опыт работы с которым имелся только на Lazarus. Несмотря на первый взгляд незначительные отличия Free Pascal от Pascal они все-таки имеются. Для использования WinAPI в Lazarus достаточно было в модулях указать «Winapi.Windows», а в IDE Lazarus необходимо подключать различные модули для работы с разными функциями, например мне понадобился именно «jwawinuser» чтобы заработала функция,

определяющая время простоя компьютера. Была составлена диаграмма классов, изображенная на рисунке 8.

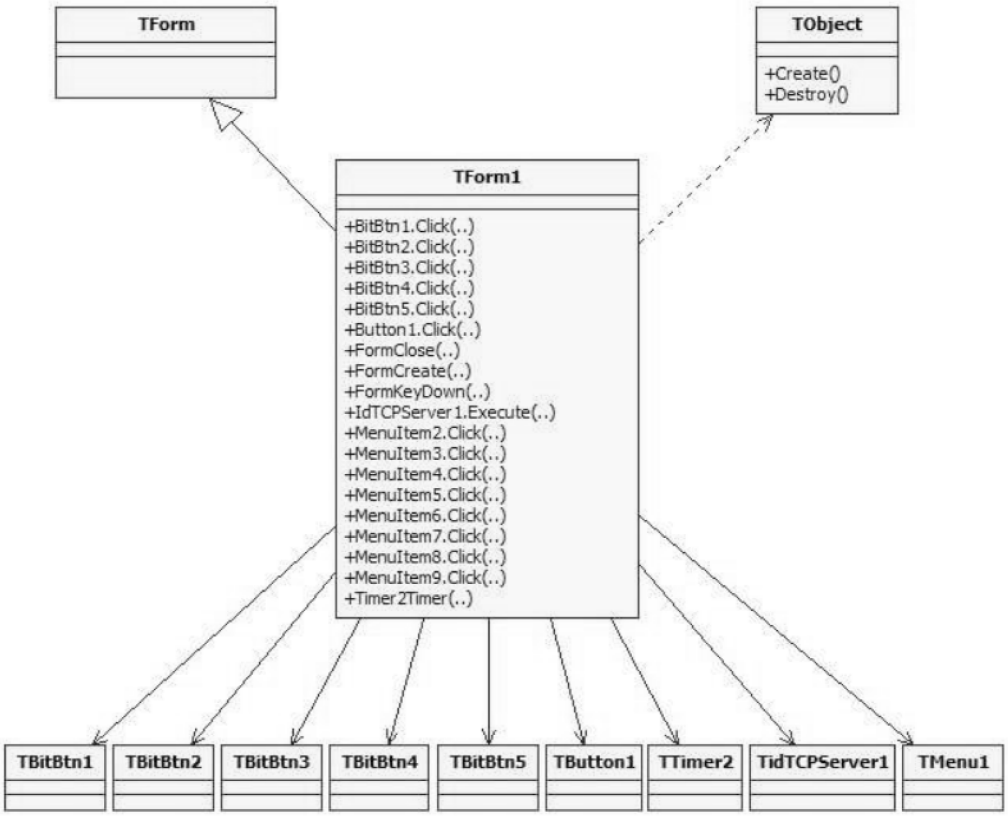


Рисунок 8 – Диаграмма классов клиентского модуля

Далее представлены описания функций, использующихся как в серверной, так и в клиентской части.

3.2.1 Функция GetVersion

Функция служит для определения текущей версии программы. Сначала получаем полную информацию о нашей программе API функцией GetFileVersionInfo, затем в цикле выделяем только версию продукта, которую записываем в результат выполения функции. Полный код функции можно найти в приложении А.

3.2.2 Процедура TForm1.FormKeyDown

Процедура служит для отображения изначально скрытых от пользователя элементов интерфейса, таких как LabeledEdit1 и Button1. Срабатывает по нажатию Ctrl+F10. Полный код процедуры можно найти в приложении А. Блок-схема изображена в приложении М.

1 – Начало процедуры TForm1.FormKeyDown

2 – Проверяем на истинность состояние нажатия клавиши Ctrl и F10

3 – Скрываем логотип компании и вместо него делаем видимыми компоненты, необходимые для ввода пароля

4 – Конец процедуры TForm1.FormKeyDown

3.2.3 Функция Lang

Функция служит для получения значения текущей раскладки клавиатуры, которое в дальнейшем будет выводиться на форму через компонент label. Полный код функции можно найти в приложении А. Блок-схема изображена в приложении Г.

Описание блок-схемы:

1 – Начало функции Lang

2 – С помощью WinAPI получаем значение текущей раскладки клавиатуры

3 – Преобразовываем строку с нулевым окончанием в строку стиля Pascal

4 – Проверяем Layout равно 409 или нет

5 – Раскладка английская

6 – Проверяем Layout равно 419 или нет

7 – Раскладка русская

8 – Конец функции Lang

3.2.4 Процедура Form1.Button1Click

Данная процедура отвечает за ввод пароля, чтобы получить доступ к интерфейсу администратора. Инициировать ее можно нажав заранее заданное сочетание клавиш. Полный код процедуры можно найти в приложении А. Блок-схема этого алгоритма представлена в приложении Д.

Описание блок-схемы:

1 – Начало процедуры Form1.Button1Click

2 – Создаем список строк SL и алгоритм шифрования blowfish

3 – Инициализируем ключ шифрования с хэшем sha1, загружаем в SL зашифрованный файл с настройками. Дешифруем его.

4 – Проверяем, совпадает или нет введенный пароль с 11 строкой SL

5 – Показываем скрытое меню, показываем изображение компании

6 – Скрываем форму введения пароля и кнопку подтверждения

7 – Очищаем форму введения пароля, показываем изображение компании, скрываем форму введения пароля

8 – Скрываем кнопку подтверждения пароля, выводим сообщение с кнопкой «ОК», что введен неверный пароль

9 – Чистим информацию о ключе шифрования, освобождаем память, занимаемую для SL и Cipher

10 – Конец процедуры Form1.Button1Click

3.2.5 Процедура TForm1.BitBtn1Click

Данная процедура отвечает за запуск браузера. При этом она проверяет, не запущен ли он уже, т.к. пользователи имеют привычку

сворачивать его. Если браузер уже запущен, алгоритм развернет его на весь экран. Полный код процедуры можно найти в приложении А. Блок-схема этого алгоритма представлена в приложении Е.

Описание блок-схемы:

1 – Начало процедуры TForm1.BitBtn1Click.

2 – Создаем список строк SL и алгоритм шифрования blowfish, инициализируем ключ шифрования с хэшем sha1.

3 – Загружаем в SL зашифрованный файл с настройками, дешифруем его.

4 – Чистим информацию о ключе шифрования, освобождаем память, занимаемую для Cipher, извлекаем дескриптор окна.

5 – Проверяем hWin отличен от нуля или нет.

6 – Получаем название текущего дескриптора окна.

7 – Проверяем, содержится или нет строка ‘Firefox’ в полученном названии окна.

8 – Выходим из цикла.

9 – Извлекаем дескриптор следующего окна.

10 – Проверяем hWin отличен от нуля или нет

11 – Разворачиваем найденное окно.

12 – Создаем процесс, исполняемый файл которого, указан в 6 строке SL.

13 – Запускаем созданный процесс, освобождаем память, занимаемую для AProcess и SL.

14 – Конец процедуры TForm1.BitBtn1Click.

3.2.6 Процедура TForm1.IdTCPServer1Execute

Данная процедура, отвечает за прием и отправку сообщений с управляющей частью ПО. Каждое сообщение шифруется, либо дешифруется. Полный код процедуры можно найти в приложении А. Блок-схема этого алгоритма представлена в приложении Ж.

Описание блок-схемы:

1 – Начало процедуры TForm1.IdTCPServer1Execute.

2 – Создаем алгоритм шифрования blowfish, инициализируем ключ шифрования с хэшем sha1.

3 – Читаем в переменнную msg строку из сокета.

4 – Расшифровываем строку msg.

5 – Проверяем, совпадает ли полученная строка со строкой ‘Info’.

6 – Формируем ответное сообщение и шифруем его.

7 – Отправляем зашифрованную строку по сети.

8 – Формируем строку для выполнения от имени администратора и создаем процесс.

9 – Передаем на выполнение сформированную строку и запускаем процесс, после чего освобождаем память, занимаемую для AProcess.

10 – Формируем ответное сообщение серверному модулю и шифруем полученную строку.

11 – Передаем по сети полученную зашифрованную строку.

12 – Чистим информацию о ключе шифрования, вызываем метод Free для Cipher

13 – Конец процедуры TForm1.IdTCPServer1Execute.

3.2.7 Процедура TForm1.Timer2Timer

Процедура, следит за тем, чтобы когда клиент отошел от ПК более чем на 2 минуты браузер закрывался. Кроме того, этот же обработчик следит за текущей раскладкой клавиатуры. Исходный код процедуры можно найти в приложении А. Блок-схема этого алгоритма представлена в приложении И.

Описание блок-схемы:

1 – Начало процедуры TForm1.Timer2Timer.

2 – Присваиваем в свойство Caption компонента Label1 строку, со значением текущей раскладки клавиатуры.

3 – Проверяем, отсутствие активности пользователя более 120 секунд или нет.

4 – Получаем дескриптор окна.

5 – Проверяем hWin – отличен от нуля или нет.

6 – Получаем название окна текущего дескриптора.

7 – Проверяем, содержит ли полученное название окна в себе строку ‘Firefox’.

8 – Прерываем цикл.

9 – Получаем дескриптор следующего окна.

10 – Проверяем hWin – отличен от нуля или нет.

11 – Отправляем команду на закрытие текущего дескриптора окна hWin.

12 – Получаем дескриптор окна

13 – Конец процедуры TForm1.IdTCPServer1Execute.

14 – Проверяем hWin – отличен от нуля или нет.

15 – Получаем название окна текущего дескриптора.

16 – Проверяем, содержит ли полученное название окна в себе строку ‘cmd’.

17 – Прерываем цикл.

18 – Получаем дескриптор следующего окна.

19 – Проверяем, hWin отличен от нуля или нет.

20 – Отправляем команду на закрытие текущего дескриптора окна hWin.

21 – Конец процедуры TForm1.Timer2Timer.

3.2.8 Процедура TForm1.Button1Click

Данная процедура запускает сканирование сети на предмет запущенных клиентских частей ПО. Из поиска автоматически удаляется адрес ПК, с которого запускаем серверную часть. Полный код процедуры можно найти в приложении Б. Блок-схема этого алгоритма представлена в приложении К.

Описание блок-схемы:

1 – Начало процедуры TForm1.Button1Click.

2 – Присваиваем переменную «c» нулю, удаляем все итемы из ComboBox1, указываем минимальное значение ProgressBar1 равное Edit6.Text.

3 – Указываем минимальное значение ProgressBar1 равное Edit7.Text, устанавливаем позицию ProgressBar1 в начало, присваиваем переменной «i» минимальное значение ProgressBar1.

4 – Проверяем i – меньше или нет по сравнению с максимальным значением ProgressBar1.

5 – Формируем IP адрес для проверки его на открытый порт клиентской части, заполняем ProgressBar1.

6 – Проверяем состояние порта на текущем IP-адресе.

7 – Проверяем состояние Connected, истина или ложь.

8 – Проверяем i – отлично от 10 или нет.

9 – Добавляем в ComboBox1 текущий проверяемый IP-адрес.

10 – Отключаемся от проверяемого IP-адреса.

11 – Конец процедуры TForm1.Button1Click.

3.2.9 Процедура TForm1.Panel1Click

Процедура выполняет шифрование, а затем и отправку зашифрованной команды клиентской части ПО. Вслед за отправкой мы принимаем ответное сообщение и записываем его в лог действий. Полный код процедуры можно найти в приложении Б. Блок-схема этого алгоритма представлена в приложении Л.

Описание блок-схемы:

1 – Начало процедуры TForm1.Panel1Click.

2 – Создаем алгоритм шифрования blowfish, инициализируем ключ шифрования с хэшем sha1.

3 – Проверяем значение CheckBox1, истина или ложь.

4 – Присваиваем хосту IdTCPClient1 текущее значение ComboBox1, а его порту значение из Edit8.

5 – Пытаемся подключиться к указанному хосту.

6 – Проверяем состояние подключения, истина или ложь.

7 – Отправляем зашифрованную строку из Edit1 по сети.

8 – Принимаем по сети ответ от клиентского модуля.

9 – Добавляем в Memo1 расшифрованную принятую строку.

10 – Конец процедуры TForm1.Panel1Click.

3.3 Интерфейс

3.3.1 Клиентский модуль программы (вид пользователя)

Рядовому пользователю доступен только интерфейс, изображенный на рисунке 9. Он позволяет одновременно и лишить пользователя множества функций, и стилизует общую картину ПК в зале ожидания.

Рисунок 9 – Интерфейс клиентского модуля для пользователя

3.3.2 Клиентский модуль программы (вид администратора)

Администратору же, после нажатия специальной комбинации клавиш и ввода пароля доступен интерфейс, изображенный на рисунке 10. Он включает в себя скрытое меню, с заранее определенными командами.

Рисунок 10 – Интерфейс клиентского модуля для администратора

3.3.3 Серверный модуль

Интерфейс программы, предназначенной для управления клиентскими ПК изображен на рисунке 11.

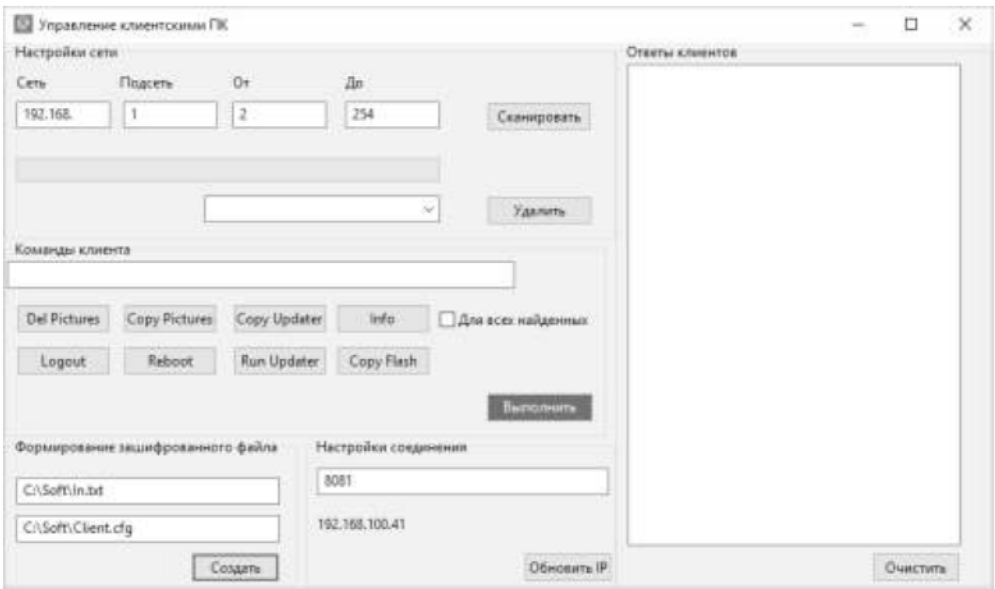


Рисунок 11 – Интерфейс клиентской части программы

Данный интерфейс задает область поиска в подсети, чтобы получить список ПК, на которых запущена серверная часть ПО. Из этого списка можно убрать ПК, работать с которыми мы не планируем. Команды отдаются либо индивидуально, либо всему списку целиком. Имеется логирование действий.

3.3.4 Модуль Updater

Т.к. модуль Updater лишь вызывается для выполнения обновления клиентского модуля, то графический интерфейс ему не нужен.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы было разработано и внедрено ПО информационно-развлекательного терминала на автомоечные комплексы «25 часов». А именно 3 его части:

1. Клиентский модуль. Служит для ограничения возможных действий пользователя и в то же время предоставления наиболее востребованных функций. Присутствует так же скрытое меню, защищенное паролем администратора, упрощающее обслуживание ПК. Листинг некоторых функций и процедур находится в приложении А.
2. Модуль «Updater», позволяющий производить обновление серверной части. Полный листинг модуля находится в приложении В.
3. Серверный модуль, запускаемый с любого ПК в пределах одной точки обслуживания, и позволяющая выполнять удаленные команды. Наиболее часто используемые команды вынесены на интерфейс формы в виде кнопок. Также имеется логирование действий. Листинг некоторых процедур находится в приложении Б.

Удалось достичь минимизации дополнительных затрат со стороны организации, внедрить функционал, строго необходимый для удобства обслуживания парка ПК. Кроме этого остается возможность изменить ПО, если появятся какие-либо дополнительные потребности.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Шкрыль, А.А. Разработка клиент-серверных приложений в Delphi / А.А. Шкрыль. – Санкт Петербург: БВХ-Петербург, 2006. – 480 с.
2. Бобровский, С.И. Технологии Delphi 2006. Новые возможности / С.И. Бобровский. – Санкт Петербург: Питер, 2006. – 288 с.
3. Фленов, М.Е. Библия Delphi (+ CD-ROM). 3-е изд., перераб. и доп. / М.Е. Фленов. – Санкт Петербург: БВХ-Петербург, 2011. – 686 с.
4. Чиртик, А.А. Delphi. Трюки и эффекты (+ CD-ROM) / А.А. Чиртик, В.В. Борисок, Ю.И. Корвель. – Санкт Петербург: Питер, 2007. – 400 с.
5. Описание компонентов Lazarus: [Электронный ресурс] http://helpdelphi.ru/.
6. Сервер Dell PowerEdge R610: [Электронный ресурс] https://krsk.au.ru/7527276/.
7. Операционная система тонких клиентов: [Электронный ресурс] http://wtware.ru/index.html#buy.
8. Microsoft Windows Server Standart 2012 R2 – Лицензии для организаций в Softline: [Электронный ресурс] http://store.softline.ru/microsoft/microsoft-windows-server-standard/.
9. Microsoft Windows 10 – Лицензии для организаций в Softline: [Электронный ресурс] http://store.softline.ru/microsoft/microsoft-windows-10/
10. RemotelyAnywhere – Remote Access Solutions: [Электронный ресурс] http://remotelyanywhere.com/template.asp?page=purchase.
11. SHA-1: [Электронный ресурс] http://kriptografea.narod.ru/Sha.html.
12. Криптография. Blowfish. Структура, описание, ключи шифрования.: [Электронный ресурс] http://alexinternetclic.ru/Blowfish.php.

ПРИЛОЖЕНИЕ А

Листинг клиентского модуля

1) Листинг функции GetVersion

```
function GetVersion(filename:String): string;
 var
   VerInfoSize: DWORD;
   VerInfo: Pointer;
   VerValueSize: DWORD;
   VerValue: PVSFixedFileInfo;
   Dummy: DWORD;
 begin
   VerInfoSize := GetFileVersionInfoSize(PChar(filename), Dummy);
   GetMem(VerInfo, VerInfoSize);
   GetFileVersionInfo(PChar(filename), 0, VerInfoSize, VerInfo);
   VerQueryValue(VerInfo, '\', Pointer(VerValue), VerValueSize);
   with VerValue^ do
   begin
     Result := IntToStr(dwFileVersionMS shr 16);
     Result := Result + '.' + IntToStr(dwFileVersionMS and $FFFF);
     Result := Result + '.' + IntToStr(dwFileVersionLS shr 16);
     Result := Result + '.' + IntToStr(dwFileVersionLS and $FFFF);
   end;
   FreeMem(VerInfo, VerInfoSize);
end;
```

2) Листинг процедуры TForm1.FormKeyDown

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
begin
  if (ssCtrl in Shift) and (Key=VK_F10) then
    begin
      Form1.Image1.Visible:=false;
      Form1.LabeledEdit1.Visible:=true;
      Form1.Button1.Visible:=true;
    end;
end;
```

3) Листинг функции Lang

```
function lang:string;
var
  lar: Array[0..$FFF] of Char;
  layout: String;
begin
  GetKeyboardLayoutName(lar) ;
  layout:=StrPas(lar);
  Case StrToInt(layout) of
    409: Result:='Eng';
    419: Result:='Rus';
  end;
end;
```

4) Листинг процедуры TForm1.Button1Click

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
  SL : TStringList;
  Cipher: TDCP_blowfish;
begin
  SL:=TStringList.Create;

  Cipher := TDCP_blowfish.Create(self);
  Cipher.InitStr('12345678',TDCP_sha1);
  SL.LoadFromFile('C:\Soft\Client.cfg');

  SL.Text:= Cipher.DecryptString(SL.Text);
  if LabeledEdit1.Text=SL[11] then //если пароль введен верно
  begin
    SetMenu(Handle,MainMenu1.Handle);
    Form1.Image1.Visible:=true;
    Form1.LabeledEdit1.Visible:=false;
    Form1.Button1.Visible:=false;
  end
    else              //если неверно
      begin
        LabeledEdit1.Text:='';       //чистим edit, скрываем форму, выводим
сообщение
        Form1.Image1.Visible:=true;
        Form1.LabeledEdit1.Visible:=false;
        Form1.Button1.Visible:=false;
        MessageDlg('Password incorrect', mtError, [mbOk], 0);
      end;
  SL.Free;
```

```
  Cipher.Burn;
  Cipher.Free;
end;
```

5) Листинг процедуры TForm1.BitBtn1Click

```
procedure TForm1.BitBtn1Click(Sender: TObject);
var
  SL : TStringList;
  Cipher: TDCP_blowfish;
  hWin: HWND;
  buff: array [0..255] of Char;
  AProcess: TProcess;
begin
  SL:=TStringList.Create;
  Cipher := TDCP_blowfish.Create(self);
  Cipher.InitStr('12345678',TDCP_sha1);
  SL.LoadFromFile('C:\Soft\Client.cfg');
  SL.Text:= Cipher.DecryptString(SL.Text);
  Cipher.Burn;
  Cipher.Free;
  hWin:= GetWindow(GetForegroundWindow, GW_HWNDFIRST);
  while hWin <> 0 do          //ищем окна firefox
    begin
      GetWindowText(hWin, buff, 255);
      if Pos('Firefox', buff) <> 0 then break;
      hWin:= GetWindow(hWin, GW_HWNDNEXT);
    end;
```

```
        if hWin<>0 then ShowWindow(hWin,SW_MAXIMIZE) else
//разворачиваем если нашли
          begin //запускаем если не нашли
            AProcess:=Tprocess.Create(nil);
            AProcess.CommandLine:=SL[6];
            AProcess.Active:=true;//AProcess.Execute;;
            AProcess.Free;
            SL.Free;
          end;
      end;
```

6) Листинг процедуры TForm1.IdTCPServer1Execute

```
procedure TForm1.IdTCPServer1Execute(AContext: TIdContext);
var
  msg : string;
  Cipher : TDCP_blowfish;
  AProcess: TProcess;
begin
  Cipher:=TDCP_blowfish.Create(self);
  Cipher.InitStr('12345678',TDCP_sha1);
  msg :=  AContext.Connection.Socket.ReadLn; //принимаем строку
  msg:= Cipher.DecryptString(msg);
   if msg='info' then
    begin
      msg:=' IP:' + IdIPWatch1.LocalIP + ' Ver:' +
GetVersion(Application.ExeName);
      msg:=Cipher.EncryptString(msg);
      AContext.Connection.Socket.WriteLn(msg);
```

```
    end
     else
    begin
     msg:='cmd.exe /c echo 456 | runas /netonly /user:"it" "' + msg + '"';
     AProcess:=Tprocess.Create(nil);
     AProcess.CommandLine:=msg;
     AProcess.Active:=true;//AProcess.Execute;
     AProcess.Free;
     msg:=IdIPWatch1.LocalIP+' komanda prinyata na ispolnenie: ' + msg;
     msg:=Cipher.EncryptString(msg);
     AContext.Connection.Socket.WriteLn(msg);
    end;
  Cipher.Burn;
  Cipher.Free;
end;
```

7) Листинг процедуры Timer2Timer

```
procedure TForm1.Timer2Timer(Sender: TObject);
var
  hWin: HWND;
  buff: array [0..255] of Char;
begin
  Label1.Caption:=lang;
  if SecondsIdle>=120 then          //2 мин до закрытия Firefox
    begin
      hWin:= GetWindow(GetForegroundWindow, GW_HWNDFIRST);
        while hWin <> 0 do
          begin
```

```
            GetWindowText(hWin, buff, 255);
            if Pos('Firefox', buff) <> 0 then break;
            hWin:= GetWindow(hWin, GW_HWNDNEXT);
          end;
        if hWin<>0 then SENDMESSAGE (hWin,WM_CLOSE,0,0);
        hWin:= GetWindow(GetForegroundWindow, GW_HWNDFIRST);
         while hWin <> 0 do
          begin
            GetWindowText(hWin, buff, 255);
            if Pos('cmd', buff) <> 0 then break;
            hWin:= GetWindow(hWin, GW_HWNDNEXT);
          end;
        if hWin<>0 then SENDMESSAGE (hWin,WM_CLOSE,0,0);
      end;
    end;
```

ПРИЛОЖЕНИЕ Б

Листинг серверного модуля

1) Листинг процедуры TForm1.Button1Click

```
procedure TForm1.Button1Click(Sender: TObject);

var
  i  : integer;

begin
  c :=0;
  ComboBox1.Items.Clear;
  ProgressBar1.Min := StrToInt(Edit6.Text);
  ProgressBar1.Max := StrToInt(Edit7.Text);
  ProgressBar1.Position := 0;

  for i := ProgressBar1.Min to ProgressBar1.Max do
    begin
      IdTelnet1.ConnectTimeout:=100;
      IdTelnet1.Host:=(Edit4.Text+Edit5.Text+'.' + IntToStr(i));
      IdTelnet1.Port:=strtoint(Edit8.Text);          //устанавливаем порт
      ProgressBar1.Position := ProgressBar1.Position + 1;
      try
        IdTelnet1.Connect;
        Application.ProcessMessages;
      except end;
      if IdTelnet1.Connected then
```

```
        begin
          if i <> 10 then
          begin
            ComboBox1.Items.Add(IdTelnet1.Host);
            c:= c+1;
          end;
          IdTelnet1.Disconnect;      //Отключаемся
        end;
    end;
    label4.Caption:=inttostr(c);
end;
```

2) Листинг процедуры TForm1.Panel1Click

```
procedure TForm1.Panel1Click(Sender: TObject);

var
i : integer;
info : string;
Cipher : TDCP_rc4;

begin
  Cipher := TDCP_rc4.Create(self);
  Cipher.InitStr('12345678',TDCP_Sha1);
  If CheckBox1.Checked = false then     //Если послать только одному

  begin
    IdTcpClient1.Host:=ComboBox1.Text;  //устанавливаем хост
    IdTcpClient1.Port:=strtoint(Edit8.Text);         //устанавливаем порт
```

```
            try
              IdTcpClient1.Connect;
            except end;     //пытаемся подключиться
            if IdTcpClient1.Connected then           //если удалось

              begin
                IdTCPClient1.Socket.WriteLn(Cipher.EncryptString(Edit1.Text));
//посылаем, что надо выполнить
                info:=IdTCPClient1.Socket.ReadLn;
                Memo1.Lines.Add(Cipher.DecryptString(info));
                IdTcpClient1.Disconnect;                 //отключаемся
              end;
          end;
          if CheckBox1.Checked=true then       //если послать всем найденным

            begin
              for i := 0 to (c-1) do

                begin
                  ComboBox1.ItemIndex:=i;
                  IdTcpClient1.Host:=ComboBox1.Text;  //устанавливаем хост
                  IdTcpClient1.Port:=strtoint(Edit8.Text);         //устанавливаем порт

                  try
                    IdTcpClient1.Connect;
                  except end;     //пытаемся подключиться
                  if IdTcpClient1.Connected then           //если удалось
                    begin
```

```
            IdTCPClient1.Socket.WriteLn(Cipher.EncryptString(Edit1.Text));
//посылаем, что надо выполнить
            info:=IdTCPClient1.Socket.ReadLn;
            Memo1.Lines.Add(Cipher.DecryptString(info));
            //IdTCPClient1.Socket.Close;
            IdTcpClient1.Disconnect;              //отключаемся
          end;
        end;
    end;
  Cipher.Burn;
  Cipher.Free;
end;
```

ПРИЛОЖЕНИЕ В

Листинг модуля «Updater»

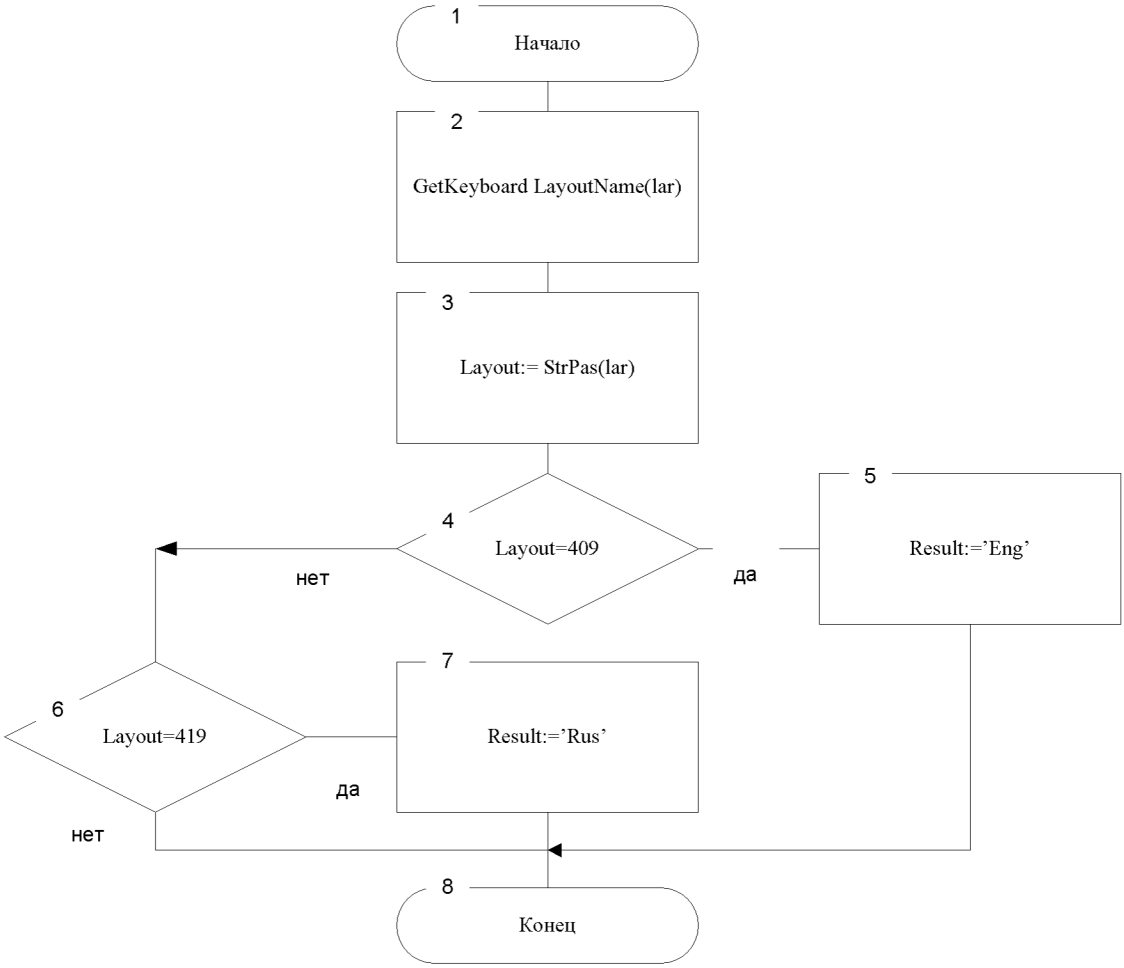
```
unit Unit1;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs,
DCPblowfish,
  DCPsha1, ShellAPI, Windows;
type
  { TForm1 }
  TForm1 = class(TForm)
    DCP_blowfish1: TDCP_blowfish;
    DCP_sha1_1: TDCP_sha1;
    procedure FormCreate(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;
implementation

{$R *.lfm}
```

```
{ TForm1 }

procedure TForm1.FormCreate(Sender: TObject);
var
  hWin: HWND;
  buff: array [0..255] of Char;
  Cipher : TDCP_blowfish;
  SL : TStringList;

begin
  hWin:= GetWindow(GetForegroundWindow, GW_HWNDFIRST);
    while hWin <> 0 do
      begin
        GetWindowText(hWin, buff, 255);
        if Pos('CLient', buff) <> 0 then break;
        hWin:= GetWindow(hWin, GW_HWNDNEXT);
      end;
    if hWin<>0 then SENDMESSAGE (hWin,WM_CLOSE,0,0);
  sleep(1000);
  Cipher:=TDCP_blowfish.Create(self);
  Cipher.InitStr('1234567878',TDCP_sha1);
  SL:=TStringList.Create; //открываем инишник в SL
  SL.LoadFromFile('C:\Soft\Updater.ini');
  SL.Text:=Cipher.DecryptString(SL.Text);
  Cipher.Burn;
  Cipher.Free;
  ShellExecute(0,'open','cmd.exe', '/c del "C:\Soft\Client.exe"
/Q','C:\Windows\system32\',SW_HIDE);
```

```
        ShellExecute(0,'open','cmd.exe',
PChar(SL[0]),'C:\Windows\system32\',SW_HIDE);
        SL.Free;
        sleep(4000);
        ShellExecute(handle ,'open','C:\Soft\Client.exe', nil, nil,
SW_MAXIMIZE);
        Application.Terminate;
      end;
      end.
```

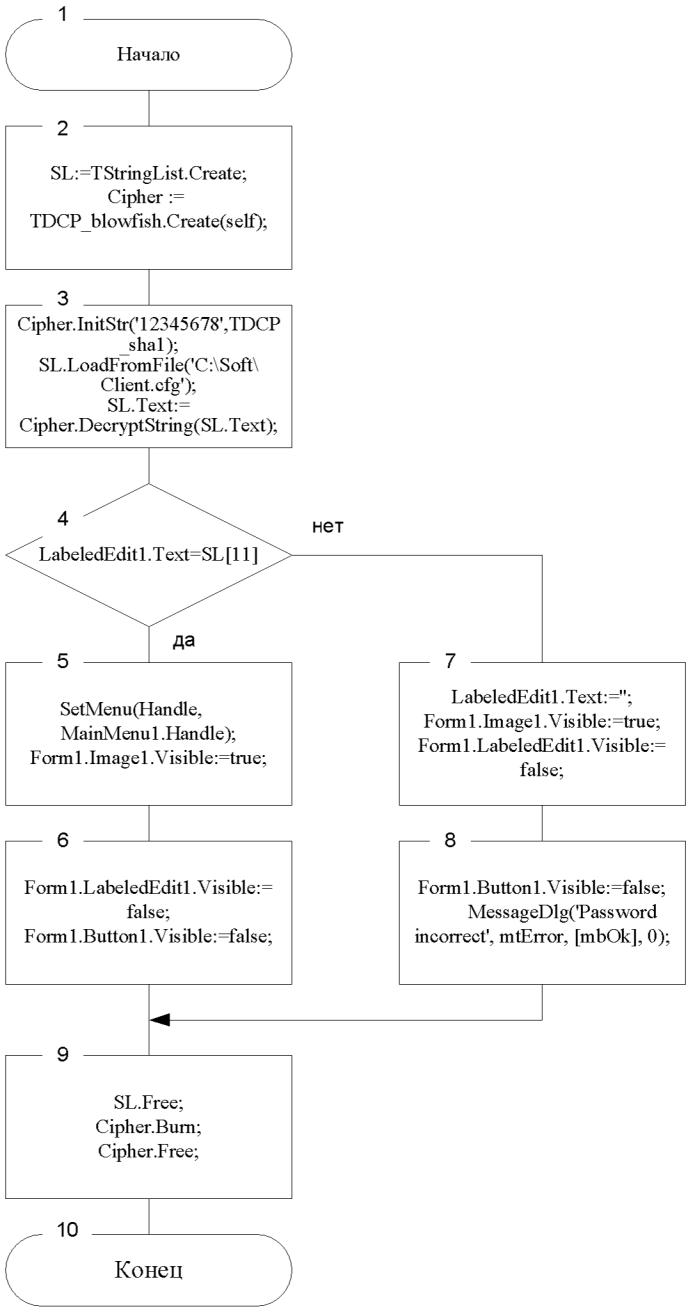
ПРИЛОЖЕНИЕ Г

Блок-схема функции Lang



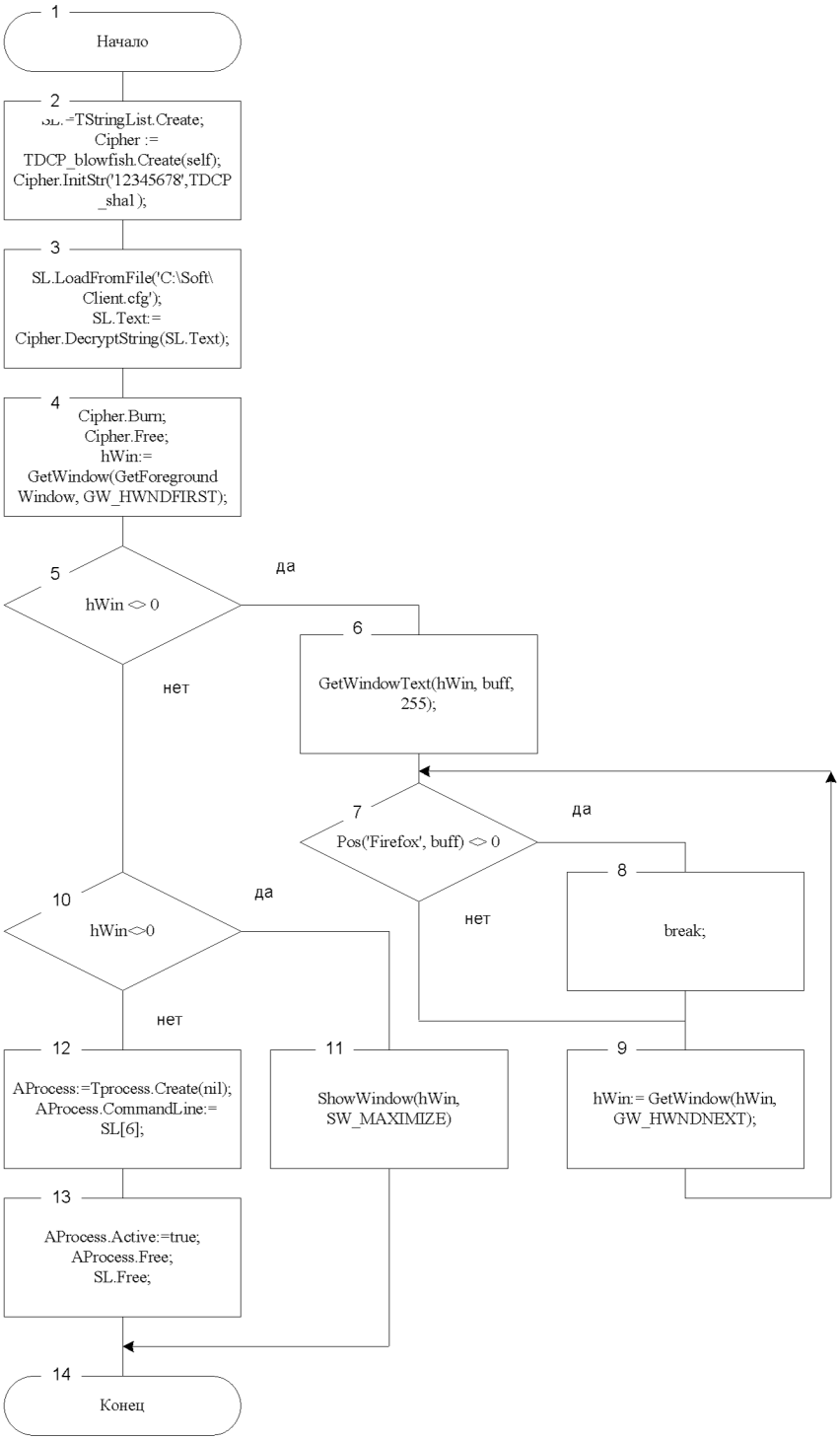
ПРИЛОЖЕНИЕ Д

Блок-схема процедуры Form1.Button1Click



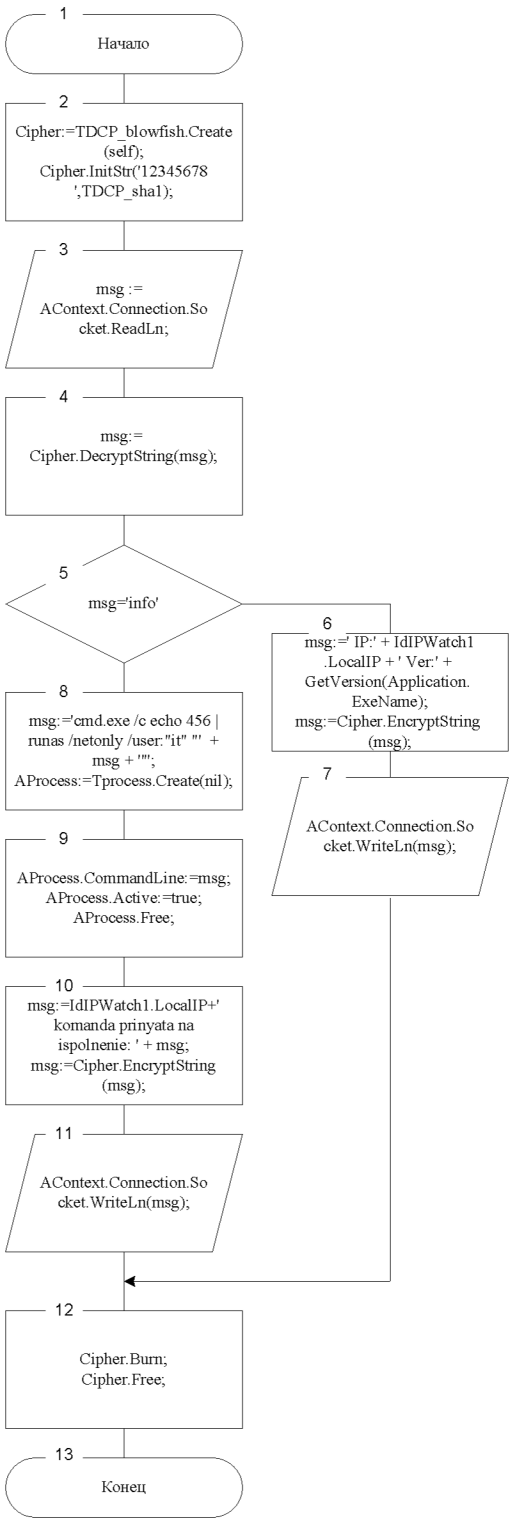
ПРИЛОЖЕНИЕ Е

Блок-схема процедуры TForm1.BitBtn1Click



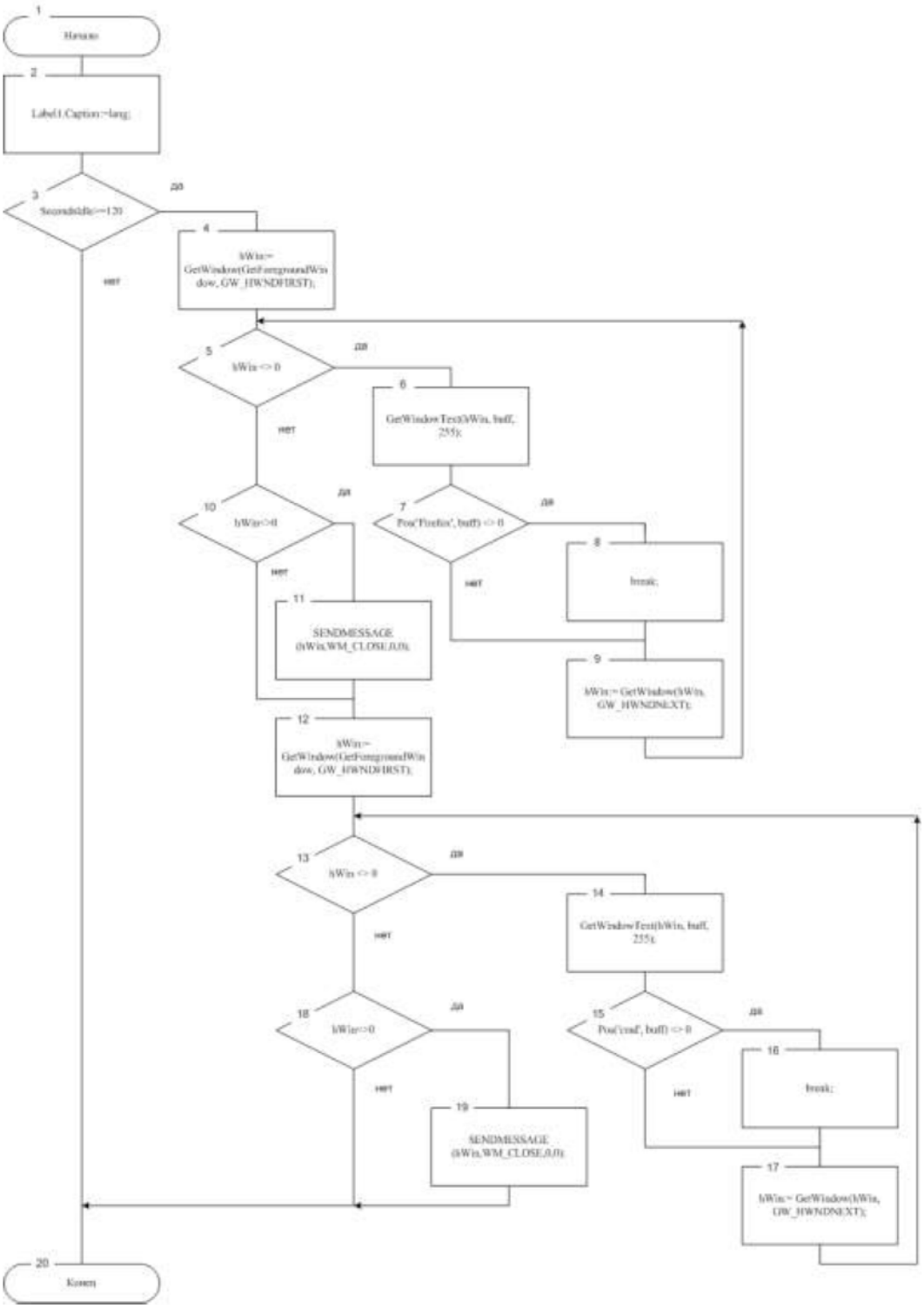
ПРИЛОЖЕНИЕ Ж

Блок-схема процедуры TForm1.IdTCPServer1Execute



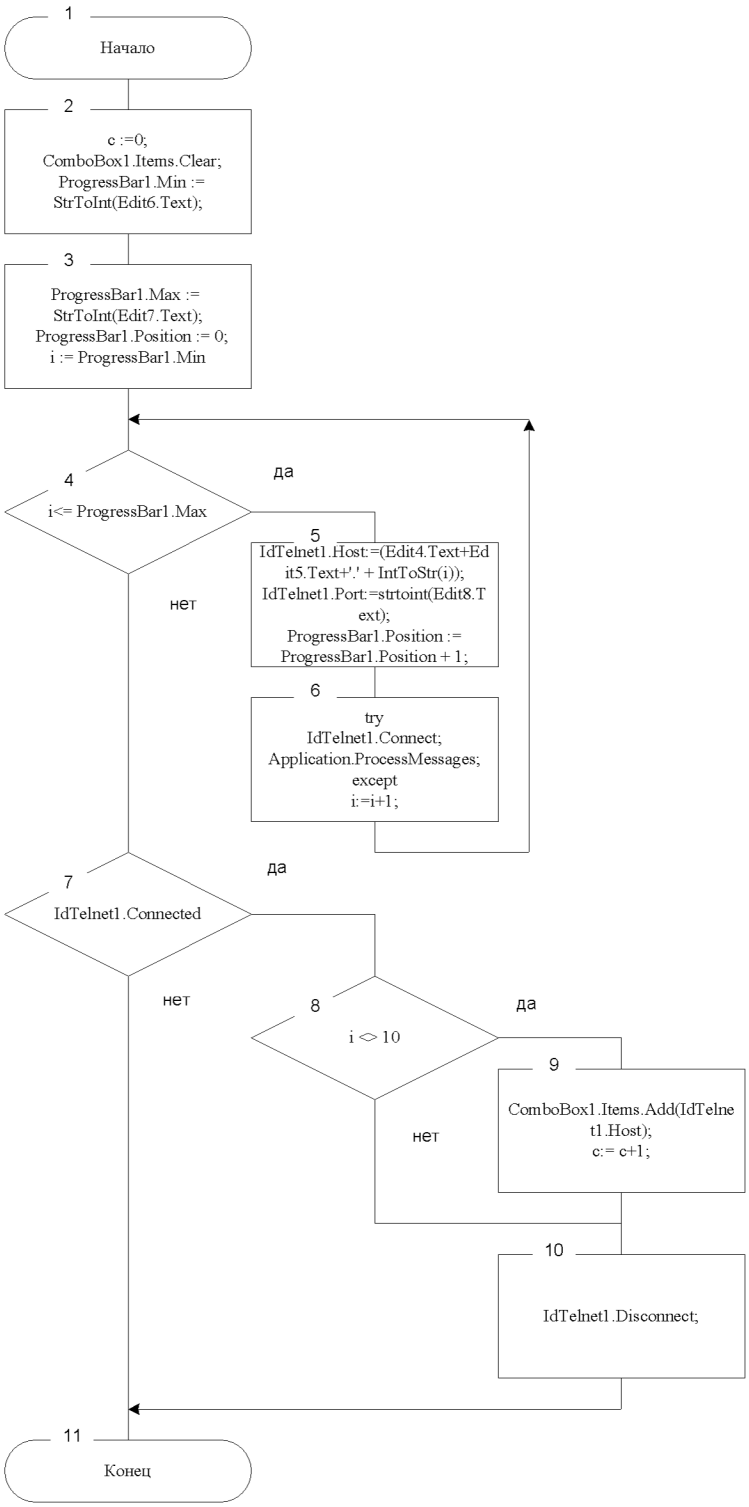
ПРИЛОЖЕНИЕ И

Блок-схема процедуры TForm1.Timer2Timer



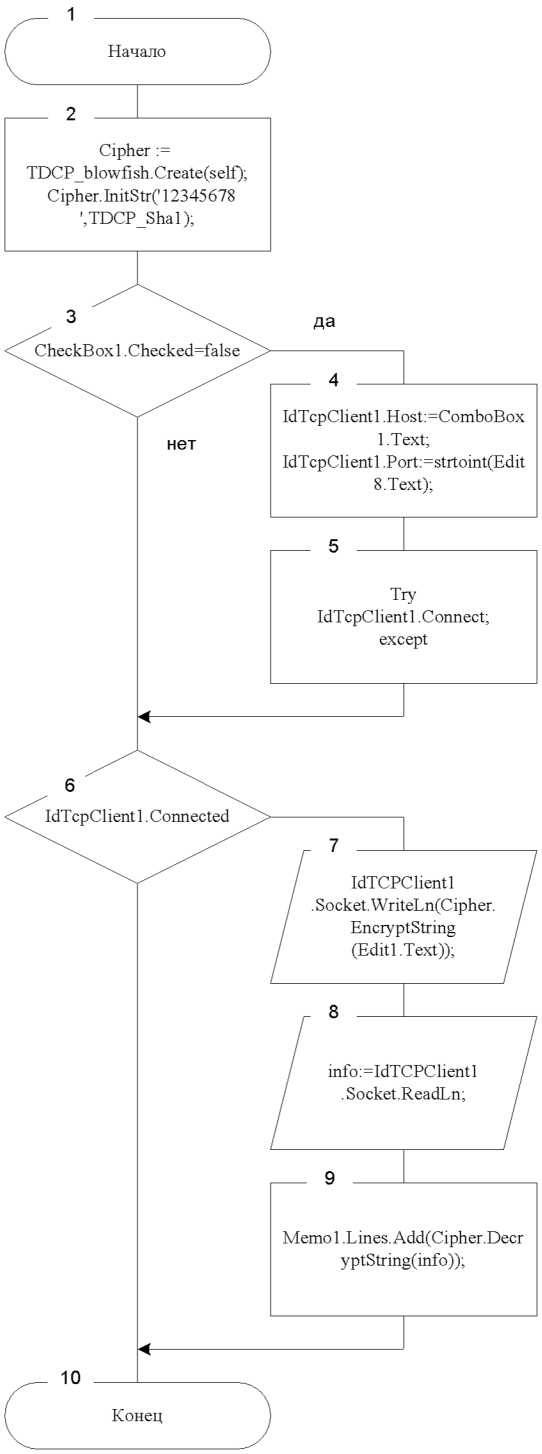
ПРИЛОЖЕНИЕ К

Блок-схема процедуры TForm1.Button1Click



ПРИЛОЖЕНИЕ Л

Блок-схема процедуры TForm1.Panel1Click



ПРИЛОЖЕНИЕ М

Блок-схема процедуры TForm1.FormKeyDown