МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Прикладной информатики

БАКАЛАВРСКАЯ РАБОТА

На тему

«ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ИНФОРМАЦИОННЫХ СИСТЕМ В КОНКРЕТНЫХ ПРЕДМЕТНЫХ ОБЛАСТЯХ» НА ПРИМЕРЕ: «РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ОПРЕДЕЛЕНИЯ КОНФИГУРАЦИИ ПК»

Исполнитель

Аллало Мурат Сергеевич

Руководитель

Ст. преподаватель кафедры «Прикладная информатика»
Степанов Сергей Юрьевич

«К защите допускаю»
Заведующий кафедрой _____

(подпись)

кандидат технических наук
Слесарева Людмила Сергеевна

«22» 06 2016 г.

Санкт–Петербург
2016

ОГЛАВЛЕНИЕ

Введение

В любых организациях всегда существует необходимость учёта и контроля аппаратной конфигурации компьютеров. Данная проблема может быть решена при помощи ручного труда, но проще, быстрее и удобнее делать это при помощи специального программного обеспечения. В данной работе описан процесс разработки автоматизированной информационной системы для учёта и контроля аппаратной конфигурации компьютеров организации.

Актуальность выбранной темы заключается в том, что существует необходимость учёта и контроля аппаратной конфигурации компьютеров в АО «Туапсинское АТП»

Объект исследования —процесс учёта и контроля аппаратной конфигурации компьютеров в АО «Туапсинское АТП».

Предмет исследования — теоретические и практические основы разработки программы учёта и контроля аппаратной конфигурации компьютеров в АО «Туапсинское АТП».

Цель курсовой работы — разработка программного приложения для учёта и контроля аппаратной конфигурации компьютеров АО «Туапсинское АТП».

Исходя из актуальности темы и поставленных целей, при написании работы были поставлены следующие задачи:

— рассмотреть теоретические основы разработки программных приложений;

— рассмотреть существующие решения в данной области;

— разработать программное приложение для учёта и контроля аппаратной конфигурации компьютеров организации.

Методологической основой написания работы являются: эмпирические методы: наблюдения и изучение процессов обработки информации в вычислительных системах; теоретические методы: теоретический анализ учебной литературы (выделение и рассмотрение отдельных сторон, признаков,

особенностей, свойств явлений); статистические методы анализа числовых (математических) характеристик эффективности работы вычислительных систем.

Теоретической основой написания работы является учебная и научная литература по вопросам программирования и разработки программных продуктов. При выполнении данной курсовой работы было использовано программное обеспечение класса текстовый процессор, программное обеспечение табличный процессор Microsoft Office Excel и операционная система Microsoft Windows.

Указанные цели и задачи определили логику изложения материала и структуру работы, состоящую из трёх глав.

Первая глава освещает основы требований к системе.

Во второй главе даётся описание существующих решений в данной области.

В третьей главе проводится описание процесса разработки программного приложения для учёта и контроля аппаратной конфигурации компьютеров организации.

Глава 1 Основания для разработки и требования к системе

1.1 Описание предметной области

Организационно - правовая форма предприятия АО «Туапсинское автотранспортное предприятие» является акционерное общество. Место нахождения ОАО «Туапсинское автотранспортное предприятие» Российская Федерация, Краснодарский край, г. Туапсе ул. Бондаренко, д.14. Руководство предприятием осуществляет директор. В структуре управления предприятием имеются 2 основные службы - служба эксплуатации, техническая служба.

Отдел эксплуатации имеет в своей структуре подотдел эксплуатации, кассу пересчета денежных средств, охрану. Основной функцией отдела эксплуатации является планирование работы маршрутов, рейсов, расписания, потребного количества автобусов, осуществление плана - графиков выхода автобусов на линии для предприятия.

Главной задачей этой службы является выполнение плана доходов от перевозки пассажиров, и поиск иных источников дохода.

В рамках функционирования АО «Туапсинское АТП» большое значение имеет подбор, учет и анализ конфигурации используемой компьютерной техники.

Необходимо обеспечить возможность проведения удалённой автоматизированной инвентаризации компьютеров в локальной вычислительной сети АО «Туапсинское АТП». Для решения данной задачи было выбрано программное приложение Microsoft Office Excel из пакета прикладного программного обеспечения Microsoft Office. В качестве средства программирования была использована встроенная среда программирования VBA.

Данное программное приложение было выбрано потому, что оно является фактическим стандартом в организациях, применяемым для решения задач табличного учёта и обработки данных. При необходимости можно использовать аналогичный свободный программный продукт OpenOffice или

аналогичный ему.

Решение должно обеспечивать удобный ввод названий компьютеров, находящихся в локальной вычислительной сети и наглядное отображение считанных данных. Должна обеспечиваться возможность предварительного просмотра и печати полученного документа. Эта возможность реализуется средствами программного приложения Microsoft Office Excel автоматически.

Также решение должно предусматривать возможность лёгкой модификации исходного кода в случае возникновения дополнительных требований к функциональности решения. Данное требование соблюдается благодаря особенностям реализации среды разработки VBA, которая представляет собой интегрированную среду разработчика с возможностями ввода и редактирования исходного кода приложения, создания форм, поддержки отладки и лёгкого доступа к этим возможностям в любой момент времени.

1.2 Обзор существующих информационных систем учета конфигурации компьютера

Чтобы получить сведения о конфигурации в текстовом файле, не обязательно устанавливать какое-либо специализированное обеспечение. В операционной системе Windows имеется диспетчер, с помощью которого можно экспортировать информацию о конфигурации ПК в текстовый документ. Для запуска обслуживающего утилита достаточно кликнуть «Пуск». После входа в «Пуск», перейти в последовательном порядке в папки «Программы», «Стандартные», «Служебные». После старта операции, спустя пару секунд, получите необходимые сведения, которые сможете поместить в текстовый документ, кликнув меню , опция . Затем укажите путь сохранения документа и имя.

Также в открытом доступе есть множество бесплатных утилит для определения конфигурации компьютера, которые имеют различные

положительные свойства. Протестировав несколько из них, можно выделить наиболее эффективные и быстродействующие утилиты, опираясь на следующие факторы: бесплатность, наличие русификатора, простой и удобный интерфейс, возможность сохранения сведений в текстовом файле, небольшой размер.

Vinaudit. Программа довольно проста в использовании, абсолютно бесплатна и не требует установки. После запуска всплывает окно, в котором четко и ясно описано, как получить конфигурацию. Помимо информации о технических показателях аппарата, программа собирает сведения об операционной системе и установленных программах.

System Information for Windows. Утилита (SIW) бесплатная, имеет небольшой размер и также не требует установки. Программа имеет приятный многофункциональный интерфейс с функцией подробной и конкретной информации о ПК.

CPU-Z. Данный обозреватель предназначен для отображения технических показаний персонального компьютера, то есть: видеокарта, процессор, ОЗУ (оперативная память) и свойства операционной системы Windows. Она абсолютно бесплатна и комфортна в использовании.

Everest. Наиболее функциональная программа, которая собирает сведения и выводит на экран информацию практически обо всем устройстве. не только предоставляет сведения о системных показателях, но и дает возможность пользователю тестировать комплектующие детали компьютера в реальном времени. Можно без проблем найти и определить состояние и возможности видеокарты, звуковой карты, процессора и т.д. Единственным недостатком является то, что бесплатная версия доступна в использовании на 30 дней.

AIDA64 — определение железа компьютера. AIDA64 — это самая популярная программа для обзора и диагностики всех особенностей компьютера. AIDA64 дает исчерпывающую информацию о составе компьютера: железо, программы, операционная система, сеть и подключаемые

устройства. AIDA64 считается самой авторитетной программой в своем классе и дает достоверную информацию о всех компонентах системы. Важной составляющей программы являются тесты для проверки производительности и стабильности работы компьютера. А еще AIDA64 может осуществлять мониторинг состояния системы, то есть, отображает текущую нагрузку на систему и т. п.

Тем не менее, для коммерческих предприятий данные программы не совсем удобны, так как не обладают полным необходимым функционалом.

1.3 Анализ информационной структуры

Программирование в Office — это прежде всего уменьшение количества повторяющихся действий (и ручной работы, которая для этого требуется). Вот примеры некоторых типичных ситуаций, когда применение программирования имеет смысл:

– с определенной периодичностью приходится изготавливать документы, очень похожие друг на друга: приказы, распоряжения в бухгалтерию, договоры, отчеты и т. п. Часто информацию можно взять из базы данных — тогда применение программирования может дать очень большой выигрыш во времени. Иногда ее приходится вводить вручную, но и тогда автоматизация даст выигрыш и во времени, и в снижении количества ошибок;

– разновидность той же ситуации: одни и те же данные нужно использовать несколько раз. Например, при заключении договора с заказчиком. Одни и те же данные (наименование, адрес, расчетный счет, номер договора, дата заключения, сумма и т. п.) могут потребоваться во множестве документов: самом договоре, счете, счете-фактуре, акте сдачи выполненных работ и так далее. Логично один раз ввести эти данные (скорее всего, в базу данных), а затем автоматически формировать (например, в Word) требуемые документы;

– когда нужно сделать так, чтобы вводимые пользователем данные автоматически проверялись. Вероятность ошибки при ручном вводе данных

зависит от многих разных факторов, но, согласно результатам некоторых исследований, она в среднем составляет около двух процентов. «Вылавливать» потом такие ошибки в уже введенных данных — очень тяжелый труд, поэтому сразу сделать так, чтобы они не возникали.

В общем, любое действие, которое приходится повторять больше нескольких раз — это возможный кандидат на автоматизацию. Например, занесение сотен контактов в Outlook, или замена ресурса в десятках проектов Project, или анализ информации из базы данных за разные периоды в таблице Excel — те ситуации, когда знание объектных моделей приложений Office спасет от часов и дней скучного труда.

Кроме того, применение программирования несет еще и другие преимущества для сотрудника, который применяет его в работе:

– повышается его авторитет в глазах руководства и других сотрудников;

– если программы этого пользователя активно используются на предприятии (им самим или другими сотрудниками), этим самым он защищает себя от сокращений, снижения зарплаты и т. п. — ведь поддерживать их и изменять в случае необходимости будет некому.

1.4 Функциональные требования к разрабатываемой информационной системе

В процессе работы с данным программным продуктом формируются и при необходимости распечатываются необходимые документы. Сначала сотрудник АО «Туапсинское АТП» заполняет базу технической возможности. Которая состоит из и адресов компьютеров. Затем, при запуске, программа формирует данные системной конфигурации каждой заданной машины. Требования к интерфейсам информационной системы

Цветовую палитру интерфейсов выдержать в стиле Windows. Разрешение экрана: 1024*768 — средний шрифт.

Интерфейс должен быть защищенным от неправильных действий пользователя.

Требования к техническому и программному обеспечению. Для ввода и корректировки данных в интерактивном режиме необходимо использовать несколько компьютеров, объединенных в локальную сеть.

Требования к программному обеспечению — операционная система Windows XP и более поздних версий, Microsoft Office не ниже 2007.

Требования к техническому обеспечению — компьютер Pentium 4, 1 ГГц или более поздних моделей стандартной конфигурации, лазерный принтер LaserJet 1100.

Глава 2 Инструменты и технологии реализации

2.1 Выбор средств офисного программирования

Язык VBA (Visual Basic for Applications) — это диалект языка Visual Basic, расширяющий возможности Visual Basic и предназначенный для работы с приложениями Microsoft Office и другими приложениями от Microsoft и третьих фирм.

При программировании в Office можно обойтись без языка VBA. Подойдет любой COM-совместимый язык, например, обычный Visual Basic, VBScript, JScript, C++, Delphi, Java и т. п. Можно использовать и .NET-совместимые языки программирования — VB.NET, C# и т. п. Все возможности объектных моделей приложений Office можно будет использовать.

Например, если сохранить следующий код в файле с расширением *.vbs и запустить его на выполнение, то будет запущен Word, в нем открыт новый документ и впечатан текст (рис. 2.1).

```
Dim oWord
Set oWord = CreateObject(«Word.Application»)
oWord.Visible = true
oWord.Documents.Add
oWord.Selection.TypeText («Привет от VBScript»)
```

Рисунок 2.1 Фрагмент листинга

Однако VBA обычно — самый удобный язык для работы с приложениями Office.

Главная причина проста — язык VBA встроен в приложения Office (и не только), и код на языке VBA можно хранить внутри документов приложений Office — документах Word, книгах Excel, презентациях PowerPoint и т. п. Конечно же, этот код можно запускать оттуда на выполнение, поскольку среда выполнения кода VBA (на программистском сленге — хост) встроена внутрь этих приложений.

В настоящее время VBA встроен:

- во все главные приложения MS Office — Word, Excel, Access, PowerPoint, Outlook, FrontPage, InfoPath;
- в другие приложения Microsoft, например, Visio и Microsoft Project;
- в более чем 100 приложений третьих фирм, например, CorelDraw и CorelWordPerfect Office 2000, AutoCAD и т. п.

У VBA есть также множество других преимуществ:

VBA — универсальный язык. Освоив его, можно не только получить ключ ко всем возможностям приложений Office и других, перечисленных выше, но и быть готовым к тому, чтобы:

- создавать полноценные приложения на Visual Basic (поскольку эти языки — близкие родственники);
- использовать все возможности языка VBScript (это — вообще урезанный VBA). В результате в распоряжении будут универсальные средства для создания скриптов администрирования Windows, для создания Web-страниц (VBScript в Internet Explorer), для создания Web-приложений ASP, для применения в пакетах DTS и заданиях на MS SQL Server, для создания серверных скриптов Exchange Server и многое-многое другое.

VBA изначально был ориентирован на пользователей, а не на профессиональных программистов (хотя профессионалы пользуются им очень активно), поэтому создавать программы на нем можно очень быстро и легко. Кроме того, в Office встроены мощные средства, облегчающие работу пользователя: подсказки по объектам и по синтаксису, макрорекордер и т. п.

При создании приложений на VBA, скорее всего, не придется заботиться о установке и настройке специальной среды программирования и наличии нужных библиотек на компьютере пользователя — MS Office есть практически на любом компьютере.

Несмотря на то, что часто приложения VBA выполняются медленно, они не ресурсоемки и очень хорошо работают, например, на сервере терминалов. Но, как правило, для программ на VBA особых требований про

производительности и нет: для написания игр, драйверов, серверных продуктов они не используется. По опыту, чаще всего проблемы с производительностью VBA-приложений — это не проблемы VBA, а проблемы баз данных, к которым они обращаются. Если проблемы действительно в VBA (обычно тогда, когда требуется сложная математика), то всегда есть возможность написать важный код на C++ и обращаться к нему как к обычной библиотеке DLL или встраиваемому приложению (Add-In) для Word, Excel, Access и т. п.

Программы на VBA по умолчанию не компилируются и поэтому вносить в них исправления очень удобно. Не нужно разыскивать исходные коды и перекомпилировать программы.

В среде программистов-профессионалов считается, что самый короткий путь «с нуля» и программ типа «Hello, World» до профессиональных программ, которые делаются под заказ — именно через связку Office — VBA (а конечно, не через C++, Java или Delphi).

2.2 Выбор макроредактора

В большинство программ Microsoft Office, таких, как Excel, Word, PowerPoint и т. п., встроено средство, которое позволяет создавать программы, вообще ничего не зная о программировании. Это средство называется макрорекордером.

Макрорекордер, как понятно из его названия — средство для записи макросов. Макрос — всего лишь еще одно название для VBA-программы, а макрорекордер — средство для его автоматического создания.

Принцип работы макрорекордера больше всего похож на принцип работы магнитофона: при нажатии на кнопку — начинается запись тех действий, которые выполняются. При нажатии на вторую кнопку — запись останавливается и её можно проиграть (то есть повторно выполнить ту же последовательность действий).

Конечно, макрорекордер позволяет написать только самые простые VBA-

программы. Однако и он может принести много пользы. Например, можно назначить клавиатурным комбинациям те слова, словосочетания, варианты оформления и т. п., которые часто приходится вводить (должность, название фирмы, продукт, ФИО директора и ответственного исполнителя и т. п.) — этим можно сэкономить много времени.

Как показывает опыт, подавляющее большинство обычных пользователей не знает о существовании макрорекордера, несмотря на то, что его применение позволило бы сэкономить им множество времени.

Перед созданием макроса в макрорекордере:

– необходимо очень тщательно спланировать макрос, хорошо продумав, что нужно будет делать и в какой последовательности. Если есть возможность, продумать подготовительные действия. Например, если нужно вставлять текущую дату в начало документа, может быть, есть смысл первой командой макроса сделать переход на начало документа (<Ctrl>+<Home>);

– посмотреть, нет ли готовой команды, которую можно сразу назначить клавише или кнопке в панели инструментов без изготовления макроса. Посмотреть можно при помощи меню Сервис → Настройка. На вкладке Команды можно перетащить нужную команду на требуемую панель управления, а нажав на этой же вкладке кнопку Клавиатура — назначить для команды нужную комбинацию клавиш;

– если необходимо при помощи макроса менять оформление текста, то правильнее вначале создать новый стиль с этим оформлением, а потом уже применять этот стиль к тексту. В этом случае можно обойтись без макроса, просто назначив стиль комбинации клавиш.

Как создать макрос в макрорекордере (для тех программ Microsoft Office, для которых такое средство предусмотрено, например, Word, Excel, PowerPoint, Project):

Приложения Microsoft Office 2007 по умолчанию настроены так, что не позволяют запускать макросы. Поэтому перед тем, как приступать к созданию макросов, в меню Сервис → Макрос → Безопасность необходимо переставить

переключатель в положение «Средняя» или «Низкая», а потом закрыть и снова открыть данное приложение. Это потребуется сделать только один раз (рис. 2.2—2.5).

В меню Сервис → Макрос выбрать команду Начать запись (рис. 2.6—2.7). В открывшемся окне потребуется определить:

Имя макроса. Правила такие: не должно начинаться с цифры, не должно быть пробелов и символов пунктуации. Максимальная длина в Excel — 64 символа, в Word — 80 символов. Можно писать по-русски.

Будет ли макрос назначен кнопке на панели управления или комбинации клавиш. Назначить макрос кнопке/комбинации клавиш и использовать другие средства для его вызова можно и потом.

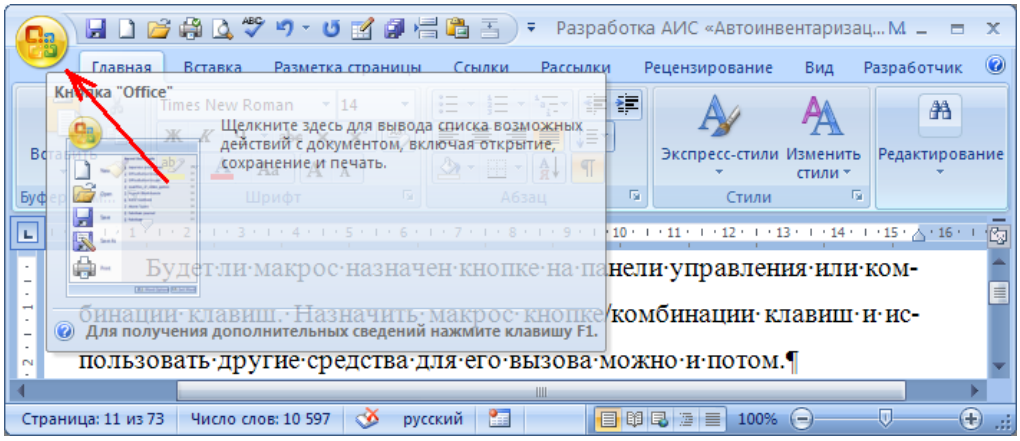

Рисунок 2.2 Настройка параметров безопасности для работы с макросами

В Word есть текущий файл и шаблон для всех вновь создаваемых документов — normal.dot. В Excel есть текущая книга, возможность создать макрос одновременно с созданием новой книги и личная книга макросов personal.xls (макросы из этой скрытой книги будут доступны в любых книгах).

После нажатия на кнопку OK или назначения кнопки/клавиатурной комбинации начнется запись макроса. Указатель мыши при этом примет вид магнитофонной кассеты и появится маленькая панель Остановить запись. На ней всего две кнопки — Остановить запись и Пауза. Если эта панель случайно была закрыта, остановить запись можно через меню Сервис → Макрос → Остановить запись.

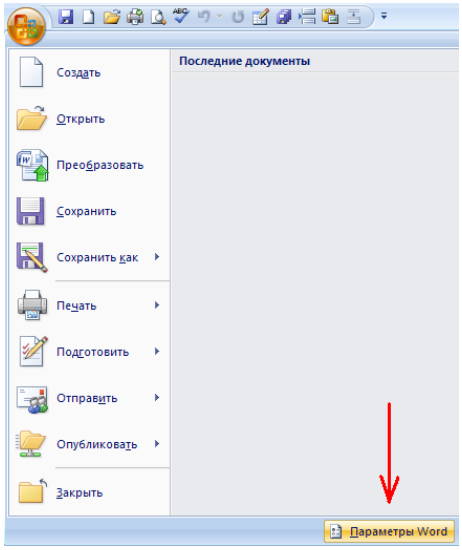

Рисунок 2.3 Настройка параметров безопасности для работы с макросами

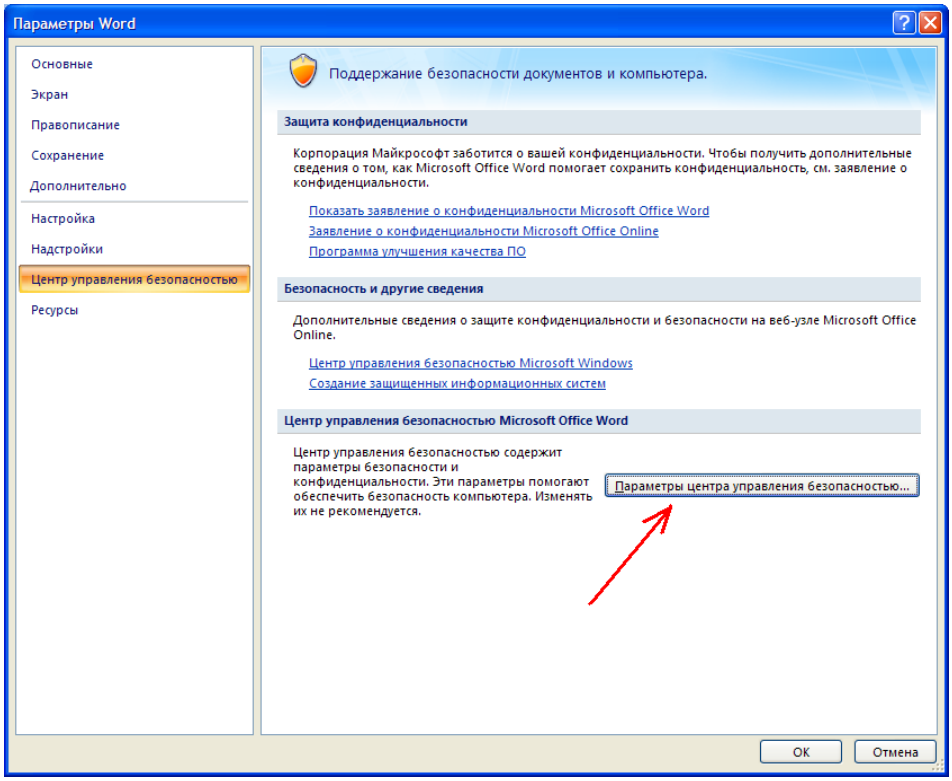

Рисунок 2.4 Настройка параметров безопасности для работы с макросами

Самый простой способ запустить макрос, которому не назначена кнопка или клавиатурная комбинация — в меню Сервис выбрать Макрос → Макросы (или нажать кнопку <Alt>+<F8>), в списке выбрать нужный макрос и нажать на

кнопку Выполнить. Из этого же окна можно просматривать/редактировать макросы, удалять/перемещать их и т. п.

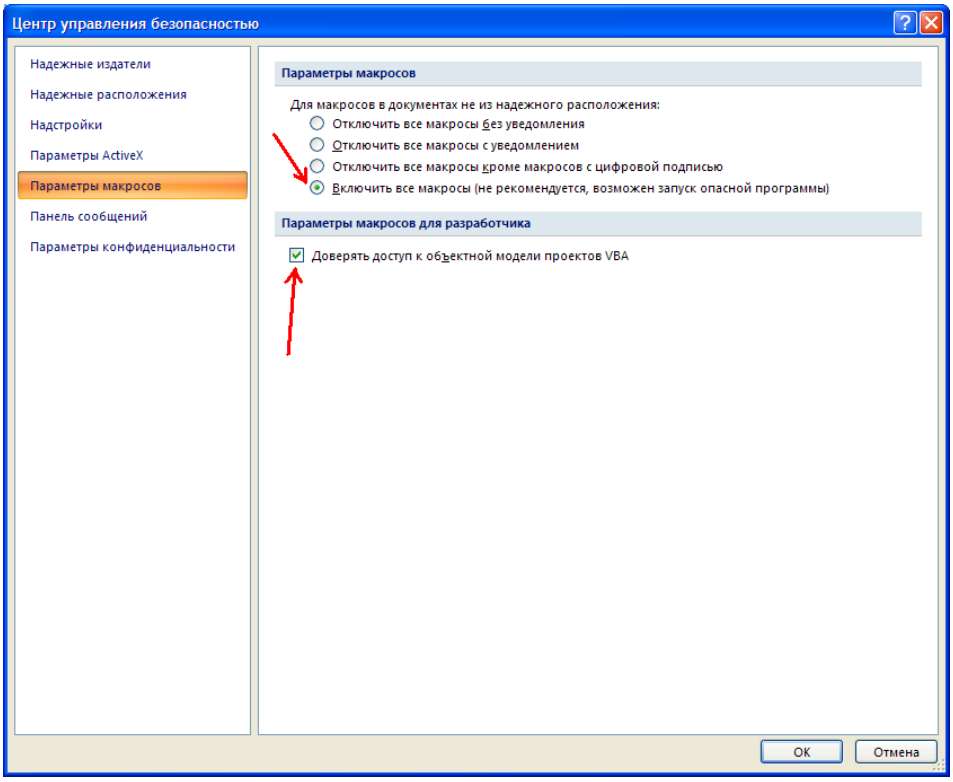

Рисунок 2.5 Настройка параметров безопасности для работы с макросами

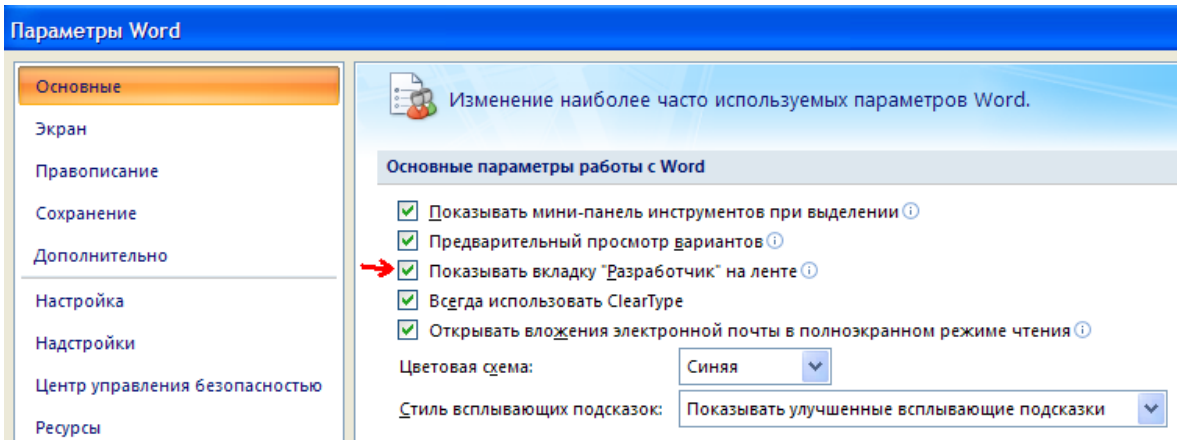

Рисунок 2.6 Обеспечение возможности записи макросов

Если макросов создано много, то получить список всех назначений клавиш (включая назначения для встроенных макросов Word) можно при помощи меню Сервис → Макрос → Макросы, затем в списке Макросы из выбрать команды Word и выбрать в списке Имя команду List Commands. В

ответ на приглашение нужно выбрать Текущие настройки меню и клавиш (иначе будет выведен полный список команд Word на 26 страниц). В документ будет вставлена таблица с текущими назначениями клавиш, которую можно распечатать.

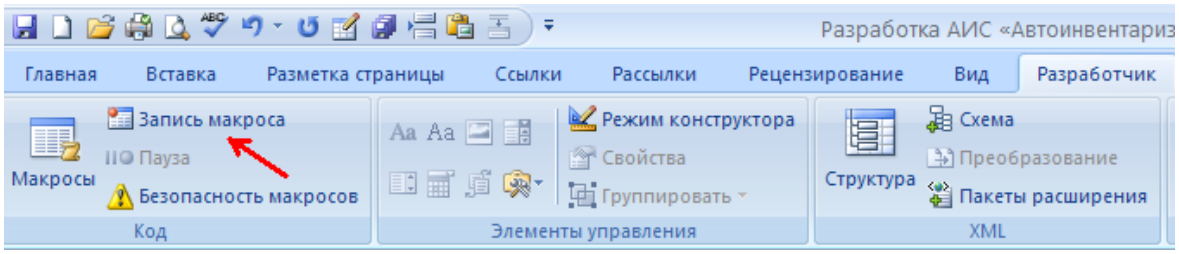

Рисунок 2.7 Обеспечение возможности записи макросов

Если уже есть значительное количество созданных при помощи макрорекордера макросов, то после освоения языка VBA есть смысл подумать над ними и, может быть, внести изменения. Чаще всего есть смысл подумать над следующими моментами:

– если в макросе повторяются какие-либо действия, возможно, есть смысл организовать цикл;

– может быть, есть смысл в ходе выполнения уточнить что-либо у пользователя (при помощи InputBox или элементов управления);

– чтобы в ходе выполнения макроса не возникало ошибок, возможно, есть смысл реализовать в нем проверку текущих условий.

И еще один очень важный момент, связанный с макрорекордером. Помимо того, что он позволяет создавать простенькие программы, пригодные для самостоятельного использования безо всяких доработок, макрорекордер — это еще и разведчик в мире объектных моделей приложений Office. Опытные разработчики часто пользуются им для того, чтобы понять, какие объекты из огромных объектных моделей приложений Office можно использовать для выполнения тех или иных действий.

Конкретный пример: нужно автоматизировать создание диаграмм в Excel. Поскольку в русской версии Excel для создания диаграммы вручную используется команду Вставка → Диаграмма, то скорее всего, в справке по

VBA нужно в первую очередь искать объект Diagram. И он там найдется — и скорее всего, будет потрачено определенное время на его изучение, прежде чем станет понятным, что это не та диаграмма. Объект Diagram представляет то, что в русской версии Excel называется «Схематическая диаграмма» (доступны из того же меню Вставка), а обычная диаграмма — это объект Chart. А вот если использовать разведчика (то есть создать диаграмму с записью в макрорекордере и посмотреть созданный код), он бы сразу указал нужное направление движения.

2.3 Служба WMI

WMI (Windows Management Instrumentarium) — это реализация Microsoft инициативы WBEM (Web-Based Enterprise Management), объявленной консорциумом фирм и направленной на снижение общей стоимости владения сетью. WMI используется для создания скриптов для полного управления Windows, в том числе:

- самой операционной системой;
- системными устройствами;
- Active Directory;
- реестром;
- счетчиками производительности;
- файловой системой и т. п.

WMI поставляется в составе Windows 2000, Windows XP и Windows Server 2003, на Windows 98, ME и NT его нужно предварительно устанавливать (дистрибутивы — на Web-сайте Microsoft).

Архитектура WMI состоит 4 главных компонентов (рис. 2.8—2.12).

Первый компонент — управляющие приложения. Как правило, это приложения или службы Windows, которые получают данные от WMI. Обычно они обеспечивают возможность взаимодействия пользователя с WMI. Роль управляющих приложений будут играть скрипты. Стандартные управляющие

приложения — SMS, MOM, Tivoli, набор инструментов, входящих в состав WMI SDK и т. п.

Второй компонент — управляемые объекты, то есть те объекты, доступ к которым можно получить при помощи WMI. Разнообразие управляемых объектов очень велико — от вентилятора, установленного в системного корпусе, до, например, SQL Server со всеми установленными базами данных.

Третий компонент — провайдеры WMI, то есть драйверы WMI, которые позволяют получить доступ к какому-либо классу объектов. Например, в стандартной поставке Windows присутствуют драйверы WMI для работы с реестром, журналом событий, аппаратными устройствами и т. п.

Четвертый компонент — программное обеспечение WMI (WMI software, winmgmt.exe) и репозитарий CIM (Common Information Model). Оба этих компонента отвечают за предоставление запрашиваемой средствами WMI информации и реакцию на вызов методов, однако у них существует разделение обязанностей: WMI software отвечает за динамическую информацию WMI (то, что непосредственно берется от операционной системы, оборудования и т. п.), а репозитарий CIM — за статическую информацию (настройки WMI на данном компьютере). Физически этот репозитарий расположен в файле CIM.REP в каталоге C:\WINNT\system32\wbem\Repository.

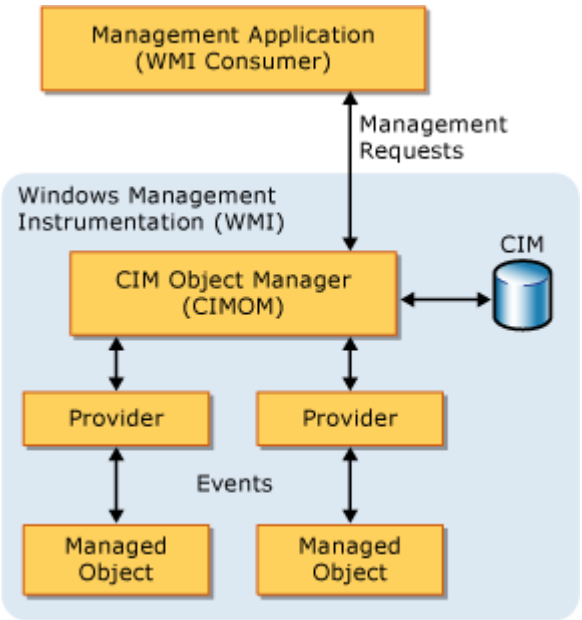

Рисунок 2.8 Архитектура WMI

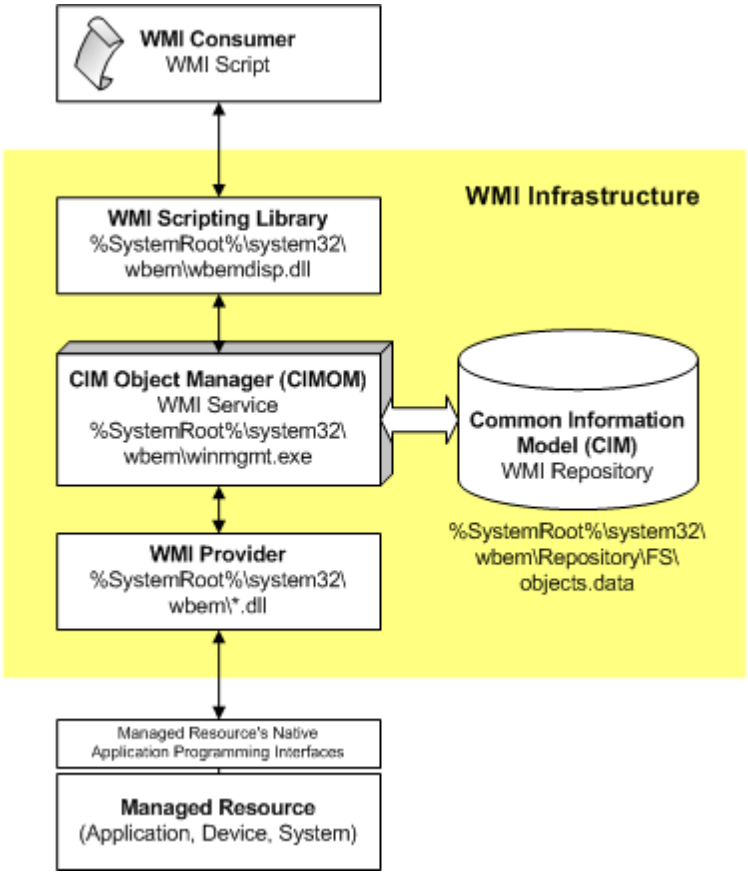

Рисунок 2.9 Архитектура WMI

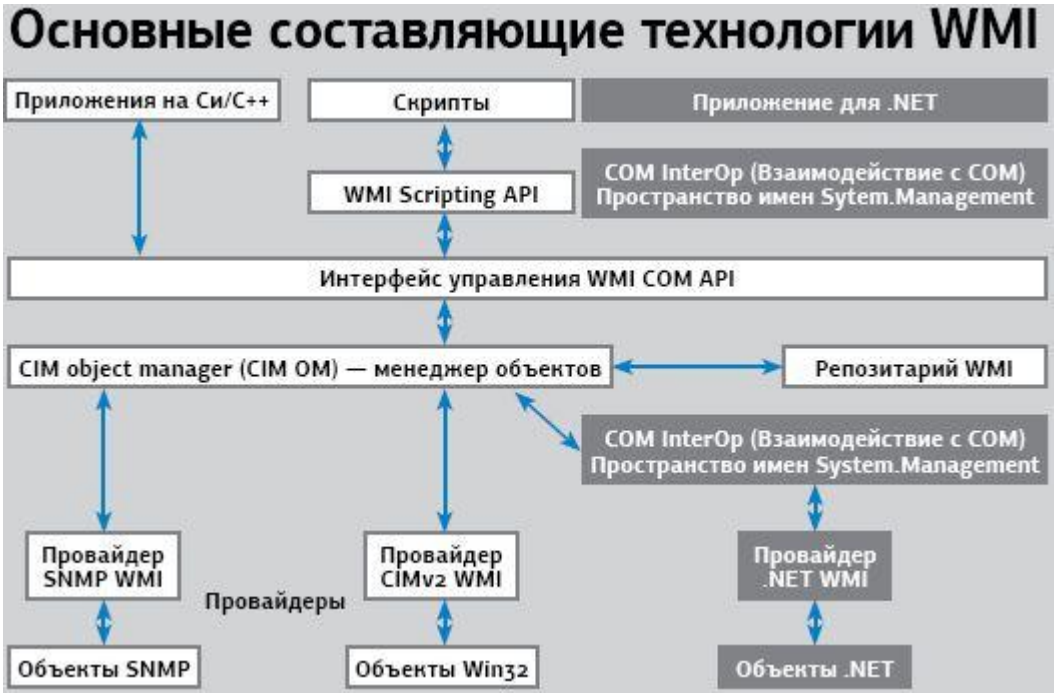

Рисунок 2.10 Архитектура WMI

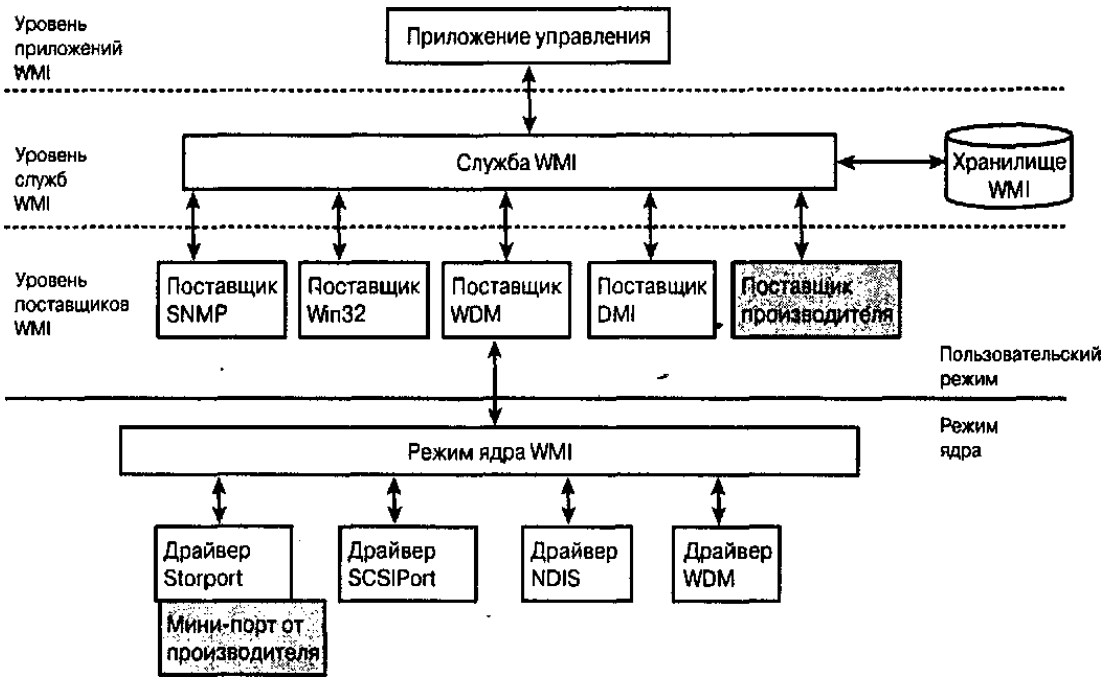

Рисунок 2.11 Архитектура WMI

Рисунок 2.12 Архитектура WMI

Основной набор стандартных средств для работы с WMI поставляется вместе с WMI SDK. В состав SDK входят:

– WMI CIM Studio — основная графическая утилита, предназначенная для просмотра пространств имен WMI, классов, их свойств и

методов. Можно использовать также для выполнения WQL-запросов;

– WMI Event Registration и WMI Event Viewer — эти средства можно использовать для регистрации событий WMI и просмотра происшедших событий (аналогично тому, как это реализовано, например, в Performance Monitor);

– WMI Object Browser — выделенная в отдельное средство часть WMI CIM Studio. Используется для просмотра классов WMI, их свойств и методов. Работать необходимо осторожно — если обратиться к большому набору экземпляров какого-либо класса (например, информации о службах), такой запрос будет выполняться очень долго, а прерывать выполнение аварийно трудно.

– Утилита mofcomp — она позволяет компилировать файлы MOF (Managed Object Format) из текстового в двоичный формат, а также добавлять их в репозитарий CIM. Используется, например, для того, чтобы обеспечить возможность обращения средствами WMI к приложению.

Еще одно средство для работы с WMI — консоль WMI Control. Она доступна, например, из Computer Management и предназначена в основном для администрирования службы WMI (резервное копирование и восстановление репозитария, настройки системы безопасности и т. п.)

Простое, но в некоторых ситуациях очень удобное средство — **Scriptomatic**, которое можно скачать с Web-сайта Microsoft. Это средство реализовано в виде приложения HTML (HTA) и предназначено для автоматического создания скрипта, выводящего информацию о значениях всех свойств всех экземпляров какого-либо класса WMI. Может использоваться как быстрый браузер по классам WMI (рис. 2.13—2.14).

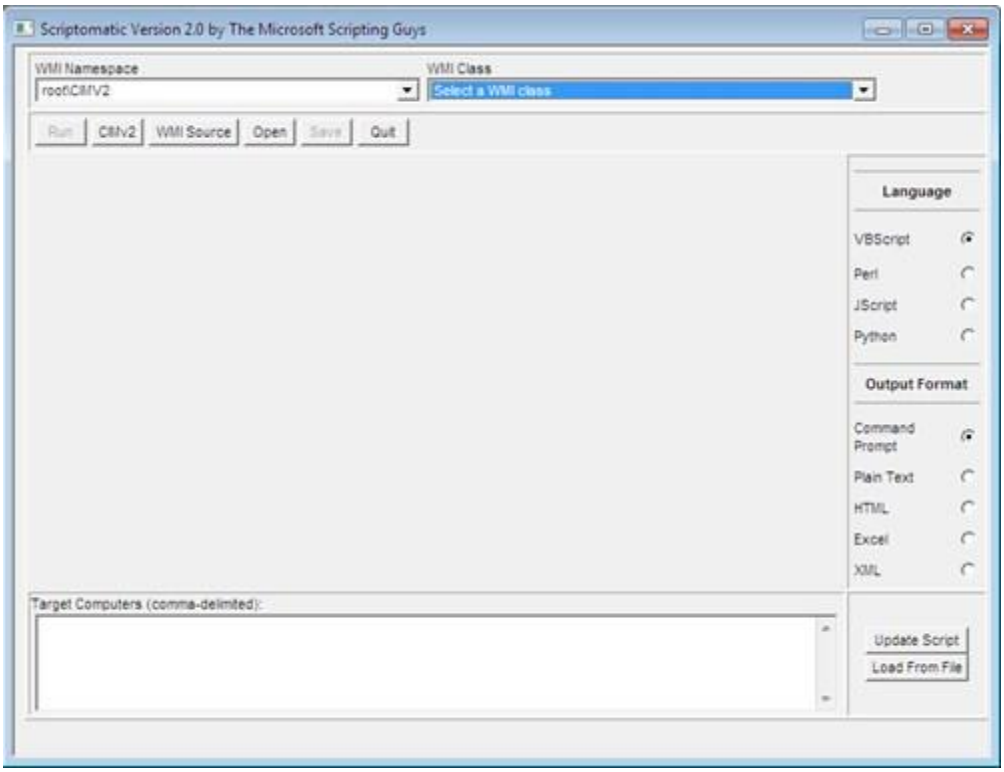

Рисунок 2.13 WMI Scriptomatic — инициализация

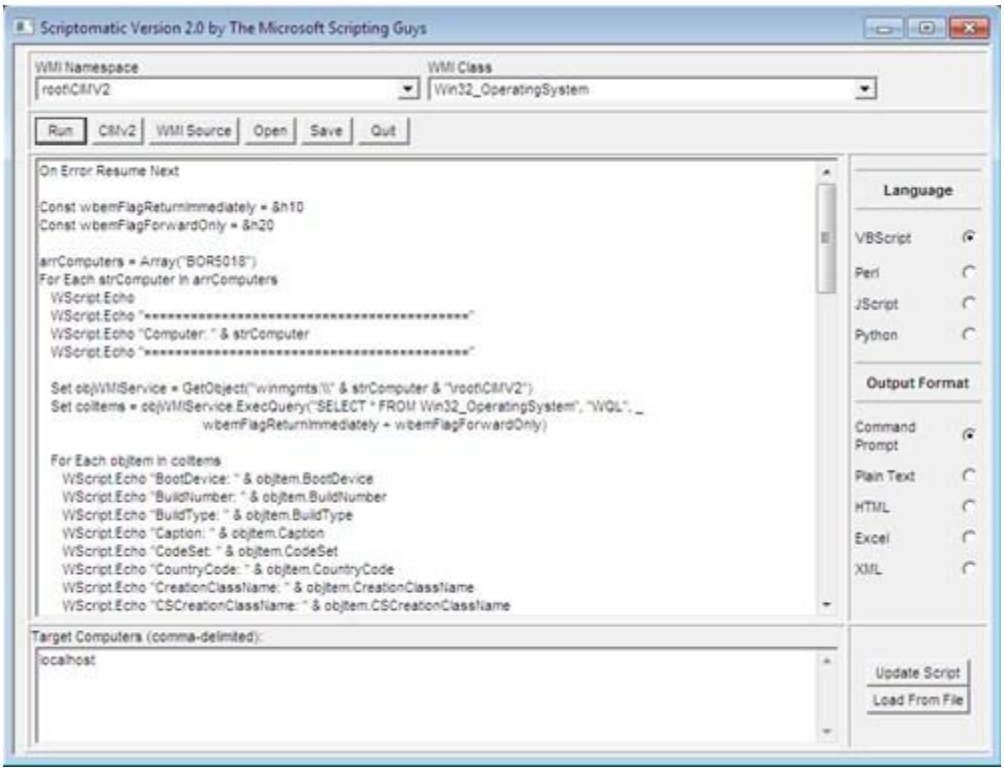

Рисунок 2.14 WMI Scriptomatic — Win32_OperatingSystem

Очень удобная возможность — установить ODBC-драйвер для WMI и получить возможность работать с WMI как с большой базой данных со

значительным количеством таблиц. Драйвер ODBC можно найти на дистрибутиве Windows 2000 Advanced Server в каталоге D:\VALUEADD\MSFT\MGMT\WBEMODBC. После установки появится системный источник данных под названием WBEM Source и справка по нему в виде файла Wbemdr32.chm. Главное — не заполнять имя пользователя и пароль для подключения, потому что при подключении возникнет ошибка.

Затем нужно нажать на кнопку OK и в окне Namespace Selection выбрать нужное пространство имен (как правило, CIMV2).

Несмотря на все перечисленные выше возможности, все-таки главный инструмент для работы с WMI (как записано в документации по WMI) — это скрипты.

Первое, что необходимо сделать в скрипте — подключиться к службе WMI на локальном или удаленном компьютере. Подключение можно выполнять двумя разными способами: при помощи объекта SwbemLocator или (более стандартный способ) при помощи так называемого моникера.

Первый способ выглядит так:

1. вначале получить объект SwbemLocator (рис. 2.15).

```
Dim oLocator
Set oLocator = CreateObject(«wbemScripting.Swbemlocator»)
```

Рисунок 2.15 Фрагмент листинга

У объекта SwbemLocator — всего лишь одно свойство и один метод — свойство security_ используется для получения ссылки на одноименный вложенный объект, и используется для настройки безопасности при подключении (если она отличается от параметров по умолчанию), а метод ConnectServer используется для подключения к службе WMI на локальном или удаленном компьютере. Если подключение осуществляется к службе WMI на локальном компьютере, воспользоваться правами другого пользователя (отличного от текущего) не удастся — таковы ограничения службы DCOM.

2. далее вызвать метод ConnectServer, который возвращает объект

SWbemServices (рис. 2.16).

```
Dim oServices
Set oServices = oLocator.ConnectServer(«LONDON», «CIMV2»)
```

Рисунок 2.16 Фрагмент листинга

Если необходимо подключиться к пространству имен по умолчанию (настраивается из консоли Computer Management, по умолчанию CIMV2) на локальном компьютере, то вызов метода может выглядеть так (рис. 2.17).

```
Set oServices = oLocator.ConnectServer()
```

Рисунок 2.17 Фрагмент листинга

3. полученный объект SWbemServices можно использовать для выполнения запросов WQL, получения ссылок на объекты WMI и т. п.

Второй способ подключения к WMI (практически стандартный) — использование моникера. Моникер — это специальная метка, которая используется для подключения к COM-объектам. При использовании моникера синтаксис получается более коротким и гибким (хотя, возможно, менее понятным).

Самый простой способ использования моникера — применение синтаксиса вида (рис. 2.18).

```
Dim oSvc
Set oSvc = GetObject(«WinMgmts:»)
```

Рисунок 2.18 Фрагмент листинга

В этом случае получается тот же объект SWbemServices с параметрами по умолчанию: то есть осуществляется подключение к пространству имен Root\CIMV2 на локальном компьютере с правами пользователя, от имени которого был запущен скрипт.

В PrimalScript подсказок по свойствам и методам нет.

Если они нужны, то можно использовать редактор VBA с синтаксисом вида (рис. 2.19).

```
Dim oSvc As SWbemServices
Set oSvc = GetObject(«WinMgmts:»)
```

Рисунок 2.19 Фрагмент листинга

В качестве дополнительных параметров при определении моникера можно указать уровень имперсонации (по умолчанию — GetObject(«WinMgmts:{impersonationLevel=impersonate}») , означает работу с правами текущего пользователя и региональные настройки подключения. Например, чтобы подключиться с английскими региональными настройками, можно использовать синтаксис вида GetObject («WinMgmts:[Locale = ms_409]»).

Далее моникеру можно передать информацию о том, к чему производится подключение:

1. имени компьютера;
2. пространству имен в нем;
3. объекту в этом пространстве имен.

Например, чтобы подключиться к компьютеру Vancouver, можно использовать синтаксис (рис. 2.20).

```
Set oSvc = GetObject(«WinMgmts://Vancouver»)
```

Рисунок 2.20 Фрагмент листинга

Чтобы подключиться на нем к пространству имен ROOT\CIMV2 (обычно работа производится только с ним, это же пространство имен выбирается по умолчанию, поэтому обычно пространство имен не указывается), синтаксис может выглядеть так (рис. 2.21).

```
Set oSvc = GetObject(«WinMgmts://Vancouver/root/cimv2»)
```

Рисунок 2.21 Фрагмент листинга

А чтобы подключиться на нем к объекту, представляющему систему логических дисков, синтаксис может быть таким (рис. 2.22).

```
Set oSvc = GetObject(«WinMgmts://Vancouver/root/cimv2:  Win32_LogicalDisk»)
```

Рисунок 2.23 Фрагмент листинга

Можно подключиться и напрямую к объекту логического диска, например, C:. Синтаксис при этом может выглядеть так (рис. 2.24).

```
Set oSWbemObject = GetObject(«WinMgmts://Vancouver/root/cimv2:
Win32_LogicalDisk.DeviceID='C:'»)
```

Рисунок 2.24 Фрагмент листинга

Но это уже получается запрос WQL. Кроме того, возвращается не объект SWbemServices, а объект SWbemObject.

Код вида (рис. 2.25).

```
Set oSvc = GetObject(«WinMgmts:\\Vancouver»)
```

Рисунок 2.25 Фрагмент листинга

также вполне допустим.

2.4 Выбор метода разработки

Microsoft рекомендует использовать только моникер, аргументируя это тем, что:

1. синтаксис проще;
2. можно возвращать не только объект SWbemServices, но и сразу объект SWbemObject.

Все примеры у Microsoft построены только на использовании моникера.

Однако при использовании объекта SWbemLocator имеются свои

преимущества:

1. только при использовании этого метода можно подключаться к компьютеру с правами другого пользователя. Требуется это не часто, но иногда все-таки требуется;
2. возможность работы с подсказками в PrimalScript сохраняется.

Объект SWbemObject можно получить при помощи всего лишь одного дополнительного шага — вызова метода Get объекта SWbemServices.

Подключаться с правами другого пользователя можно только к удаленному компьютеру. При попытке подключиться с новыми правами к службе WMI на локальном компьютере, возникнет ошибка.

После успешного подключения к службе WMI на локальном или удаленном компьютере (при помощи SWbemLocator или моникера), возвращается ссылка на объект SWbemServices (рис. 2.26).

```
Dim oLocator
Set oLocator = CreateObject(«wbemScripting.Swbemlocator»)
Dim oSvc
Set oSvc = oLocator.ConnectServer()
MsgBox TypeName(oSvc)
```

Рисунок 2.26 Фрагмент листинга

Этот объект — основной для работы с WMI. Для него предусмотрено всего одно свойство Security_ и 18 методов. Ниже приведены только наиболее часто используемые, по остальным можно получить информацию при помощи WMI SDK.

– AssociatorsOf — этот метод позволяет получить коллекцию объектов, ассоциированных с указанным вами. Вместо него можно воспользоваться командой WQL ASSOCIATORS OF
– ExecMethod — выполнить метод объекта WMI
– ExecNotificationQuery — выполнить запрос для получения событий

– ExecQuery — выполнить запрос WQL (обычно для получения коллекции объектов)

– Get — получить ссылку на указанный объект WMI

– InstancesOf — получить коллекцию всех экземпляров указанного класса

– SubclassesOf — получить коллекцию подклассов указанного класса.

Для многих методов предусмотрен также вариант Async — асинхронное выполнение данного метода.

В практической работе чаще всего требуется получить ссылку на нужный объект WMI при помощи метода Get или воспользоваться одним из методов Exec... Метод Get возвращает ссылку на объект SWbemObject, представляющий указанный объект WMI. Синтаксис может быть таким (рис. 2.27).

```
Dim oWbemObj
Set oWbemObj = oSvc.Get(«Win32_LogicalDisk.DeviceID='C:'»)
```

Рисунок 2.27 Фрагмент листинга

Объект SWbemObject — это не сам нижележащий объект WMI, а всего лишь оболочка, в которую одет этот объект. Например, у полученного объекта SWbemObject нет ни свойства FreeSpace, ни свойства FileSystem, ни свойства VolumeName. Зато при помощи свойства Properties_ этого объекта можно получить информацию о всех свойствах нижележащего объекта, а при помощи свойства Methods_ — о всех методах. Чтобы случайно не перепутать свойства SWbemObject и нижележащего объекта, названия всех свойств и методов SWbemObject заканчиваются на подчеркивание.

Система WMI изначально построена так, чтобы она была в определенной степени самодокументируемой — например, чтобы информацию о всех свойствах и методах объектах WMI можно было получить не при помощи просмотрщика объектов, а просто из кода скрипта. Например, чтобы получить информацию о всех свойствах объекта WMI, представляющего диск «C:»,

можно использовать код вида (рис. 2.28).

```
For Each item In oWbemObj.Properties_
WScript.Echo item.Name & vbTab & item.Value
Next
```

Рисунок 2.28 Фрагмент листинга

а для получения информации о методах — такую же конструкцию (рис. 2.29).

```
For Each item In oWbemObj.Methods_
WScript.Echo item.Name
Next
```

Рисунок 2.29 Фрагмент листинга

В первом случае свойство Properties возвращает стандартную коллекцию SWbemPropertySet с обычными свойствами и методами типа Add, Remove, Item, Count, которая состоит из объектовSWbemProperty. Важнейшие свойства SWbemProperty таковы:

– Value — значение реального свойства. Это свойство используется по умолчанию;

– Name — комментариев не требует;

– IsArray — работает ли это свойство с массивом значений или нет;

– Qualifiers — позволяет копнуть на уровень глубже и получить ссылку на коллекцию SWbemQualifierSet, коллекцию допустимых значений для этого свойства (если для него такой набор предусмотрен).

Для методов все устроено точно также: свойство Methods_ возвращает коллекцию SWbemMethodSet, состоящую из объектов SWbemMethod. Важнейшие свойства этого объекта — Name, Qualifiers, InParameters,OutParameters.

Создатели WMI настоятельно рекомендуют работать с объектами WMI

через их оболочку — SWbemObject. Например, чтобы получить информацию о значении свойства Freespace для диска C:, можно использовать синтаксис вида (рис. 2.30).

```
For Each item In oWbemObj.Properties_
If item.Name = «FreeSpace» Then MsgBox item.value/1024/1024
Next
```

Рисунок 2.30 Фрагмент листинга

а чтобы поменять значение свойства, можно воспользоваться кодом вида (рис. 2.31).

```
For Each item In oWbemObj.Properties_
If item.name = «VolumeName» Then item.value = «System»
Next
oWbemObj.put_
```

Рисунок 2.31 Фрагмент листинга

Вызов метода put_ здесь обязателен, иначе значение изменится только в оперативной памяти, а не на компьютере.

Для вызова метода объекта WMI рекомендуется воспользоваться методом ExecMethod_ (рис. 2.32).

```
Set process = GetObject(«winmgmts: {impersonationLevel =
impersonate}!Win32_Process»)
Set method = process.Methods_(«Create»)
Set inParam = method.inParameters.SpawnInstance_()
inParam.CommandLine = «calc.exe»
Set outParam = process.ExecMethod_(«Create», inParam)
```

Рисунок 2.32 Фрагмент листинга

Здесь вызывается метод Create объекта process (типа SWbemObject) и ему

передаётся параметр CommandLine. Параметр — тот же самый объект SWbemObject, который создается через коллекцию inParameters.

Несмотря на рекомендации Microsoft, большинство разработчиков обращаются к свойствам и методов объектов WMI не через коллекции Properties_ и Methods_ объекта SWbemObject, а напрямую. Для этого используется тот же самый SWbemObject.

Синтаксис обращения к свойству при этом может выглядеть так (рис. 2.33).

```
MsgBox oWbemObj.VolumeName
```

Рисунок 2.33 Фрагмент листинга

а синтаксис вызова метода (рис. 2.34).

```
Dim oLocator
Set oLocator = CreateObject(«wbemScripting.Swbemlocator»)
Dim oSvc
Set oSvc = oLocator.ConnectServer()
Dim oWbemObj
Set oWbemObj = oSvc.Get(«Win32_Process»)
oWbemObj.Create («calc.exe»)
```

Рисунок 2.34 Фрагмент листинга

Несмотря на то, что свойства и методы нижележащего метода не видны из просмотрщика, они вполне доступны.

По умолчанию все методы в скриптах WMI вызываются синхронно, то есть пока не завершится выполнение одного метода, выполнение скрипта дальше не пойдет. Такое поведение наиболее логично и надежно, но иногда такой подход неприемлем с точки зрения производительности. При выполнении ресурсоемких операции, которые производятся сразу на многих компьютерах (например, получение десятков тысяч сообщений из журналов

ошибок) выполнение операций желательно распараллелить, чтобы получить выигрыш во времени. Для этого методы можно запускать асинхронно, настроив предварительно соответствующий приемник события. Подробно об этом — в статье «Making an Asynchronous Call Using The Scripting API» в документации WMI SDK.

Однако у асинхронных методов — свои недостатки. При использовании асинхронных вызовов все данные накапливаются в области, принадлежащей службе WMI и передаются ей только после полного их накопления. В результате:

– скрипт, который мог бы обрабатывать поступающие данные, простаивает;

– если данных очень много, то их скопление может вызвать ошибки службы WMI, связанные с нехваткой памяти.

Поэтому есть еще один вариант вызова методов — полусинхронный. Подробнее о нем — в статье «Making a Semisynchronous call» в документации. Основное преимущество использования полусинхронного метода — данные сразу же передаются вызвавшему метод скрипту (без накопления), основной признак использование такого подхода — использование для флагов «Wbem Flag Return Immediately» и «Wbem Flag Forward Only» вместе (получается значение 48, которое часто фигурирует в примерах скриптов). Для работы с асинхронными и полусинхронными операциями используется объект SWbemSink.

2.5 Язык запросов WQL

Основная функциональность WMI заключена в свойствах и методах объектов этой библиотеки. Особенность работы с ними заключаются в том, как можно найти нужный объект.

Самый простой вариант — получить коллекцию всех объектов определенного класса. Для этой цели можно использовать метод InstancesOf

объекта SWbemServices. Этот метод всегда возвращает коллекцию, даже если она состоит только из одного объекта. Для того, чтобы получить доступ к элементам этой коллекции, проще всего использовать конструкцию For ... Each (рис. 2.35).

```
Dim oLocator
Set oLocator = CreateObject(«wbemScripting.Swbemlocator»)
Dim oSvc
Set oSvc = oLocator.ConnectServer()
Dim oCol
Set oCol = oSvc.InstancesOf(«win32_product»)
For Each item In oCol
WScript.Echo item.Name
Next
```

Рисунок 2.35 Фрагмент листинга

Отфильтровать нужный объект можно, например, так (рис. 2.36).

```
Dim oCol
Dim oProductOffice
Set oCol = oSvc.InstancesOf(«win32_product»)
For Each item In oCol
If (InStr(item.name, «Microsoft Office») <> 0) Then
Set oProductOffice = Item
End If
Next
MsgBox oProductOffice.Name
```

Рисунок 2.36 Фрагмент листинга

Однако, конечно, с точки зрения производительности такой подход трудно признать самым правильным. Приходится помещать ссылки на все

установленные продукты в коллекцию и перебирать ее, чтобы найти нужный. Намного удобнее воспользоваться встроенным в WMI языком запросов — **WQL** (WMI Query Language, другое название — SQL for WMI). Этот язык позиционируется как ANSI SQL-совместимый, хотя ограничений у него очень много (например, при помощи него нельзя изменять данные — операторов INSERT, UPDATE, DELETE в нем не предусмотрено).

В WQL предусмотрено три типа запросов:

– запросы к данным;

– запросы к событиям;

– запросы к структуре WMI — они позволяют информацию о структуре классов WMI.

Общий синтаксис запроса WQL выглядит так:

SELECT свойства FROM имя_класса WHERE свойство оператор значение

Если использовать запрос WQL для получения информации об установленных продуктах, то в самом простом варианте он может выглядеть так (рис. 2.37).

```
Dim oCol
Set oCol = oSvc.ExecQuery(«select * from Win32_Product»)
For Each item In oCol
WScript.Echo item.Name
Next
```

Рисунок 2.37 Фрагмент листинга

Если, как в обычном SQL, указать только нужные столбцы, вернется та же коллекция объектов SWbemObject, но только с указанными свойствами плюс свойство, которое определено как ключевое. В примере ниже вернется явно указанное свойство Version и ключевое свойство Name. Если попробовать обратиться еще и к свойству InstallState, возникнет ошибка (рис. 2.38).

```
Dim oCol
Set oCol = oSvc.ExecQuery(«select version from Win32_Product»)
For Each item In oCol
Wscript.Echo item.Version
WScript.Echo item.Name
WScript.Echo TypeName(item)
Next
```

Рисунок 2.38 Фрагмент листинга

Выражение FROM в WQL-запросе — это имя класса, коллекцию экземпляров которого нужно получить.

Выражение WHERE определяет фильтр в запросе — таким образом в коллекции остаются только те объекты, которые нужны (рис. 2.39).

```
Dim oCol
Set oCol = oSvc.ExecQuery(«select * from Win32_Product WHERE description = _
' Microsoft Office — профессиональный выпуск версии 2003'»)
For Each item In oCol
WScript.Echo item.Name
Next
```

Рисунок 2.39 Фрагмент листинга

С применением фильтра WHERE связаны некоторые особенности, которые необходимо учитывать:

— по умолчанию в коллекцию-результат запроса попадают все экземпляры указанного класса вместе с их подклассами. Если подклассы не нужны, то фильтр WHERE может выглядеть так:

SELECT * FROM CIM_MediaAccessDevice WHERE __CLASS = _ 'Win32_CDROMDrive'

Перед словом __CLASS стоит двойное подчеркивание.

— в фильтре WHERE используются стандартные операторы SQL: =, <, >,

<>, <=, >=. У операторов IS и IS NOT — специальное назначение: они используются только для сравнения значения свойства с NULL:

SELECT * FROM Win32_Fan WHERE Description IS Null

Запросы WQL можно тестировать из CIM Studio из поставки WMI SDK.

2.6. Работа с событиями

Внутри службы WMI реализована служба работы с событиями — WMI Event Service. Эта служба поддерживает фильтрацию событий и обеспечение реакции на них.

Скрипт, приложение или просто администратор (при помощи утилиты CIM Event Registration) могут зарегистрировать подписку на события, выступив в роли построителя подписки — Subscriber Builder. События представляются в виде системного класса _Event, на появление которых реагирует подписка.

Подписка (subscription) — определение события, представляющего интерес для построителя события. Фактически подписка — это требование для службы WMI выполнить определенные действия, если произойдет событие, подпадающее под условие подписки. Подписка создается при помощи специального WQL-запроса (SELECT). Настоятельно рекомендуется определить в этом запросе максимально строгие условия — чтобы избежать обработки ненужных событий.

Еще один компонент архитектуры работы с событиями — получатель события, Event Consumer. Этот тот блок кода в скрипте (или приложении), который получает уведомление о наступлении события и реагирует на него. Получатели могут быть временными и постоянными.

Временные получатели — это получатели, которым будут передаваться оповещения о событиях только тогда, когда этот скрипт/приложение запущен. Временные получатели обычно регистрируют свои события при помощи вызова метода ExecNotificationQuery.

Информация о постоянных получателях хранится в репозитарии CIM.

При наступлении события служба WMI проверяет, запущено ли требуемое приложение/скрипт, которое должно быть получателем. Если оно запущено, то оно оповещается, а если нет — то вначале служба WMI производит запуск этого приложения. В WMI SDK поставляется такой постоянный получатель — WMI Event Viewer Tool. Например, чтобы зарегистрировать постоянного получателя событий WMI (например, для появления новых сообщений в журнале приложений), нужно выполнить следующие действия:

— записать информацию о приложении Event Viewer в репозитарий CIM. Для этого необходимо выполнить в командной строке команду (рис. 2.40) mofcomp c:\winnt\system32\wbem\eviewer.mof (файл eviewer.mof устанавливается вместе с WMI SDK);

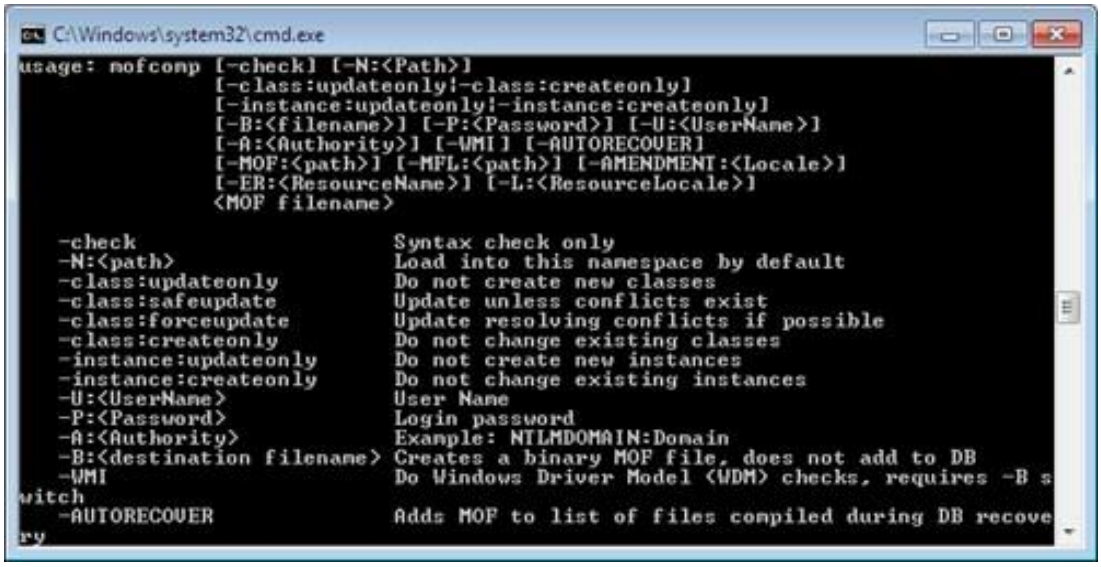

Рисунок 2.40 Компилятор MOF-файлов для расширения структуры WMI

— зарегистрировать нового Event Consumer для службы WMI. Для этого необходимо открыть WMI Event Registration, подключиться к предлагаемому по умолчанию пространству имен root\cimv2 и в ниспадающем списке слева выбрать Consumers. Далее раскрыть контейнер _EventConsumer → EventViewerConsumer, щелкнуть правой кнопкой мыши по EventViewerConsumer и в контекстном меню выбрать New Instance. Для создаваемого экземпляра достаточно заполнить только поле Name. После нажатия OK созданный получатель появится в дереве Event Registration;

— затем создать фильтр для интересующего события. Для этого в ниспадающем списке нужно выбрать Filters, щелкнуть правой кнопкой мыши по строке _EventFilter и точно так же в контекстном меню выбрать NewInstance. Для создаваемого объекта фильтра нужно заполнить следующие поля:

— Name (например, TestFilter);

— Query (например, для перехвата событий журнала приложений) — Select * from __InstanceCreationEvent Where TargetInstance ISA 'Win32_NTLogEvent') QueryLanguage — WQL;

— последнее, что осталось сделать — зарегистрировать для этого фильтра приемник события. Для этого в правой части экрана щелкнуть правой кнопкой мыши по созданному объекту получателя и в контекстном меню выбрать Register. В столбце Reg должен появиться зеленый флажок: получатель и фильтр настроены для приемы событий.

Для проверки созданной конфигурации достаточно выполнить короткий скрипт, который сгенерирует сообщений в журнал событий приложений, например (рис. 2.41).

```
Dim oShell
Set oShell = CreateObject(«Wscript.Shell»)
oShell.LogEvent 0, «TestMessage»
```

Рисунок 2.41 Фрагмент листинга

Если все сделано правильно, то появится окно WMI Event Viewer с информацией о происшедшем событии.

Любое событие в WMI представляется в виде экземпляра класса _Event, однако все события разделяются на три основные категории:

— intrisinc events — события, которые встроены в модель CIM и к ним можно обращаться сразу же (никаких драйверов устанавливать не надо). Систему таких событий изменять нельзя — они заранее предопределены и встроены в WMI;

— extrinsic events — внешние события, для отслеживания которых

необходимы специальные драйверы — extrinsic event providers. Для работы с ними необходимо, помимо установки таких драйверов, еще и поместить классы таких событий в репозитарий WMI. Вместе с WMI поставляются два внешних провайдера — для реакции на события SNMP и для отслеживания событий реестра

— timer events — события таймера, самые простые события для перехвата. Можно использовать как события абсолютного времени (сработает в указанное время), так и относительные (через полчаса после запуска скрипта).

Для того, чтобы зарегистрировать событие и иметь возможность на него реагировать, необходимо при помощи метода ExecNotificationQuery() объекта SWbemServices выполнить запрос WQL по специальной форме, например (рис. 2.42).

```
Dim oE
Set oE = oSvc.ExecNotificationQuery_
(«SELECT * FROM __InstanceModificationEvent » & _
«WITHIN 10 WHERE TargetInstance ISA 'Win32_Account'»)
```

Рисунок 2.42 Фрагмент листинга

oE в данном случае — объект SWbemEventSource.

Событие __Instance Modification Event (с двумя подчеркиваниями) — одно из девяти событий, на которые можно настроить реакцию:

— __Class Creation Event, __Class Deletion Event, __Class Modification Event — соответственно, создание, удаление и изменение класса;

— __Instance Creation Event, __Instance Deletion Event, __Instance Modification Event — создание, удаление и изменение объекта;

— __Namespace Creation Event, __Namespace Deletion Event, __Namespace Modification Event — то же самое для пространства имен (используется редко).

В данном случае отслеживаются изменения, которые вносятся в уже существующие объекты учетных записей Windows, поэтому используется

событие __InstanceModificationEvent.

Оператор WITHIN 10 говорит о том, что опрос будет производиться каждые 10 секунд (чтобы не расходовать лишние системные ресурсы). К этому оператору можно добавлять выражение GROUP BY: сгруппированные по какому-то признаку события за период рассматриваются как одно событие и HAVING — порог на количество событий. Если количество событий ниже этого порога, то событие считается не наступившим. Полный синтаксис запроса с использованием всех этих операторов может выглядеть, например, так (рис. 2.43).

```
SELECT * FROM __InstanceModificationEvent _
WITHIN 10 WHERE TargetInstance ISA 'Win32_PrintJob' _
GROUP WITHIN 30 BY TargetInstance.Owner _
HAVING NumberOfEvents > 5
```

Рисунок 2.43 Фрагмент листинга

Последняя часть запроса — WHERE TargetInstance ISA 'Win32_Account' позволяет отфильтровать источник событий.

Настоятельно рекомендуется определять его как можно более точно. В данном случае оператор ISA используется, чтобы указать, что интересуют все вложенные классы класса Win32_Account (это будут классы Win32_SystemAccount, Win32_Group и Win32_UserAccount).

Если нужно включить в запрос дополнительные условия, они включаются через AND (рис. 2.44).

```
SELECT * FROM __InstanceModificationEvent _
WHERE TargetInstance ISA 'Win32_PrintJob' _
AND TargetInstance.Owner = 'IvanIvanov'
```

Рисунок 2.44 Фрагмент листинга

TargetInstance — единственное свойство объекта SWbemEventSource. Оно

позволяет получить ссылку на объект, который послужил источником события и, следовательно, обратиться к любому из его свойств.

После выполнения запроса получается объект SWbemSource. Далее можно вызывать его единственный метод NextEvent. Этот метод ожидает появления события и при ее появлении возвращает стандартный объект SWbemObject, который представляет пойманное событие. А чтобы добраться до объекта, который и вызвал это событие, как обычно для SWbemObject, используется одно из его невидимых свойств — TargetInstance. Чтобы опрос происходил постоянно, есть смысл поместить этот код в бесконечный цикл (рис. 2.45).

```
Dim oLocator, oSvc, oEventSource, i, oWbemObject
Set oLocator = CreateObject(«wbemScripting.Swbemlocator»)
Set oSvc = oLocator.ConnectServer()
Set oEventSource = oSvc.ExecNotificationQuery _
(«SELECT * FROM __InstanceModificationEvent» & _
« WITHIN 1 WHERE TargetInstance ISA 'Win32_Account'»)
i = 0
Do While i = 0
Set oWbemObject = oEventSource.NextEvent
Wscript.Echo oWbemObject.TargetInstance.Name
Loop
```

Рисунок 2.45 Фрагмент листинга

В этом случае при любом внесении изменений в учетные записи на компьютере, к которому есть подключение (если этот компьютер — контроллер домена, то при любом внесении изменений в домен), в стандартную консоль вывода будет выводиться имя этой учетной записи.

2.7 Классы WMI

Классов в WMI предусмотрены сотни. Все классы поделены на три большие группы:

– классы, относящиеся к оборудованию;

– классы, относящиеся к операционной системе;

– классы, относящиеся к установленным приложениям.

Эти классы позволяют не только получать пассивную информацию, но и управлять компьютером.

Классы WMI для работы с оборудованием можно разделить на девять больших категорий:

– классы для работы с устройствами охлаждения (Win32 _ Fan, Win32 _ TemperatureProbe и т. п.);

– для работы с устройствами ввода (Win32 _ Keyboard, Win32 _ PointingDevice);

– с дисками (Win32 _ CDROMDrive, Win32 _ DiskDrive, Win32 _ FloppyDisk);

– с материнской платой / контроллерами / портами (Win32 _ MotherboardDevice, Win32 _ BIOS, Win32 _ PhysicalMemory, Win32 _ Processor и т. п.);

– с сетью (главный — Win32 _ NetworkAdapter);

– с питанием (Win32 _ Battery, Win32 _ PowerManagementEvent);

– с печатью (Win32 _ Printer, Win32 _ PrintJob);

– с модемами (Win32 _ POTSModem);

– с видеоподсистемой (Win32 _ DesktopMonitor, Win32 _ DisplayConfiguration, Win32 _ VideoController, Win32 _ VideoSettings).

Практически все эти классы предназначены для возможности только получения информации на чтения, без возможности внесения изменений. Исключение составляют только:

– класс Win32 _ Fan — возможность программным образом устанавливать скорость вращения вентилятора (если такая возможность

поддерживается материнской платой);

– класс Win32 _ LogicalDisk — при помощи него можно менять имя тома (свойство Volume Name);

– класс Win32 _ NetworkAdapterConfiguration. У этого класса — десятки методов, которые позволяют менять сетевые настройки для адаптера программным образом.

Например, чтобы настроить компьютер на использование DHCP, можно использовать код вида (рис. 2.46).

```
Set colNetAdapters = objWMIService.ExecQuery _
 («Select * from Win32_NetworkAdapterConfiguration where IPEnabled=TRUE»)
For Each objNetAdapter In colNetAdapters
 errEnable = objNetAdapter.EnableDHCP()
 If errEnable = 0 Then
 Wscript.Echo «DHCP has been enabled.»
 Else
 Wscript.Echo «DHCP could not be enabled.»
 End If
Next
```

Рисунок 2.46 Фрагмент листинга

Чтобы наоборот, настроить нужный статический IP-адрес, можно использовать код (рис. 2.47).

При помощи других методов этого класса можно настраивать домен DNS, параметры работы с серверами DNS и WINS, IPSec и фильтрацию пакетов TCP/IP, добавлять дополнительные сетевые протоколы, службы и клиенты — в общем, все, что доступно на графическом экране или через реестр.

Поскольку одна из целей WMI — обеспечить полный контроль над операционной системой, набор классов в этом разделе очень большой (несколько сотен). Большая часть этих классов (как и классов для работы с оборудованием) предназначена для получения информации и их свойства

доступны только на чтение. Также есть классы, у которых существуют методы, при помощи которых можно программно управлять компьютером.

```
Set colNetAdapters = objWMIService.ExecQuery _
 («Select * from Win32_NetworkAdapterConfiguration where IPEnabled=TRUE»)
strIPAddress = Array(«192.168.1.141»)
strSubnetMask = Array(«255.255.255.0»)
strGateway = Array(«192.168.1.100»)
strGatewayMetric = Array(1)
For Each objNetAdapter in colNetAdapters
 errEnable = objNetAdapter.EnableStatic(strIPAddress, strSubnetMask)
 errGateways = objNetAdapter.SetGateways(strGateway, strGatewaymetric)
 If errEnable = 0 Then WScript.Echo «The IP address has been changed.»
 Else WScript.Echo «The IP address could not be changed.»
 End If: Next
```

Рисунок 2.47 Фрагмент листинга

Классы Win32 _ BaseService, Win32 _ Service и Win32 _ SystemDriver представляют службы Windows: Win32 _ Services — обычные, Win32 _ SystemDriver — драйверы (на самом деле большой разницы между службами и драйверами для Windows нет и информация о них хранится в одной ветви реестра), Win32 _ BaseServices — и то, и другое вместе. У эти классов есть важные методы:

— Change — возможность изменить любые параметры службы: режим запуска, отображаемое имя, имя пользователя/пароль, от имени которого будет запускаться эта служба, даже заменить исполняемый файл службы (указать новый путь к нему). Поскольку чаще всего меняется режим запуска (например, Internet Connection Sharing на всех компьютерах можно перевести в режим Disabled), то для него предусмотрен отдельный метод ChangeStartMode;

— Create — возможность из скрипта создать новую службу на компьютере, Delete — соответственно, ее удалить (то же самое можно сделать

при помощи утилит Resource Kit);

— PauseService, ResumeService, StopService, StartService — изменить состояние службы. Проверить, можно ли службу остановить или приостановить, можно при помощи свойств AcceptStop и AcceptPause.

Наиболее часто встречающиеся ситуации для работы со службами:

— необходимо централизованно изменить пароль для всех служб, работающих от имени определенной учетной записи;

— перевести какую-либо службу (чаще всего ICS и Server) в режим Disabled;

— отследить состояние службы и в случае необходимости ее запустить;

— запустить скриптом те службы, для которых стоит автозапуск, но в настоящее время по каким-то причинам они не работают.

Для всех этих ситуаций используются классы Win32_BaseService. Во многих ситуациях их применение позволит администратору сэкономить множество времени.

Ниже перечислены некоторые другие классы для работы с операционной системой:

— Win32_Share — позволяет создавать, удалять и настраивать параметры общих каталогов на компьютере. Настройка разрешений производится при помощи класса Win32_LogicalShareSecuritySetting;

— Win32_StartupCommand — возможность посмотреть, какие команды выполняются при запуске Windows с информацией о том, из какой ветви реестра производится их запуск;

— Win32_Account, Win32_UserAccount, Win32_Group — возможность получить доступ к учетным записям — локальным — если обращение идет к обычному компьютеру, или доменным — если обращение идет к контроллеру домена. В отличие от ADSI, при помощи этих объектов нельзя вносить изменения в учетные записи — они предназначены только для чтения информации и мониторинга;

— Win32_BootConfiguration — возможность получить информацию о

параметрах загрузки Windows (все доступно только на чтение).

Последняя большая категория классов WMI — классы, предназначенные для работы с установленными продуктами. Большинство таких классов предназначены для использования в установочных скриптах разработчиками программных продуктов. Однако возможности класса Win32_Product могут оказать очень полезными и для обычных администраторов. При помощи этого класса можно получить коллекцию, представляющие все установленные продукты в Windows. Для выполнения с ними различных операций предусмотрен набор методов этого класса:

— Install — возможность установить приложение (потребуется указать полный путь к пакету MSI). Admin — провести административную установку;

— Advertise — вывести пакет в списке доступных для установки приложений в консоли Add/Remove Programs;

— Reinstall, Upgrade, Configure, Uninstall — то, что делают эти методы, понятно из названия.

Например для установки программы на удаленный компьютер необходимо выполнить скрипт следующего вида (рис. 2.48).

```
Set oSoftware = oSvc.Get(«Win32_Product»)
errReturn = oSoftware.Install(«\\FS1\Distrib\myApp.msi»,,True)
Wscript.Echo errReturn
```

Рисунок 2.48 Фрагмент листинга

Некоторые моменты, связанные с применением объекта Win32_Product:

— в Windows 2003 Server этот объект по умолчанию не установлен. Доустановить его можно, выбрав в Add/Remove Programs → Add/Remove Windows Components → Management and Monitoring Tools компонент WMI Installer Provider;

— работать этот компонент может только через Windows Installer. Соответственно, он сможет увидеть только программы, установленные при помощи Windows Installer (обычно при помощи пакетов MSI);

— при автоматизированной установке программного обеспечения WMI Installer Provider по умолчанию будет обращаться к сетевому каталогу с дистрибутивом от имени локальной системной учетной записи (которая аутентифицировать на других компьютерах не может). Поэтому нужно или открывать к сетевому каталогу с дистрибутивом гостевой доступ (от имени учетной записи Guest), или копировать предварительно файлы дистрибутива на локальный диск, или использовать специальный синтаксис для указания имени пользователи и пароля.

Глава 3 Процесс разработки информационной системы

Сначала были определены требования по поводу состава собираемой информации. Было решено собирать автоматически информацию о следующих основных устройствах компьютеров в локальной вычислительной сети: процессорах, ОЗУ, НЖМД и видеокартах.

Информация о мониторах должна вводиться вручную, поскольку интерфейс управления Windows WMI не позволяет получать автоматически эти сведения. Также должен быть лист с итоговой информацией о конфигурации компьютеров в локальной вычислительной сети.

На первом этапе решения данной задачи была создана рабочая книга в программе Microsoft Office Excel. В ней были созданы необходимые листы. Причём, названия компьютеров копируются с первого листа на все остальные, а данные сводной таблицы копируются с остальных листов. Поэтому заполнять названия компьютеров нужно только на первом листе, а сводную таблицу заполнять вручную не нужно.

Результат выполнения данного этапа показан на рис. 3.1—3.7. Пример печати сводной таблицы параметров компьютеров в локальной вычислительной сети показан на рис. 3.8.

Далее, путём нажатия сочетания клавиш Alt+F11 был произведён запуск среды программирования VBA.

В этой среде сначала был создан программный модуль, куда был помещён следующий фрагмент исходного кода на языке Visual Basic for Applications (рис. 3.9—3.10).

Данный код необходим для обеспечения возможности ожидания завершения выполнения программного процесса.

Далее, для каждого из листов, на которых программным способом должна быть собрана информация о конфигурации компьютеров локальной вычислительной сети, был введён следующий программный код, состоящий из повторяющихся и неповторяющихся фрагментов.

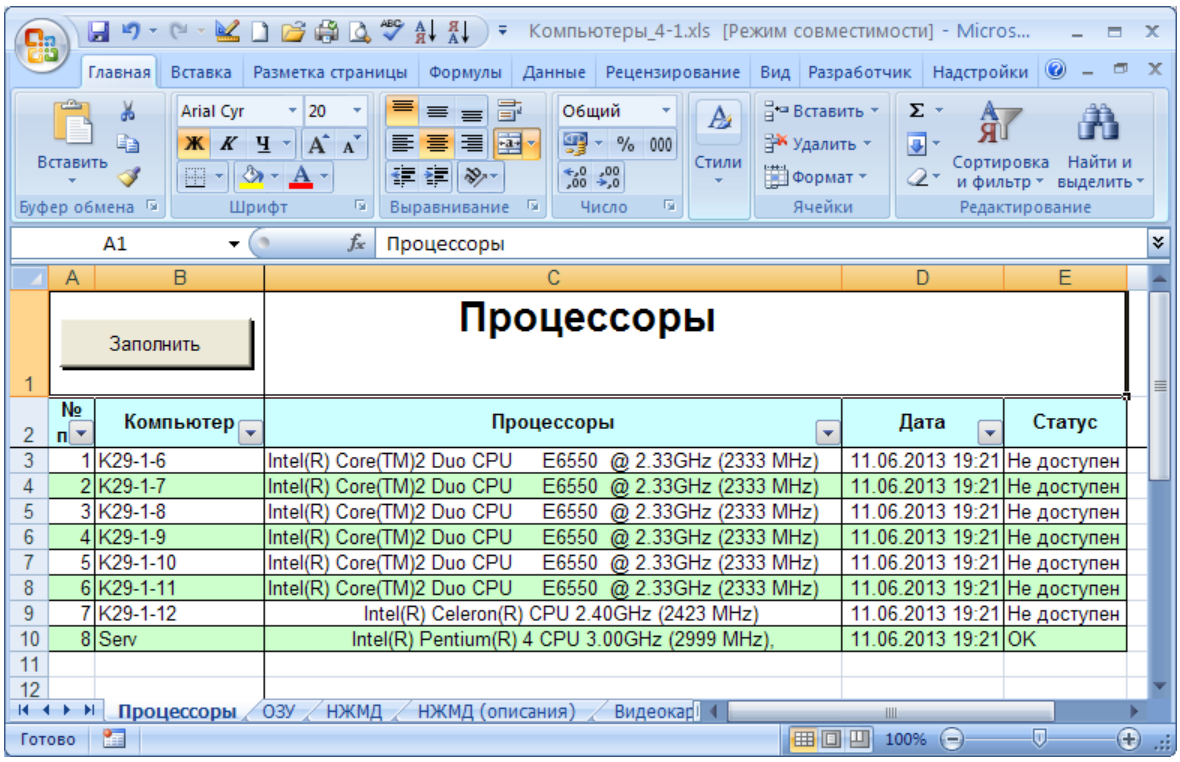

Рисунок 3.1 Лист «Процессоры»

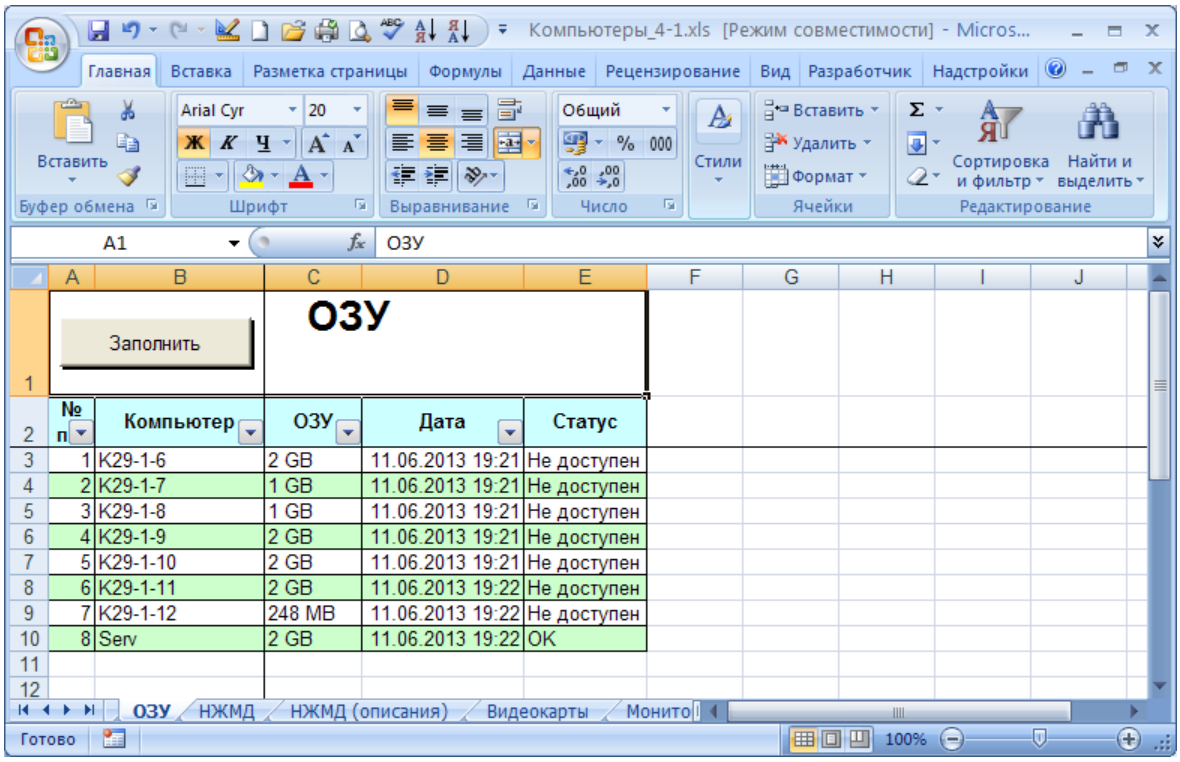

Рисунок 3.2Лист «ОЗУ»

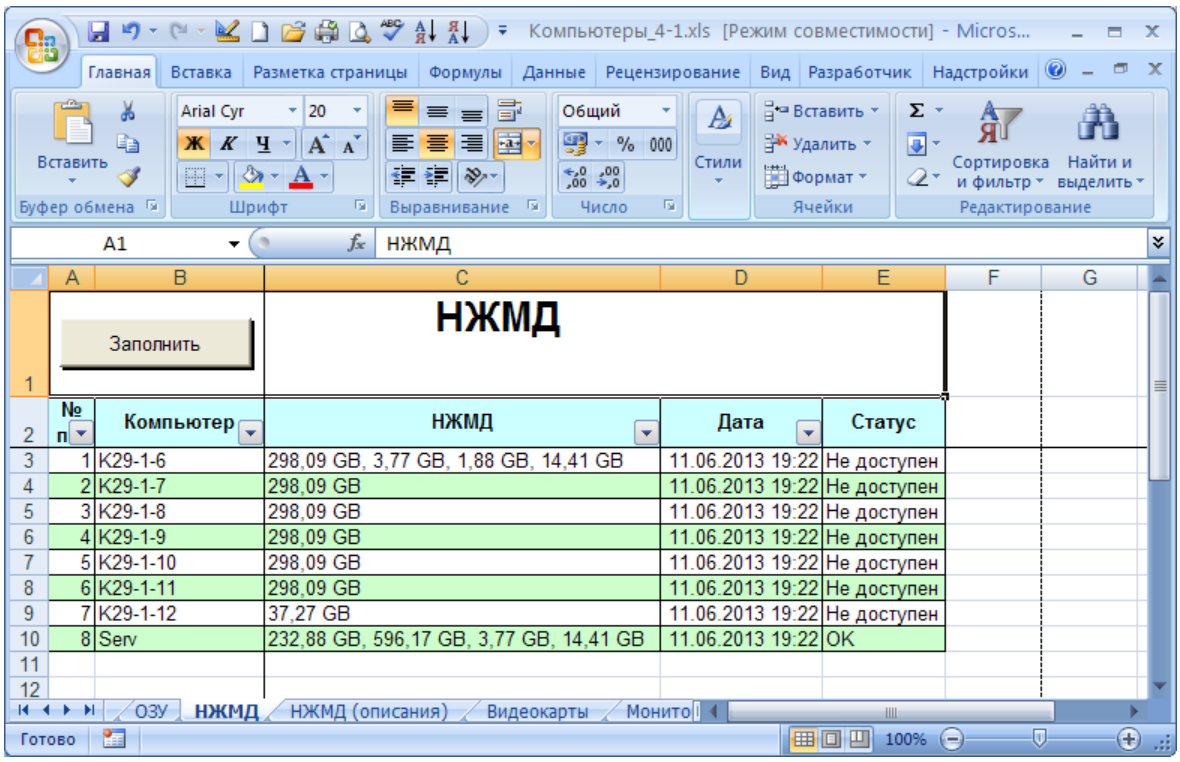

Рисунок 3.3 Лист «НЖМД»

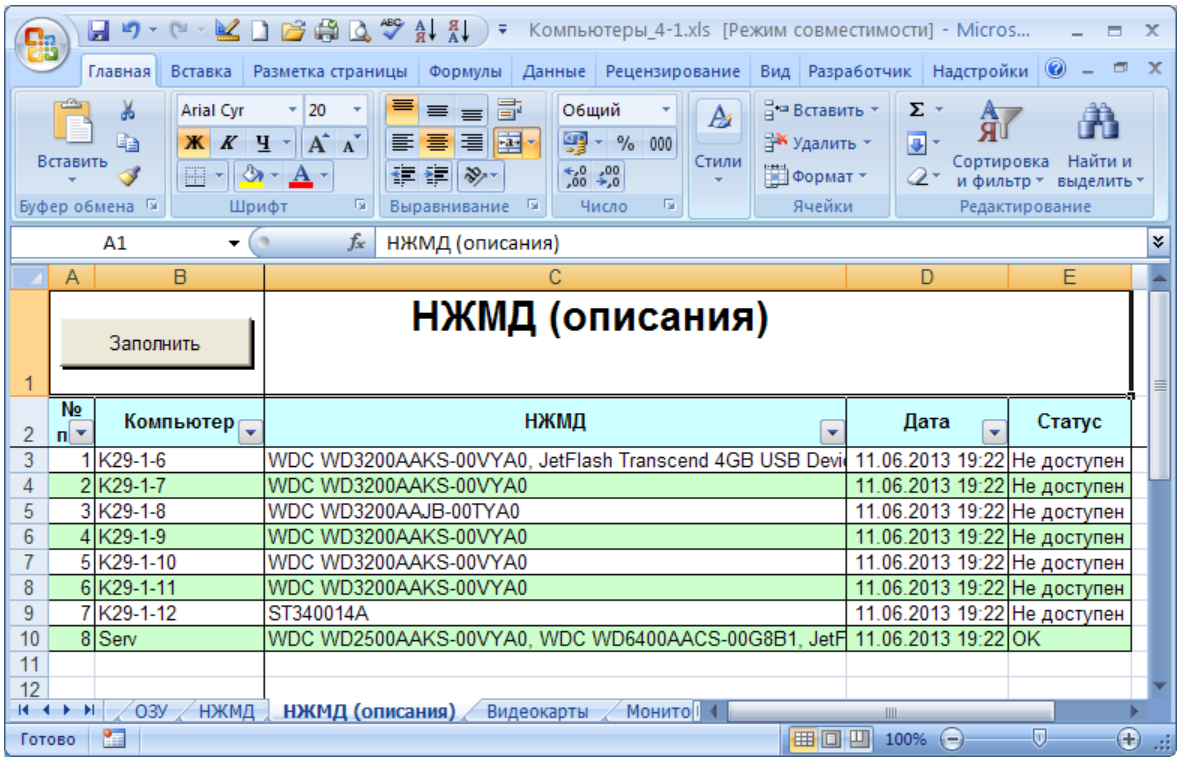

Рисунок 3.4 Лист «НЖМД (описания)»

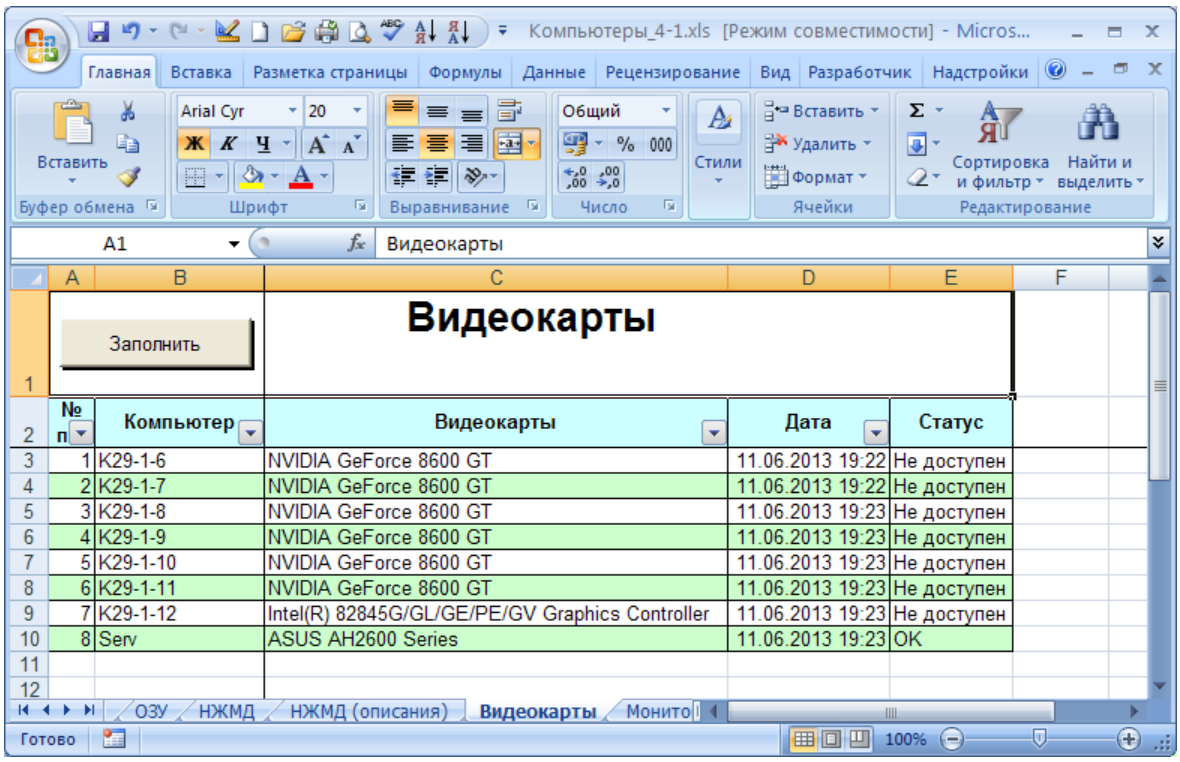

Рисунок 3.5 Лист «Видеокарты»

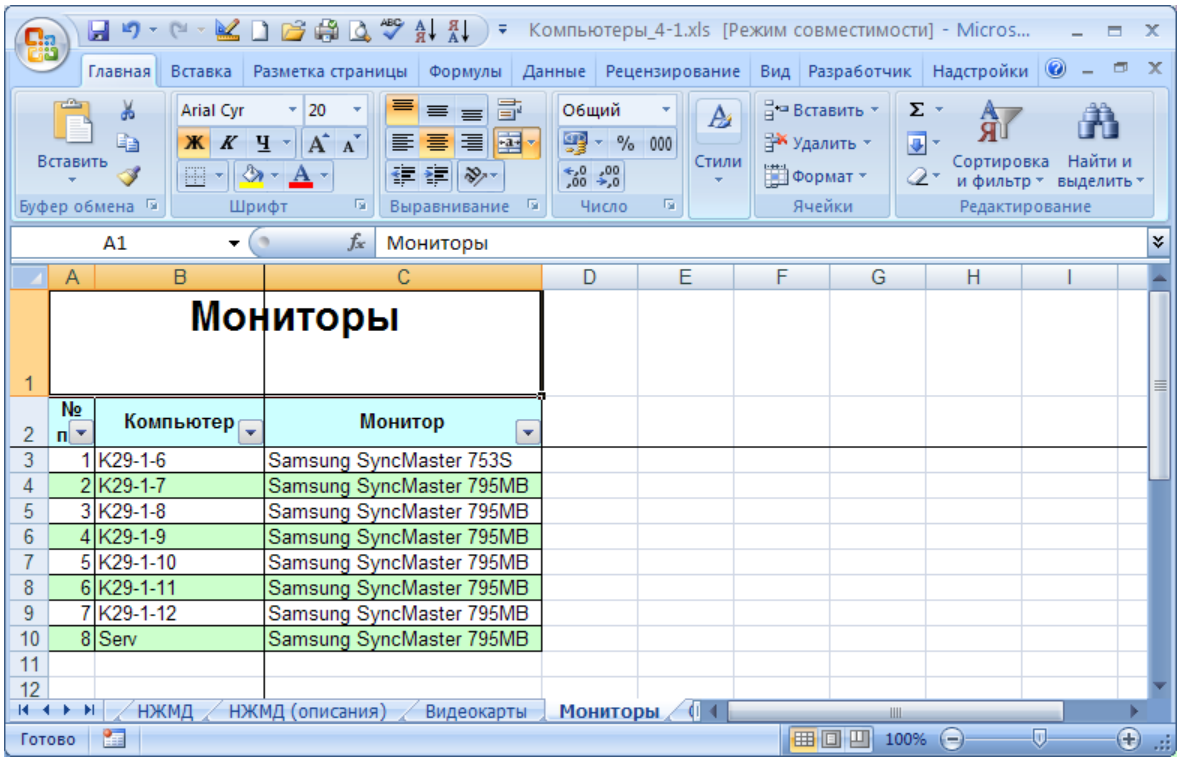

Рисунок 3.6 Лист «Мониторы»

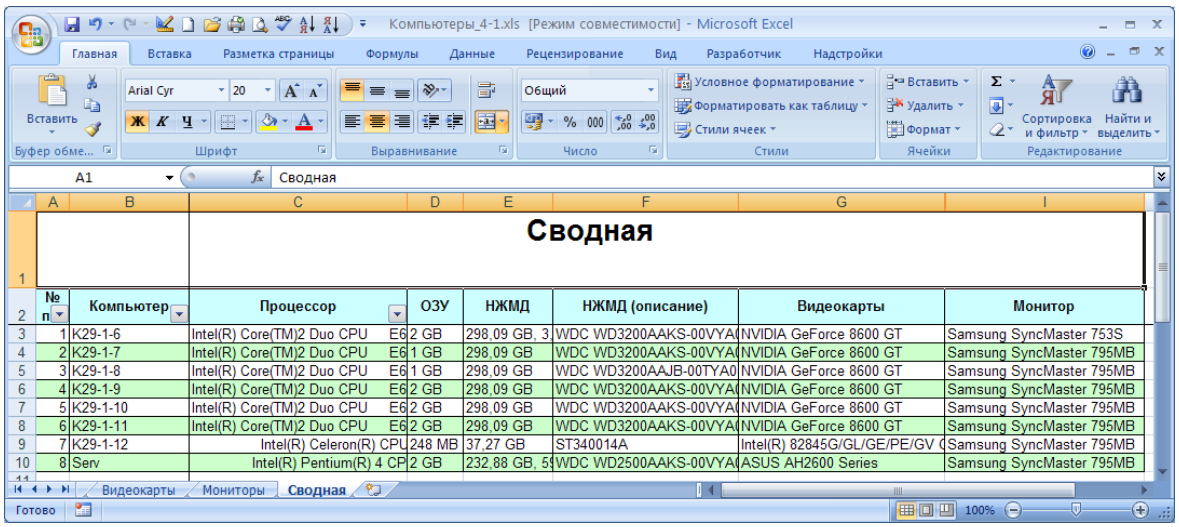

Рисунок 3.7 Лист «Сводная»

Сводная

№ п/п	Компьютер	Процессор	ОЗУ	НЖМД	НЖМД (описание)	Видеокарты	Монитор
1	K29-1-6	Intel(R) Core(TM)2 Duo CPU	E6	2 GB	298,09 GB, 3	WDC WD3200AAKS-00VYA0	NVIDIA GeForce 8600 GT	Samsung SyncMaster 753S
2	K29-1-7	Intel(R) Core(TM)2 Duo CPU	E6	1 GB	298,09 GB	WDC WD3200AAKS-00VYA0	NVIDIA GeForce 8600 GT	Samsung SyncMaster 795MB
3	K29-1-8	Intel(R) Core(TM)2 Duo CPU	E6	1 GB	298,09 GB	WDC WD3200AAJB-00TYA0	NVIDIA GeForce 8600 GT	Samsung SyncMaster 795MB
4	K29-1-9	Intel(R) Core(TM)2 Duo CPU	E6	2 GB	298,09 GB	WDC WD3200AAKS-00VYA0	NVIDIA GeForce 8600 GT	Samsung SyncMaster 795MB
5	K29-1-10	Intel(R) Core(TM)2 Duo CPU	E6	2 GB	298,09 GB	WDC WD3200AAKS-00VYA0	NVIDIA GeForce 8600 GT	Samsung SyncMaster 795MB
6	K29-1-11	Intel(R) Core(TM)2 Duo CPU	E6	2 GB	298,09 GB	WDC WD3200AAKS-00VYA0	NVIDIA GeForce 8600 GT	Samsung SyncMaster 795MB
7	K29-1-12	Intel(R) Celeron(R) CPU	248 MB	37,27 GB	ST340014A	Intel(R) 82845G/GL/GE/PE/GV G	Samsung SyncMaster 795MB
8	Serv	Intel(R) Pentium(R) 4 CPU	2 GB	232,88 GB, 59	WDC WD2500AAKS-00VYA0	ASUS AH2600 Series	Samsung SyncMaster 795MB

Рисунок 3.8 Пример печати сводной таблицы параметров компьютеров в локальной вычислительной сети

Повторяющая часть начинается со следующего фрагмента (рис. 3.11—3.14).

Далее следует код обработки полученных результатов, который меняется в зависимости от их вида и количества.

Для листа «Процессоры» он имеет следующий вид (рис. 3.15—3.16).

Для листа «ОЗУ» он имеет следующий вид (рис. 3.17—3.23).

Для листа «НЖМД» он имеет следующий вид (рис. 3.24—3.30).

Для листа «Видеокарты» он имеет следующий вид (рис. 3.31—3.32).

Общая оставшаяся часть имеет вид (рис. 3.33—3.34).

```
' подключение системных библиотек подпрограмм
Declare Function OpenProcess Lib «kernel32» (ByVal dwDesiredAccess As Long, ByVal
bInheritHandle As Long, ByVal dwProcessId As Long) As Long
Declare Function CloseHandle Lib «kernel32» (ByVal hObject As Long) As Long
Declare Function GetExitCodeProcess Lib «kernel32» (ByVal hProcess As Long, lpExitCode As
Long) As Long
Sub WaitFor(intProcID) ' начало подпрограммы ожидания завершения выполнения процесса с
кодом intProcID
 ' получить дескриптор процесса
 lngProcHandle = OpenProcess(&H400, 0, intProcID)
  If lngProcHandle <> 0 Then ' если дескриптор процесса не равен 0, то
 Do ' начало цикла с постусловием
 GetExitCodeProcess lngProcHandle, intProcID ' получить код завершения процесса
  DoEvents ' обработать системные события
```

Рисунок 3.9 Фрагмент листинга

```
  Loop While intProcID = &H103 ' до тех пор, пока код завершения процесса не станет равен
 &H103
   CloseHandle lngProcHandle ' закрыть дескриптор процесса
  End If ' конец условия lngProcHandle <> 0
 End Sub ' конец подпрограммы ожидания завершения выполнения процесса
```

Рисунок 3.10 Фрагмент листинга

```
' начало подпрограммы обработки события Click для объекта-кнопки cmdFill
Private Sub cmdFill_Click()
 ' в случае ошибки продолжать выполнение программы со строки, следующей после строки,
в которой возникла ошибка
 On Error Resume Next
  intRow = 3 ' номер строки
  intCompColumn = 2 ' номер колонки с названием компьютера
 intCPUColumn = 3 ' номер колонки с параметрами процессоров
 intDateColumn = 4 ' номер колонки с датой/временем сканирования
 intStateColumn = 5 ' номер колонки состояния результата сканирования
  ' получить объект «Активный лист» для текущей рабочей книги
 Set objSheet = ThisWorkbook.ActiveSheet
  ' начало цикла перебора строк колонки с названиями компьютеров до тех пор,
 ' пока ячейка, заданная номером строки и номером столбца не пустая
 Do While objSheet.Cells(intRow, intCompColumn) <> «»
```

Рисунок 3.11 Фрагмент листинга

```
  ' получить название компьютера из ячейки, заданной номером строки и номером столбца
  strComputer = objSheet.Cells(intRow, intCompColumn)
   ' сформировать строку с командой проверки доступности компьютера в локальной
вычислительной сети.
  ' Результат команды проверки доступности ping будет выведен в файл results.txt во
временной папке текущего пользователя.
  ' Путь ко временной папке текущего пользователя получить при помощи переменной
окружения TEMP
  strCmd = «cmd /c ping » & strComputer & « -n 1 > » & _
  Environ(«Temp») & «\result.txt»
   ' ожидать завершения выполнения команды проверки доступности компьютера
 ' в локальной вычислительной сети (ping)
  WaitFor Shell(strCmd, vbHide)
   intFile = FreeFile ' получить ранее не использованный номер для работы с файлом
```

Рисунок 3.12 Фрагмент листинга

```
 Open Environ(«Temp») & «\result.txt» For Input As intFile ' открыть файл results.txt для последовательного чтения
 For intCounter = 1 To 7 ' начало цикла «7 раз»
' если в файле, заданном номером, признак нахождения указателя считывания записей не находится в конце файла
 If Not EOF(intFile) Then
' то считать данные до конца строки (EOL -- два символа с кодами 13 -- CR и 10 -- LF) в переменную strResult
 Line Input #intFile, strResult
 End If ' конец условия Not EOF(intFile)
 Next intCounter ' конец цикла «7 раз»
```

Рисунок 3.13 Фрагмент листинга

```
Close intFile ' закрыть файл

If InStr(strResult, «TTL») > 0 Then ' если в переменной strResult содержится фрагмент «TTL» (компьютер «пингуется»), то
 strObject = «winmgmts:\\» & strComputer & «\root\cimv2» ' сформировать строку с параметрами получения информации о компьютере

Set objWMIService = GetObject(strObject)' получить объект информации о компьютере

If Err = 0 Then ' при отсутствии ошибки
```

Рисунок 3.14 Фрагмент листинга

```
' выполнить запрос WMI SQL к объекту информации о компьютере для получения
информации о процессорах
Set objItems = objWMIService.ExecQuery(«Select * From Win32_Processor», , 48)

' очистить переменную с параметрами устройства
strValue = «»

For Each objEntry In objItems ' начало цикла для каждого элемента информации о процессоре
If strValue = «» Then ' если значение переменной пусто, то
' получить и задать значению переменной название и тактовую частоту процессора
strValue = objEntry.Name & « (» & objEntry.CurrentClockSpeed & « MHz)»
Else ' иначе, если не пусто, то
' получить и добавить к значению переменной название и тактовую частоту процессора
strValue = strValue & «, » & objEntry.Name & « (« & objEntry.CurrentClockSpeed & « MHz)»
```

Рисунок 3.15 Фрагмент листинга

```
End If ' конец условия strValue = «»

DoEvents ' обработать системные события
Next objEntry ' конец цикла для каждого элемента информации о процессоре

' записать в ячейку, заданную номером строки и номером столбца, полученное значение
' параметров процессоров
objSheet.Cells(intRow, intCPUColumn) = strValue
```

Рисунок 3.16 Фрагмент листинга

```
' выполнить запрос WMI SQL к объекту информации о компьютере для получения
информации о логической конфигурации ОЗУ
Set objItems = objWMIService.ExecQuery(«Select * From
Win32_LogicalMemoryConfiguration», , 48)
```

Рисунок 3.17 Фрагмент листинга

```
For Each objEntry In objItems ' для каждого элемента информации о логической конфигурации ОЗУ
lngValue = objEntry.TotalPhysicalMemory ' получить общий объём физической памяти в килобайтах
Next ' конец цикла для каждого элемента информации о логической конфигурации ОЗУ

If Len(lngValue) Mod 3 = 0 Then ' если количество символов числа делится нацело на 3, то
intPower = Int(Len(lngValue) / 3) - 1 ' найти количество групп разрядов и уменьшить его на 1
Else
intPower = Int(Len(lngValue) / 3) ' найти количество групп разрядов
End If
```

Рисунок 3.18 Фрагмент листинга

```
' расчитать значение общего объёма физической памяти с учётом кратных долей килобайта и округлить до сотых долей
lngValue = FormatNumber(lngValue / 1024 ^ intPower)

' в зависимости от значения переменной intPower -- количества групп разрядов -- выбрать обозначение кратной доли
Select Case intPower
Case 0 ' если значение < 1000, то кратная доля будет
strUnit = « KB» ' килобайт
Case 1 ' если значение < 1000 ^ 2, то кратная доля будет
strUnit = « MB» ' мегабайт
Case 2 ' если значение < 1000 ^ 3, то кратная доля будет
strUnit = « GB» ' гигабайт
Case 3 ' если значение < 1000 ^ 4, то кратная доля будет
```

Рисунок 3.19 Фрагмент листинга

```
strUnit = « ТВ» ' терабайт
Case Else ' в любом другом случае кратная доля будет
strUnit = «» ' никакая
End Select

' записать в ячейку, заданную номером строки и номером столбца, полученное значение
' объёма физической памяти компьютера и обозначение кратной доли
objSheet.Cells(intRow, intRAMColumn) = lngValue & strUnit

' выполнить запрос WMI SQL к объекту информации о компьютере для получения
информации о физической конфигурации ОЗУ
Set objItems = objWMIService.ExecQuery(«Select * From Win32_PhysicalMemory», , 48)
```

Рисунок 3.20 Фрагмент листинга

```
For Each objEntry In objItems ' начало цикла для каждого элемента информации о физической
конфигурации ОЗУ
lngTempValue = objEntry.Capacity ' получить объём физической памяти элемента (модуля
памяти) в килобайтах

If Len(lngTempValue) Mod 3 = 0 Then ' если количество символов числа делится нацело на 3,
то
intPower = Int(Len(lngTempValue) / 3) - 1 ' найти количество групп разрядов и уменьшить его
на 1
Else
intPower = Int(Len(lngTempValue) / 3) ' найти количество групп разрядов
End If ' конец условия Len(lngTempValue) Mod 3 = 0
```

Рисунок 3.21 Фрагмент листинга

```
' расчитать значение объёма физической памяти элемента (модуля памяти)
' с учётом кратных долей килобайта и округлить до сотых долей
lngTempValue = FormatNumber(lngTempValue / 1024 ^ intPower)

' в зависимости от значения переменной intPower -- количества групп разрядов -- выбрать обозначение кратной доли
Select Case intPower
Case 1 ' если значение < 1000, то кратная доля будет
strUnit = « KB» ' килобайт
Case 2 ' если значение < 1000 ^ 2, то кратная доля будет
strUnit = « MB» ' мегабайт
Case 3 ' если значение < 1000 ^ 3, то кратная доля будет
strUnit = « GB» ' гигабайт
```

Рисунок 3.22 Фрагмент листинга

```
Case 4 ' если значение < 1000 ^ 4, то кратная доля будет
strUnit = « TB» ' терабайт
Case Else ' в любом другом случае кратная доля будет
strUnit = «» ' никакая
End Select ' конец выбора вариантов значения переменной intPower

DoEvents ' обработать системные события
Next objEntry ' конец цикла для каждого элемента информации о физической конфигурации ОЗУ
```

Рисунок 3.23 Фрагмент листинга

```
' выполнить запрос WMI SQL к объекту информации о компьютере для получения информации о НЖМД
Set objItems = objWMIService.ExecQuery(«Select * From Win32_DiskDrive», , 48)

' очистить переменную с параметрами устройства
strValue = «»

For Each objEntry In objItems ' начало цикла для каждого элемента информации о НЖМД
lngValue = objEntry.Size ' получить объём памяти НЖМД в байтах

If Len(lngValue) Mod 3 = 0 Then ' если количество символов числа делится нацело на 3, то
intPower = Int(Len(lngValue) / 3) - 1 ' найти количество групп разрядов и уменьшить его на 1
Else
intPower = Int(Len(lngValue) / 3) ' найти количество групп разрядов
```

Рисунок 3.24 Фрагмент листинга

```
End If ' конец условия Len(lngValue) Mod 3 = 0

' расчитать значение объёма памяти НЖМД
' с учётом кратных долей байта и округлить до сотых долей
lngValue = FormatNumber(lngValue / 1024 ^ intPower)

' в зависимости от значения переменной intPower -- количества групп разрядов -- выбрать обозначение кратной доли
```

Рисунок 3.25 Фрагмент листинга

```
Select Case intPower
Case 0 ' если значение < 1000, то кратная доля будет
strUnit = « bytes» ' байт
Case 1 ' если значение < 1000 ^ 2, то кратная доля будет
strUnit = « KB» ' килобайт
Case 2 ' если значение < 1000 ^ 3, то кратная доля будет
strUnit = « MB» ' мегабайт
Case 3 ' если значение < 1000 ^ 4, то кратная доля будет
strUnit = « GB» ' гигабайт
Case 4 ' если значение < 1000 ^ 5, то кратная доля будет
strUnit = « TB» ' терабайт
Case Else ' в любом другом случае кратная доля будет
strUnit = «» ' никакая
End Select ' конец выбора вариантов значения переменной intPower
```

Рисунок 3.26 Фрагмент листинга

```
If Left(objEntry.Caption, 1) = «\» Then ' если первый символ названия «\», то
' получить информацию о производителе НЖМД первым способом
strCaption = « (» & objEntry.Manufacturer & « » & objEntry.Description & «)»
Else ' иначе, если первый символ названия не «\», то
' получить информацию о производителе НЖМД вторым способом
strCaption = « (» & objEntry.Caption & «)»
End If ' конец условия Left(objEntry.Caption, 1) = «\»

' очистить переменную с информацией о производителе НЖМД
strCaption = «»

If strValue = «» Then ' если значение переменной пусто, то
' задать значению переменной объём НЖМД и название производителя
strValue = lngValue & strUnit & strCaption
```

Рисунок 3.27 Фрагмент листинга

```
Else
 ' добавить к значению переменной объём НЖМД и название производителя
 strValue = strValue & «, » & lngValue & strUnit & strCaption
 End If ' конец условия strValue = «»

 DoEvents ' обработать системные события
 Next objEntry ' конец цикла для каждого элемента информации о НЖМД

 ' записать в ячейку, заданную номером строки и номером столбца, полученное значение
 ' параметров НЖМД
 objSheet.Cells(intRow, intHDDColumn) = strValue
```

Рисунок 3.28 Фрагмент листинга

```
' выполнить запрос WMI SQL к объекту информации о компьютере для получения
информации о НЖМД
 Set objItems = objWMIService.ExecQuery(«Select * From Win32_DiskDrive», , 48)

 ' очистить переменную с параметрами устройства
 strValue = «»

 For Each objEntry In objItems ' начало цикла для каждого элемента информации о НЖМД
 If Left(objEntry.Caption, 1) = «\» Then ' если первый символ названия «\», то
 ' получить информацию о производителе НЖМД первым способом
 strCaption = objEntry.Manufacturer & « » & objEntry.Description
 Else ' иначе, если первый символ названия не «\», то
 ' получить информацию о производителе НЖМД вторым способом
 strCaption = objEntry.Caption
 End If ' конец условия Left(objEntry.Caption, 1) = «\»
```

Рисунок 3.29 Фрагмент листинга

```
If strValue = «» Then ' если значение переменной пусто, то
' получить и задать значению переменной название производителя НЖМД
strValue = strCaption
Else
' получить и добавить к значению переменной название производителя НЖМД
strValue = strValue & «, » & strCaption
End If ' конец условия strValue = «»

DoEvents ' обработать системные события
Next objEntry ' конец цикла для каждого элемента информации о НЖМД

' записать в ячейку, заданную номером строки и номером столбца, полученное значение
' параметров НЖМД
objSheet.Cells(intRow, intHDDColumn) = strValue
```

Рисунок 3.30 Фрагмент листинга

```
' выполнить запрос WMI SQL к объекту информации о компьютере для получения
информации о конфигурации видеокарт
Set objItems = objWMIService.ExecQuery(«Select * From Win32_DisplayConfiguration», , 48)

' очистить переменную с параметрами устройства
strValue = «»

For Each objEntry In objItems ' начало цикла для каждого элемента информации о
конфигурации видеокарт
If strValue = «» Then ' если значение переменной пусто, то
' получить и задать значению переменной название видеокарты
strValue = objEntry.Caption
Else
' получить и добавить к значению переменной название видеокарты
strValue = strValue & «, » & objEntry.Caption
```

Рисунок 3.31 Фрагмент листинга

```
Else
' получить и добавить к значению переменной название видеокарты
strValue = strValue & «, » & objEntry.Caption
End If ' конец условия strValue = «»

DoEvents ' обработать системные события
Next objEntry ' конец цикла для каждого элемента информации о конфигурации видеокарт

' записать в ячейку, заданную номером строки и номером столбца, полученное значение
' параметров видеокарт
objSheet.Cells(intRow, intVideoColumn) = strValue
```

Рисунок 3.32 Фрагмент листинга

```
' записать в ячейку, заданную номером строки и номером столбца, значение «OK»
objSheet.Cells(intRow, intStateColumn) = «OK»
Else ' иначе, при наличии ошибки
' записать в ячейку, заданную номером строки и номером столбца, значение «Не доступен»
objSheet.Cells(intRow, intStateColumn) = «Не доступен»

MsgBox Err.Description ' вывести сообщение об ошибке

' очистить информацию об ошибке
Err.Clear
End If ' конец условия Err = 0

DoEvents ' обработать системные события
Else ' иначе, если компьютер не пингуется, то
' записать в ячейку, заданную номером строки и номером столбца, значение «Не доступен»
```

Рисунок 3.33 Фрагмент листинга

```
 objSheet.Cells(intRow, intStateColumn) = «Не доступен»
 End If ' конец условия InStr(strResult, «TTL») > 0

 ' записать в ячейку, заданную номером строки и номером столбца, значение текущей даты и
времени
 objSheet.Cells(intRow, intDateColumn) = Now

 ' увеличить номер строки на 1
 intRow = intRow + 1
 Loop ' конец цикла While objSheet.Cells(intRow, intCompColumn) <> «»

 ' вывести сообщение «Готово»
 MsgBox «Готово»
End Sub ' конец подпрограммы обработки события Click для объекта-кнопки cmdFill
```

Рисунок 3.34 Фрагмент листинга

Заключение

На основании изложенного материала и проведённого анализа можно сделать следующие выводы.

В работе изложены теоретические и практические аспекты создания автоматизированных систем обработки данных для АО «Туапсинское АТП». Разработана информационная система и электронная таблица автоматизированной системы в среде системы управления электронными таблицами Microsoft Office Excel. Данная система управления электронными таблицами была выбрана потому, что пакет прикладного программного обеспечения Office фирмы Microsoft используется для автоматизации различных видов офисной деятельности на большинстве фирм, предприятий и организаций не только в нашей стране, но и во всём мире. А программа Excel, входящая в комплект поставки данного пакета прикладных программ, позволяет разрабатывать приложения для автоматизированной обработки электронных таблиц, которые можно применять для автоматизации различных сфер деятельности, связанных с автоматизированной обработкой табличной информации.

В работе изложена методология разработки автоматизированной информационной системы АО «Туапсинское АТП»; описана среда проектирования электронных таблиц и других объектов, связанных с ними, — Microsoft Excel; а также встроенная в приложения Microsoft Office среда программирования VBA, применяемая для создания программ, связанных с автоматизированной обработкой данных. Разработана информационная модель предметной области, произведено проектирование электронной книги для этой предметной области для АО «Туапсинское АТП» и реализовано в виде связанных между собой таблиц в программе Microsoft Excel, а затем разработаны дополнительные объекты для работы с этой электронной книгой.

Помимо этого разработаны программные модули, которые позволяют автоматизировать процесс занесения информации в созданную электронную

книгу в АО «Туапсинское АТП».

В настоящее время существуют различные варианты реализации подобных автоматизированных информационных систем, но данная разработка выгодно отличается от них тем, что имеет низкую стоимость и возможность произвольной адаптации под конкретные нужды АО «Туапсинское АТП».

Данная разработка может применяться в различных сферах практической деятельности, связанных с предметной областью данной автоматизированной информационной системы, а именно с учётом и контролем аппаратной конфигурации компьютеров организации.

Также данная разработка позволяет эффективно решить проблему учёта и контроля аппаратной конфигурации компьютеров АО «Туапсинское АТП».

Список источников литературы

1. Аляев Ю., Козлов О. Алгоритмизация и языки программирования Pascal, C++, Visual Basic. — М.: Финансы и статистика, 2010.
2. Аляев Ю., Козлов О. Алгоритмизация и моделирование информационных систем. — М.: Финансы и статистика, 2009.
3. Биллиг В.А., Дехтяръ М.И.VBA и Office 2007. Офисное программирование. М.: Издательский отдел «Русская Редакция», ТОО «Cheannel Trading Ltd.», 2008.
4. Биллиг В.А.VBA в Office 2007. Офисное программирование. М.: Издательско-торговый дом «Русская Редакция», 2009.
5. Боуман Д, Эмерсон С., Дарновски М. Практическое руководство по SQL. — Киев: Диалектика, 2009.
6. Гарнаев А.Ю.Самоучитель VBA. СПб.: БХВ-Петербург, 2011.
7. Гилуа М.М. Множественная модель данных в информационных системах. — М.: Наука, 2009.
8. Голицына О. Л., Попов И. И. Основы алгоритмизации и программирования: Учебное пособие. — М.: Форум: Инфра-М, 2008.
9. Грабер М. Введение в SQL. — М.: Лори, 2010. — 379 с.
10. Грабер М. Справочное руководство по SQL. — М.: Лори, 2009. — 291 с.
11. Грейди Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ / Пер. с англ. — 2-е изд. — М.: Бином, 2008.
12. Дейт К. Введение в системы баз данных // 6-издание. — Киев: Диалектика, 2008. — 784 с.
13. Джеймс Фокселл. Освой самостоятельно Visual Basic.NET за 24 часа. — М.: Вильямс, 2009.
14. Джексон Г. Проектирование реляционных баз данных для использования с микроЭВМ. — М.: Мир, 2010. — 252 с.
15. Диго С.М. Проектирование и использование баз данных. — М.:

Финансы и статистика, 2009. — 208 с.

16. Дэн Кларк. Объектно-ориентированное программирование в Visual Basic .NET. — СПб.: Питер, 2009.

17. Зак Д. Самоучитель Visual Basic.Net. — Киев: BHV; СПб.: Питер, 2011.

18. Злуф М.М. Query-by-Example: язык баз данных //СУБД. — 2008. — №3. — С. 149—160.

19. Информатика: Практикум по технологии работы на компьютере / Под ред. Н. В. Макаровой. М.: Финансы и статистика, 2009.

20. Камминг С. VBA для «чайников». 3-е изд. М.: Издательский дом «Диалектика», 2009.

21. Кетков Ю., Кетков А. Практика программирования: Visual Basic, C++ Builder. — СПб.: BHV, 2010.

22. Кириллов В.В. Структуризованный язык запросов (SQL). — СПб.: ИТМО, 2009. — 80 с.

23. Кузнецов С.Д. Введение в системы управления базами данных // СУБД. — 2011. — №1, 2, 3, 4, 2009. — №1, 2, 3, 4, 5.

24. Кузнецов С.Д. Стандарты языка реляционных баз данных SQL: краткий обзор // СУБД. — 2006. — № 2. — С. 6—36.

25. Кузьменко В. Г. VBA 2007. М.: ЗАО «Издательство БИНОМ», 2010.

26. Пономарев В. Visual Basic.NET: — Экспресс-курс. — СПб.: BHV — Санкт-Петербург, 2009.

27. Санна П. и др.Visual Basic® для приложений (версия 5) в подлиннике / Пер. с англ. СПб.: BHV – Санкт-Петербург, 2009.

28. Семакин И. Г., Шестаков А. П. Основы программирования: Учебник. — М.: Мастерство, 2010.

29. Харитонова И., Вольман Н. Программирование в Access 2007: Учебный курс. СПб.: Питер, 2008.