ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc138011521)

[ГЛАВА 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc138011522)

[1.1 Системный реестр Windows: что это такое и как он работает 4](#_Toc138011523)

[1.2 Обзор аналогов 5](#_Toc138011524)

[ГЛАВА 2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 9](#_Toc138011525)

[2.1 Выбор программного обеспечения 9](#_Toc138011526)

[2.2 Разработка интерфейса программного средства 9](#_Toc138011527)

[2.3 Тестирование приложения 16](#_Toc138011528)

[ЗАКЛЮЧЕНИЕ 19](#_Toc138011529)

[Список использованных источников 20](#_Toc138011530)

[Приложение А 21](#_Toc138011531)

# ВВЕДЕНИЕ

С каждым годом информационные технологии занимают все большую роль в жизни людей и организаций.

Основным преимуществом, которым обладают современные компьютерные технологии — это возможность получения максимально большого объема всей необходимой информации, которая необходима в процессе обучения или реализации определенного бизнес-проекта.

Распространение современных компьютерных технологий оказывает большое влияние на процесс развития рынка труда. Благодаря созданию этих технологий появилось возможность существенно ускорить работу в практически любой компании и наладить коммуникацию между людьми, даже если они находятся друг от друга на большом расстоянии.

Современные компьютерные технологии способствуют развитию науки, существенно облегчая процесс вычисления и создания научных проектов.

Целью курсового проекта является наcтройка параметров протокола TCP/IP на языке Delphi.

Задачи:

1. Изучить материал, который необходим для выполнения работы
2. Разработать программу
3. Протестировать программу

Для выполнения поставленной задачи, необходимо современное оборудование, которое позволит комфортно работать. Программа разрабатывалась на операционной системе Windows 10, в среде разработки Embarcadero RAD Studio 11.

Объект исследования: использование среды Embarcadero RAD Studio 11, возможностей языка Delphi и его библиотек для решения поставленной задачи.

Предмет исследования: наcтройка параметров протокола TCP/IP.

# ГЛАВА 1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Системный реестр Windows: что это такое и как он работает

Реестр Windows или системный реестр - иерархически построенная база данных параметров и настроек в большинстве операционных систем Microsoft Windows. Реестр содержит информацию и настройки для аппаратного обеспечения, программного обеспечения, профилей пользователей, предустановки. Большинство изменений в Панели управления, ассоциации файлов, системные политики, список установленного ПО фиксируются в реестре. Реестр Windows был введён для упорядочения информации, хранившейся до этого во множестве INI-файлов, обеспечения единого механизма (API) записи-чтения настроек и избавления от проблем коротких имён, отсутствия разграничения прав доступа и медленного доступа к ini-файлам, хранящимся на файловой системе |FAT16, имевшей серьёзные проблемы быстродействия при поиске файлов в директориях с большим их количеством. Со временем (окончательно - с появлением NTFS) проблемы, решавшиеся реестром, исчезли, но реестр остался из-за обратной совместимости, и присутствует во всех версиях Windows включая последнюю. Поскольку сейчас не существует реальных предпосылок для использования подобного механизма, Microsoft Windows - единственная (не считая ReactOS) операционная система из используемых сегодня, в которой используется механизм реестра операционной системы. Реестр Windows функционирует как централизованный хранилище данных, содержащий информацию, которая периодически используется приложениями и операционной системой для определения настроек, интерфейса и исполнения программных задач. Реестр можно представить в виде древовидной структуры, где узлы являются различными разделами, а листья - ключами и значениями.

Реестр Windows работает следующим образом: при запуске операционной системы загружается ядро (с ядра начинается работа всей операционной системы), инициализируется реестр файлами, хранящимся на жестком диске, и копируется в оперативную память в качестве дерева. После того как реестр загружен в оперативную память, все программы и службы могут получить доступ к его функциональности.

Данные в реестре хранятся в виде ключей и значений. Ключи представляют собой контейнеры для значений, которые хранятся в них. Поиск ключей и значений в реестре Windows может быть выполнен с помощью редактора реестра, который включен в стандартный набор программного обеспечения операционной системы.

Однако, доступ к системному реестру должен быть ограничен, чтобы предотвратить ошибки и утечки информации. Для этого в Windows есть система прав доступа, которая определяет, кто может просматривать или изменять информацию в реестре. Таким образом, злоумышленники не смогут легко изменить настройки системы и нанести вред компьютеру.

Защита доступа к файлам и папкам через системный реестр Windows является одним из методов защиты компьютера. С помощью настройки прав доступа и изменения значений в реестре можно ограничивать доступ  
к информации на компьютере. Однако, важно учитывать, что ошибочно измененные настройки могут привести к неправильной работе компьютера, поэтому использование данного метода должно быть осуществлено с осторожностью.

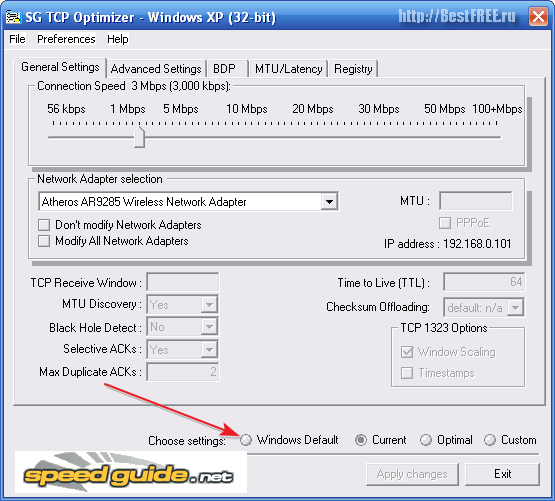
## 1.2 Обзор аналогов

SG TCP Optimizer

Маленькая программа для восстановления и оптимизации настроек сети и Интернета из серии «не работало — запустили — заработало». Помимо автоматического восстановления настроек позволяет оптимизировать десятки разнообразных параметров, которые влияют на производительность сетевой подсистемы ПК, причем как вручную, так и автоматически.

Рисунок 1

Интерфейс программы SG TCP Optimizer



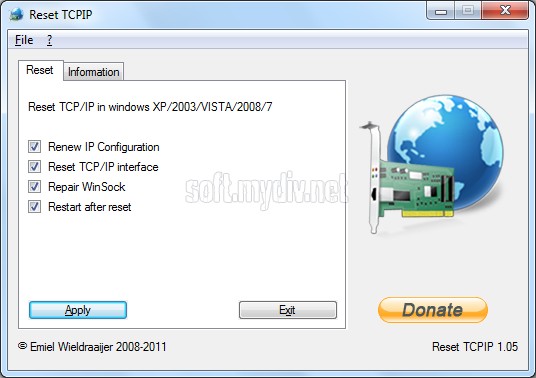
**Reset-TCPIP**

Небольшая, но крайне полезная программа, позволяющая восстановить соединение с Интернет, в случае его неисправности, путем сброса к начальным настройкам протокола TCP/IP в ОС Windows. Протокол TCP/IP в Windows является одним из необходимых компонентов, для соединения с сетью интернет. В случае изменения каких-либо настроек, соединение с интернет может быть недоступно. Программа Reset-TCPIP позволяет легко вернуть настройки протокола к первоначальным, и исправить все проблемы, возникшие, возможно, из-за случайного изменения настроек TCP/IP.

Поскольку протокол TCP/IP является ключевым компонентом Windows, удалить его и заново переустановить не получится, однако в случае возникновения каких-либо проблем с ним, можно просто сбросить его настройки на стандартные, при помощи утилиты Reset-TCPIP. Программа довольно удобна в использовании и не вызовет ни малейших затруднений даже у начинающих пользователей, так как выполняет всего одну функцию - сброс настроек протокола соединения с интернет - TCP/IP.

Рисунок 2

Интерфейс программы Reset-TCPIP



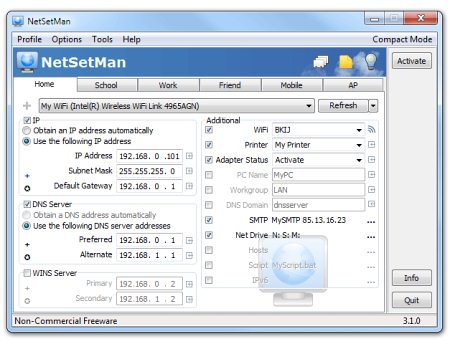
**NetSetMan**

[NetSetMan](http://www.netsetman.com/) - относительно небольшая бесплатная программа, позволяющая с легкостью переключаться между шестью различными сетевыми профилями, в число параметров которых входят: IP-адрес, маска подсети, шлюз, DNS-сервера, имя компьютера, рабочая группа, запускаемые скрипты и т.д. Имеется русский интерфейс.

В новой версии добавленные расширенные настройки TCP/IP (вместо "Additional Addresses"), представлена поддержка "NetSetMan Touch", исправлено множество обнаруженных ошибок.

Рисунок 3

Интерфейс программы [NetSetMan](http://www.netsetman.com/)



# ГЛАВА 2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

**Задание:** Разработка программного средства управляющего основными настройками протокола TCP/IP текущего компьютера путем модификации веток системного реестра.

## 2.1 Выбор программного обеспечения

В качестве операционной системы выберем ОС Windows 10.

К ее преимуществам относят:

* высокий запас по производительности;
* огромное количество программного обеспечения, доступного для скачивания;
* высокий уровень безопасности (относится как к защите от вредоносного ПО, так и от атак на виртуальные машины);
* возможность индивидуальной разработки ПО (обеспечивается поддержкой контейнеризации Kubernetes, широкими возможностями для масштабирования);
* работа с Windows Subsystem for Linux: позволяет разрабатывать ПО для операционной системы Linux, включая и виртуальные машины.

Клиентская операционная система осуществляет две основные функции: она предоставляет пользователю ряд тех или иных сервисов и управляет ресурсами компьютера, на котором она выполняется. Собственно, выбор операционной системы и определяется, во-первых, тем, какие у нее имеются ресурсы, а во-вторых, тем, какие сервисы требуются пользователю, — не все операционные системы способны работать с тем или иным аппаратным обеспечением, да и запросы пользователя (в том числе корпоративного) порой бывают столь высоки, что выбор операционных систем, способных их удовлетворить, оказывается весьма невелик.

Перечислим наиболее часто встречающиеся потребности пользователя:

Возможность применять офисные приложения (то есть готовить документы с помощью текстовых процессоров, электронных таблиц, средств презентационной графики и т.д.).

Возможность обращаться к ресурсам локальной сети и Интернета (например, к сетевым принтерам, файлам на сетевом диске или на Web-сайтах, к Web-приложениям и почтовым серверам).

Надежность, средства защиты данных, устойчивость к сбоям.

Delphi — среда программирования на основе языка Object Pascal. Он считается наследником Turbo Pascal, а тот — «чистого» Pascal Никлауса Вирта, созданного в 1970 году.

Изначальный Pascal обладал наиболее быстрым компилятором, но его IDE работала только на базе DOS. Когда появилась Windows, возникла потребность создания среды программирования для этой платформы. Раньше, чтобы создать элементарную программу, разработчикам приходилось писать несколько страниц кода. В компании Borland понимали, что нужно как-то облегчить их жизнь, поэтому решили доработать Pascal, который к тому моменту уже почти не использовался.

Delphi — язык программирования, который используется для разработки прикладного программного обеспечения. Программисты любят его, потому что это объектно-ориентированное программирование, которое даёт высокую скорость разработки, сокращает количество возможных ошибок и обеспечивает эффективную работу полученного приложения.

Когда стоит задача разработать приложение под Windows, Delphi — вне конкуренции. В нём есть много готовых компонентов, эффективный код и быстрая компиляция. А синтаксис менее запутан, чем, например, у C++.

В сравнении с другими языками программирования у него есть несколько важных преимуществ:

* Высокая скорость компиляции. Чтобы скомпилировать небольшое приложение в MSVC++ вам потребуется около 10 минут, в Delphi — пара секунд. Такая скорость достигается за счёт того, что Delphi раз и навсегда компилирует все модули, из которых строятся приложения, а потом просто подключает их к вашей программе. То есть, не нужно постоянно ждать пока перекомпилируются используемые библиотеки.
* Система компонентов. Компоненты — шаблоны, из которых можно строить программы. С их помощью вы можете создать довольно сложное программное обеспечение, не написав ни строчки кода. Предположим, вы хотите сделать текстовый редактор вроде Word. Это несложно в реализации, потому что все его «детали» по отдельности уже есть в палитре компонентов. Также вы можете интегрировать в среду Delphi подпрограммы-эксперты. Например, Expert Interface позволяет встраивать пункты меню и диалоговые окна.
* Оптимизирующий 32-битовый компилятор. У Delphi есть быстрый компилятор, который не только сообщает об ошибках, но и предупреждает о них, отправляет подсказки. Это помогает создать более ясный код. Если ошибок несколько, Delphi укажет на все сразу, что ускорит процесс исправления.
* Визуальный конструктор программ. В Delphi вам не нужно отвлекаться на выяснение всех деталей Win API и работать над логикой программы, если захотите программировать под MS Windows.
* Меньше возможных ошибок. В C-подобных языках присваивание — лишь выражение возвращающее значение переменной слева, и новички часто ошибаются. В Delphi такая ошибка исключена, так как присваивание — операция, не возвращающая значения.
* Возможность создания программ для любых операционных систем и платформ. Этим Delphi выгодно отличается от более популярного C#, который используется только для программирования на платформах .NET и Mono.

Отдельно стоит упомянуть ещё одно важное применение Delphi перед остальными языками — он отлично подходит для обучения программированию. Его часто рекомендуют в качестве первого языка для тех, кто только собирается стать программистом.

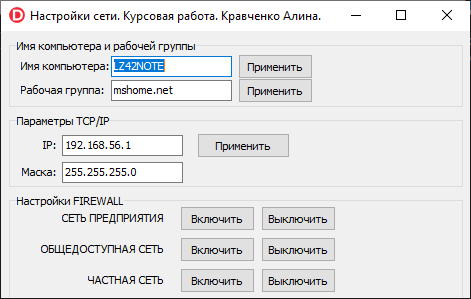
## 2.2 Разработка интерфейса программного средства

Создадим проект и разместим на главной форме приложения основные элементы управления с поясняющими надписями.

Внешний вид спроектированной формы представлен на рисунке 4.

Рисунок 4.

Внешний вид спроектированной формы.



Данные в реестре упорядочены в древовидные структуры, состоящие из узлов, которые в терминологии Windows называются ключами. Каждый ключ имеет один родительский ключ, один или несколько дочерних ключей и ноль или несколько параметров типа имя\_параметра =значение. Исключение представляют корневые ключи (они не имеют родителей) и дочерние ключи низшего уровня (они не имеют дочерних ключей). По умолчанию несистемное ПО регистрирует свою информацию в корневом ключе с именем *HKEY\_CURRENT\_USER*. Получить доступ к реестру можно написав в командной строке ("ПУСК > "Выполнить") слово "RegEdit" - при этом запуститься программа для редактирования реестра. Окно этой программы поделено на две части. В левой (более узкой панели) показана древовидная структура ключей. Сами установки называются параметрами, находящимися в правой панели. Каждый параметр имеет своё имя, значение и тип. Параметры бывают строкового типа, двоичного и типа DWORD. Их очень много, но их назначение зависит от того, в каком ключе находится той или иной параметр. Ключи делятся между шестью основными разделами:

* HKEY\_CLASSES\_ROOT – Содержит информацию об OLE, операциях перетаскивания (drag-and-drop - с англ. перетащить-и-отпустить) и ярлыках. В данном разделе можно так же указать программы, запускаемые при активизации файлов определённого типа. Данный раздел является псевдонимом для ветви HKEY\_LOCAL\_MACHINE\Software\Classes
* HKEY\_CURRENT\_USER – Содержит индивидуальные установки для каждого пользователя, зарегистрированного в системе. Данный раздел является псевдонимом для ветви HKEY\_USERS
* HKEY\_LOCAL\_MACHINE – Содержит аппаратные и программные установки, необходимые для функционирования оборудования и программ. Данный раздел так же хранит конфигурацию Windows.
* HKEY\_USERS – Содержит установки пользователей и соответствующие конфигурационные данные, такие как цвет окна, расположение элементов на рабочем столе, обои, заставки.
* HKEY\_CURRENT\_CONFIG – Содержит информацию о текущем аппаратном профиле. Если вы не используете аппаратные профили, данный раздел содержит установки Windows по умолчанию.
* HKEY\_DYN\_DATA – В отличие от других разделов, которые хранят статистические данные (неизменяющиеся во время сеанса), данный раздел содержит указатели на динамические данные (постоянно изменяющиеся во время работы компьютера). Windows использует данный раздел для отслеживания профилей оборудования plug-and-play, статистики по производительности и драйверов виртуальных устройств VxD.Все данные системного реестра заключаются в двух файлах, находящихся в директории Windows - это System.dat и User.dat. Работа с реестром при помощи её же средств осуществляется с помощью ряда API-функций специально предназначенных для этого Для работы с реестром на Delphi можно использовать стандартную библиотеку функций Windows API. Вот примеры некоторых операций с реестром:

**Чтение значения из реестра:**

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_LOCAL\_MACHINE;

if Reg.OpenKey('Software\Microsoft\Windows\CurrentVersion', False) then

begin

ShowMessage(Reg.ReadString('ProgramFilesDir'));

Reg.CloseKey;

end;

finally

Reg.Free;

end;

end;

**Запись значения в реестр:**

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_CURRENT\_USER;

if Reg.OpenKey('Software\MyApp', True) then

begin

Reg.WriteString('MySetting', 'MyValue');

Reg.CloseKey;

end;

finally

Reg.Free;

end;

end;

**Удаление значения из реестра:**

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_CURRENT\_USER;

if Reg.OpenKey('Software\MyApp', True) then

begin

Reg.DeleteValue('MySetting');

Reg.CloseKey;

end;

finally

Reg.Free;

end;

end;

**Удаление ключа из реестра:**

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_CURRENT\_USER;

if Reg.DeleteKey('Software\MyApp') then

ShowMessage('Key deleted')

else

ShowMessage('Key not found');

finally

Reg.Free;

end;

end;

Нужно обратить внимание, что для работы с реестром необходимы права администратора.

Для настройки протокола TCP/IP через реестр необходимо знать соответствующие ключи и значения в реестре Windows.

Например, для настройки IP-адреса и маски подсети можно использовать следующие ключи:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\IPAddress

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\SubnetMask

Например, чтобы задать IP-адрес 192.168.0.1 и маску подсети 255.255.255.0, можно использовать следующий код:

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_LOCAL\_MACHINE;

if Reg.OpenKey('SYSTEM\CurrentControlSet\Services\Tcpip\Parameters', True) then

begin

Reg.WriteString('IPAddress', '192.168.0.1');

Reg.WriteString('SubnetMask', '255.255.255.0');

Reg.CloseKey;

end;

finally

Reg.Free;

end;

end;

Для применения изменений необходимо перезагрузить сетевой адаптер или выполнить команду "ipconfig /renew" в командной строке.

Изменение имени компьютера и рабочей группы через реестр:

1. Необходимо открыть редактор реестра, нажав Win+R и введя команду regedit.

2. Перейти к следующему разделу: HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName.

3. Внутри этого раздела найти параметр ComputerName и дважды кликнуть на нем.

4. Изменить значение параметра на новое имя компьютера и нажмите ОК.

5. Перейти к разделу HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\ComputerName\ActiveComputerName.

6. Внутри этого раздела найти параметр ComputerName и дважды кликнуть на нем.

7. Изменить значение параметра на новое имя компьютера и нажать ОК.

8. Перейти к разделу HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters.

9. Внутри этого раздела найти параметр Domain и дважды кликнуть на нем.

10. Изменить значение параметра на новое имя рабочей группы и нажать ОК.

Настройка параметров сетевой защиты через реестр:

1. Откройте редактор реестра, нажав Win+R и введя команду regedit.

2. Перейдите к следующему разделу: HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy.

3. Внутри этого раздела находятся различные подразделы, отвечающие за разные типы сетей и защиты.

4. Для изменения параметров защиты для конкретного типа сети выберите соответствующий подраздел.

5. Внутри подраздела находятся параметры, отвечающие за различные аспекты защиты, например, блокирование определенных портов или протоколов.

6. Дважды кликните на параметре, который вы хотите изменить, и измените его значение.

7. Нажмите ОК, чтобы сохранить изменения.

Изменение имени подключения через реестр:

1. Откройте редактор реестра, нажав Win+R и введя команду regedit.

2. Педрейдите к следующему разделу: HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles.

3. Внутри этого раздела находятся подразделы, соответствующие различным сетевым подключениям.

4. Для изменения имени подключения выберите соответствующий подраздел.

5. Внутри подраздела найдите параметр ProfileName и дважды кликните на нем.

6. Измените значение параметра на новое имя подключения и нажмите ОК.

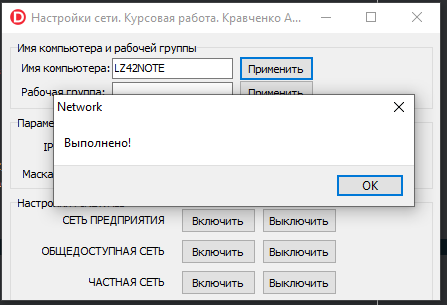
Для выполнения этих действий необходимо потратить большое количество действий и потратить много времени. Удобнее разработать программу, которая будет выполнять все эти действия за один или два клика мышью.

При нажатии соответствующей кнопки модифицируется соответствующее значение ключа реестра и выводится сообщение о завершении операции.

Результат выполнения операции представлен на рисунке 5.

Рисунок 5.

Результат выполнения операции.



## 2.3 Тестирование приложения

Тестирование — это наблюдение за работой приложения в разных искусственно созданных ситуациях. Таким образом тестируют производительность, уровень защиты и удобство пользования программой.

Основные задачи тестирования:

1. выявить ошибки и оперативно их устранить;
2. проверить соответствие продукта заявленным требованиям;
3. оценить качество работы разработчиков;
4. получить информацию, необходимую для принятия дальнейших решений.

**Почему важно тестировать программы**

Вот типичные программные баги:

* Вы вводите в поле ответ на вопрос и нажимаете Enter. После этого программа неожиданно завершает работу, не сохранив информацию. И та же ошибка повторяется в следующий раз.
* Другой случай: вы играете, например, в какую-нибудь стрелялку. Неожиданно персонажи начинают хаотично двигаться, конвульсивно дёргаться, терять или отращивать конечности. И вообще ведут себя не так, как им положено. Некоторое время программа не реагирует на нажатие клавиш, после чего выдаёт «Game over».
* Ещё один пример: вы заходите в личный кабинет интернет-магазина. Нажимаете «Оплатить», а вас выкидывает на главную страницу, да ещё и разлогинивает.

Программ без ошибок не бывает: любая может выдать непредсказуемый результат в ответ на самые обычные действия. Разработчик, скорее всего, не заметит этих дефектов в коде, зато конечному пользователю они могут отравить жизнь. Бывают ошибки мелкие и незначительные, а бывают и такие, что всё перестаёт работать.

Прежде чем новая версия компьютерной программы, сайта или мобильного приложения попадает к пользователю, она должна пройти через руки инженеров-тестировщиков. Они ищут места в коде, где программа работает не так, как задумано. Чтобы найти как можно больше ошибок, тестировщики моделируют разные ситуации, которые могут возникнуть при использовании приложения.

Пользователи непредсказуемы. Они могут делать то, что категорически не предусмотрено программой. Тестировщик должен проверить все возможные и невозможные сценарии их поведения и убедиться, что программа продолжит работать.

Вообще, у тестирования есть философия, которая строится на том, что в любой программе по определению есть ошибки и найти их все невозможно. А если вы почему-то не нашли ошибку, значит, просто плохо искали. Удачный тест для тестировщика — тот, при котором нашли баг. А если всё нормально работало, значит, тест неудачный и свою задачу не выполняет.

Ошибки возникают не только при программировании, но и при проектировании системы, и даже на этапе разработки технического задания. Поэтому и тестируют код не только в самом конце работы, а на разных этапах.

Есть несколько видов тестирования:

* **Статическое**, без запуска программы, и **динамическое** — с запуском. Статическое обычно делают в самом начале работы: инженеры проверяют проектную документацию и спецификации, вычитывают уже написанный код. Затем проводят динамическое тестирование: программу запускают и проверяют, как она ведёт себя во время работы, определяют время отклика и то, насколько она загружает процессор и память.
* С помощью **функционального тестирования** проверяют, как программа решает задачи, нужные клиенту. При **нефункциональном** исследуют производительность системы, её надёжность и защищённость, работу с окружением — операционной системой и оборудованием.
* Ещё один способ — **тестирование по принципу чёрного и белого ящика**. В первом случае тестировщик не смотрит на код и работает только с программным интерфейсом. Он проверяет производительность программы, все ли нужные функции реализованы, ищет ошибки в её интерфейсе и поведении. Во втором — инженер имеет доступ к коду. Он проверяет структуру и логику всей программы или отдельных её компонент. Часто этим занимается сам программист.
* **Ручное и автоматическое тестирование.** В первом случае работу кода проверяют вручную, без использования программных средств. Во втором — применяют специально написанные автоматические тесты, которые постоянно обновляют.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была проведена аналитическая работа, по результатам которой было выявлено, что системный реестр Windows предоставляет средства для настроек основных свойств протокола TCP/IP.

Разработана программа, считывающая значения настроек протокола и позволяющая изменять их.

Проведенные исследования показали, что использование системного реестра является эффективным инструментом для работы с настройками сетевых протоколов.

Получены навыки разработки программ, работающих с разделами, ключами и значениями реестра. Так же приобретены знания по разработке интерфейса программ. Рассмотрены способы активации и дезактивации программных средств сетевой защиты для различных типов сетевого соединения.

# Список использованных источников

1. Книттель, Брайан Windows 7. Скрипты, автоматизация и командная строка / Брайан Книттель. - М.: Питер, 2011. - 149 c.
2. Кокорева Реестр Windows 2000 / Кокорева. - Москва: Наука, 2003. - 909 c.
3. Кокорева, Ольга Реестр Windows XP / Ольга Кокорева. - М.: БХВ-Петербург, 2005. - 548 c.
4. Кузан, Д.Я. Программирование Win32 API в Delphi (+ CD-ROM) / Д.Я. Кузан. - М.: БХВ-Петербург, 2013. - 665 c.
5. Курячий, Г. Операционная система Линукс / Г. Курячий. - М.: ДМК Пресс, 2009. - 706 c.
6. Лебедь, М.П. Microsoft Visual FoxPro? Это не так уж сложно... / М.П. Лебедь. - М.: Нобель Пресс, 2014. - 205 c.
7. Натан, Адам WPF 4. Подробное руководство / Адам Натан. - М.: Символ-плюс, 1986. - 336 c.
8. Павлов, Станислав Введение в Windows Embedded CE 6.0. Версия R2 / Станислав Павлов , Павел Белевский. - М.: Богородский полиграфический комбинат, 2009. - 248 c.
9. Павлов, Станислав Основы Windows Embedded Standart 2009 (+ CD-ROM) / Станислав Павлов. - М.: ДМК Пресс, 2010. - 368 c.
10. Побегайло, Александр Системное программирование в Windows / Александр Побегайло. - М.: БХВ-Петербург, 2006. - 128 c.
11. Редкар, Теджасви Платформа Windows Azure / Теджасви Редкар , Тони Гвидичи. - М.: ДМК Пресс, 2014. - 656 c.
12. Румянцев, П.В. Работа с файлами в Win 32 API / П.В. Румянцев. - М.: Горячая линия - Телеком, 2009. - 892 c.
13. Солдатов, В.П. Программирование драйверов Windows / В.П. Солдатов. - М.: Бином. Лаборатория знаний, 2014. - 383 c.
14. Теджасви, Редкар Платформа Windows Azure / Редкар Теджасви. - М.: ДМК Пресс, 2017. - 418 c.
15. Топорков, С. Windows XP для продвинутых пользователей / С. Топорков. - М.: ДМК Пресс, 2005. - 421 c.

# Приложение А

**Текст программы.**

unit UMAIN;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Registry, WinSock;

type

TForm1 = class(TForm)

GroupBox1: TGroupBox;

GroupBox2: TGroupBox;

GroupBox3: TGroupBox;

GroupBox4: TGroupBox;

Label1: TLabel;

Label2: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Button1: TButton;

Button2: TButton;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Edit3: TEdit;

Edit4: TEdit;

Edit5: TEdit;

Button3: TButton;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Button4: TButton;

Button5: TButton;

Button6: TButton;

Button7: TButton;

Button8: TButton;

Button9: TButton;

procedure FormCreate(Sender: TObject);

function GetNTDomainName: string;

function readreg(root:string; key:string; val:string):string;

procedure writereg(root:string; key:string; val:string);

procedure delreg(root:string; key:string; val:string);

procedure writeregW(root:string; key:string; val:integer);

procedure setIP(ip:string;mask:string);

procedure Button1Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button9Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

function Tform1.readreg(root:string; key:string; val:string):string;

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_LOCAL\_MACHINE;

if Reg.OpenKey(key, False) then

begin

readreg:=(Reg.ReadString(val));

Reg.CloseKey;

end;

finally

Reg.Free;

end;

end;

procedure Tform1.writereg(root:string; key:string; val:string);

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_LOCAL\_MACHINE;

if Reg.OpenKey(root, True) then

begin

Reg.WriteString(key, val);

Reg.CloseKey;

end;

finally

Reg.Free;

end;

end;

procedure Tform1.writeregW(root:string; key:string; val:integer);

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_LOCAL\_MACHINE;

if Reg.OpenKey(root, True) then

begin

Reg.WriteInteger(key, val);

Reg.CloseKey;

end;

finally

Reg.Free;

end;

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

setIP(edit3.text,edit4.text);

ShowMessage('Выполнено!');

end;

procedure TForm1.Button4Click(Sender: TObject);

begin

writereg('SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\DomainProfile', 'EnableFirewall', '1');

ShowMessage('Выполнено!');

end;

procedure TForm1.Button5Click(Sender: TObject);

begin

writereg('SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\DomainProfile', 'EnableFirewall', '0');

ShowMessage('Выполнено!');

end;

procedure TForm1.Button6Click(Sender: TObject);

begin

writereg('SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\PublicProfile', 'EnableFirewall', '1');

ShowMessage('Выполнено!');

end;

procedure TForm1.Button7Click(Sender: TObject);

begin

writereg('SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\PublicProfile', 'EnableFirewall', '0');

ShowMessage('Выполнено!');

end;

procedure TForm1.Button8Click(Sender: TObject);

begin

writereg('SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile', 'EnableFirewall', '1');

ShowMessage('Выполнено!');

end;

procedure TForm1.Button9Click(Sender: TObject);

begin

writereg('SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile', 'EnableFirewall', '0');

ShowMessage('Выполнено!');

end;

procedure Tform1.delreg(root:string; key:string; val:string);

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_CURRENT\_USER;

if Reg.DeleteKey(key) then

ShowMessage('Key deleted')

else

ShowMessage('Key not found');

finally

Reg.Free;

end;

end;

procedure Tform1.setIP(ip:string;mask:string);

var

Reg: TRegistry;

begin

Reg := TRegistry.Create;

try

Reg.RootKey := HKEY\_LOCAL\_MACHINE;

if Reg.OpenKey('SYSTEM\CurrentControlSet\Services\Tcpip\Parameters', True) then

begin

Reg.WriteString('IPAddress', ip);

Reg.WriteString('SubnetMask', mask);

Reg.CloseKey;

end;

finally

Reg.Free;

end;

end;

function GetComputerNetName: string;

var

buffer: array[0..255] of char;

size: dword;

begin

size := 256;

if GetComputerName(buffer, size) then

Result := buffer

else

Result := ''

end;

function GetLocalIP: String;

const WSVer = $101;

var

wsaData: TWSAData;

P: PHostEnt;

Buf: array [0..127] of Char;

begin

Result := '';

if WSAStartup(WSVer, wsaData) = 0 then begin

if GetHostName(@Buf, 128) = 0 then begin

P := GetHostByName(@Buf);

if P <> nil then Result := iNet\_ntoa(PInAddr(p^.h\_addr\_list^)^);

end;

WSACleanup;

end;

end;

function TForm1.GetNTDomainName: string;

var hReg: TRegistry;

begin

hReg := TRegistry.Create;

hReg.RootKey := HKEY\_LOCAL\_MACHINE;

hReg.OpenKey('SYSTEM\CurrentControlSet\Services\Tcpip\Parameters', false);

GetNTDomainName := hReg.ReadString('ICSDomain');

hReg.CloseKey;

hReg.Destroy;

end;

procedure TForm1.Button1Click(Sender: TObject);

var

s:string;

begin

s:=readreg('','Software\Microsoft\Windows\CurrentVersion','ProgramFilesDir');

ShowMessage('Выполнено!');

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

Edit1.Text:= GEtComputerNetName;

Edit2.Text:= GetNTDomainName;

Edit3.TExt:= GetLocalIP;

Edit4.Text:='255.255.255.0';

end;

end.