Лабораторная работа №3
Пакетные файлы и сценарии в ОС семейства Windows

Цель работы: Изучить принципы построения и организации пакетных файлов и сценариев в среде ОС семейства Windows.

3.1. Краткие теоретические сведения

Пакетный файл — это неформатированный текстовый файл ASCII, содержащий одну или несколько команд ОС. Имена пакетных файлов имеют расширения .cmd или .bat. ОС при работе с пакетным файлом последовательно обрабатывает его команды после ввода его имени в строке командной оболочки или запуска из другой программы.

Другой разновидностью пакетного файла является сценарий, представляющий собой программу, состоящую из набора инструкций для работы приложения или служебной утилиты. Инструкции в сценариях обычно выражаются с использованием правил и синтаксиса соответствующего приложения или служебной утилиты в сочетании с простыми управляющими операторами, такими как операторы циклов и условные операторы.

Пакетные файлы и сценарии часто называют командными файлами, содержащими любые команды. Некоторые команды, такие как **For**, **Goto** и **If**, позволяют выполнять обработку условий в пакетных файлах. В частности, **If** позволяет запускать команды в зависимости от выполнения заданного условия. Другие команды позволяют управлять вводом и выводом, а также запускать другие пакетные файлы. Совместно с командами, изученными в предыдущих лабораторных работах, вышеуказанные команды позволяют создавать пакетные файлы практически для любых целей управления работой и администрирования ОС Windows.

Следующее, что необходимо отметить при организации пакетных файлов и сценариев, является применение переменных, задающих поведение командной оболочки или ОС в целом и так называемых пакетных параметров командного интерпретатора, которые используются в пакетном файле для получения информации о настройках среды.

Имеется возможность определить поведение среды командной оболочки или всей ОС с помощью двух типов переменных среды: *системных и локальных*. Системные переменные определяют поведение глобальной среды ОС. Локальные переменные определяют поведение среды в конкретном экземпляре командного интерпретатора **Cmd.exe**.

Системные переменные среды задаются заранее в ОС Windows и доступны для всех ее процессов. Только пользователи с привилегиями администратора могут изменять эти переменные.

Локальные переменные среды доступны в случае, когда пользователь, для которого они были созданы, входит в систему. В частности, локальные переменные реестра **HKEY_CURRENT_USER** подходят только для текущего пользователя, но определяют поведение глобальной среды ОС.

В следующем списке представлены различные типы переменных в порядке убывания их приоритета:

- встроенные системные переменные,
- системные переменные реестра **HKEY_LOCAL_MACHINE**,
- локальные переменные реестра **HKEY_CURRENT_USER**,
- все переменные среды и пути, указанные файле Autoexec.bat,
- все переменные среды и пути, указанные в сценарии входа в систему, если он имеется,

- переменные, используемые интерактивно в пакетном файле или сценарии.

Чтобы иметь возможность подставить значение в переменную среды из командной строки или в пакетном файле (сценарии), следует заключить имя соответствующей переменной (**Приложение 4**) в символы процентов (%), например, **Set** MyPath=%CD%. Символы процентов указывают на то, что командный интерпретатор должен обратиться к значению переменной без посимвольного ее разложения и сравнения.

Командный интерпретатор **Cmd.exe** может оперировать переменными с %0 по %9. При использовании пакетных параметров переменная %0 заменяется именем пакетного файла, а переменные с %1 по %9 — на соответствующие аргументы командной строки. Для доступа к переменным больше %9 необходимо воспользоваться командой **Shift**. Параметр %* ссылается на все аргументы, которые передаются пакетному файлу, за исключением %0.

В качестве примера, рассмотрим копирование содержимого из каталога 1 (Folder1) в каталог 2 (Folder2), где параметр %1 заменяется значением Folder1, а параметр %2 соответственно значением Folder2. В пакетном файле **Mybatch.bat** следует ввести следующую строку:

Xcopy %1*.* %2

Используйте пакетный файл Mybatch.bat следующим образом:

Mybatch.bat C:\folder1 D:\folder2

Результат будет таким же, как и при записи в пакетный файл строки:

Xcopy C:\folder1*.* D:\folder2\

С пакетными параметрами можно также использовать модификаторы. Модификаторы используют информацию о текущем диске и каталоге как часть или полное имя файла (каталога).

Синтаксис модификатора: **%~xy,** где **x** — символьное сокращение действия, определяемое модификатором, **y** — идентификатор переменной (в диапазоне от 1 до 9).

В табл. 3.1 и 3.2 описаны модификаторы, выполняемые ими действия, и даны возможные комбинации модификаторов и квалификаторов для получения более сложных результатов. В этих таблицах %1 и переменную среды PATH можно заменить другими значениями пакетных параметров.

Таблица 3.1. Модификаторы и выполняемые ими действия

№ п/п.	Модификатор	Описание
1.	%~1	расширение %1 и удаление любых кавычек (" ")
2.	%~f1	замена %1 полным путем
3.	%~d1	замена %1 именем диска
4.	%~p1	замена %1 путем
5.	%~n1	замена %1 именем файла
6.	%~x1	замена %1 расширением имени файла

7.	%~s1	замена путем, содержащим только короткие имена
8.	%~a1	Замена %1 атрибутами файла
9.	%~t1	замена %1 датой и временем модификации файла
10.	%~z1	замена %1 размером файла
11.	%~$PATH:1	поиск в каталогах, перечисленных среди переменных среды PATH, замена %1 полным именем первого найденного файла. Если переменная среды не определена или поиск не обнаружил файлов, модификатор выдает пустую строку.

Таблица 3.2. Комбинации модификаторов и квалификаторов

№ п/п.	Модификатор	Описание
1.	%~dp1	замена %1 именем диска и путем
2.	%~nx1	замена %1 именем файла и расширением
3.	%~dp$PATH:1	поиск в каталогах, перечисленных в переменной среды PATH, и замена %1 именем диска и путем к первому найденному файлу.
4.	%~ftza1	замена %1 строкой, аналогичной результату работы команды **Dir**

Еще один модификатор, являющийся уникальным, имеет вид %*. Он представляет все аргументы, переданные пакетному файлу. Этот модификатор не используется в комбинации с модификатором %~.

Подводя промежуточные итоги по теоретическому материалу, необходимо напомнить еще о двух возможностях, а именно о конвейерах команд и «каналах», рассмотренных в предыдущих лабораторных работах, информация о которых доступна в **Приложениях 1** и **2** настоящего лабораторного практикума. Наряду с рассмотренными командами и утилитами, модификаторами и квалификаторами, они являются инструментами для расширения функционала пакетных файлов и сценариев при их построении и организации.

Исполняющим механизмом, позволяющим реализовать задуманные в пакетном файле или сценарии действия, является сервер сценариев ОС Windows, который позволяет быстро запустить пакетный файл или сценарий, введя его имя строке командной оболочки. Сервер сценариев служит контроллером средств обработки сценариев ОС. Однако, в отличие от других средств обработки сценариев, сервер сценариев ОС Windows не требует много памяти и является идеальным средством, как для интерактивных, так и для пакетных сценариев, таких как сценарий входа в систему или сценарий администрирования.

Существуют две версии сервера сценариев, доступных в окне командной оболочки: ***Wscript.exe*** — позволяет задавать параметры выполнения сценариев в окне свойств, и ***Cscript.exe*** — позволяет задавать параметры выполнения сценариев с помощью ключей командной строки.

В ранних версиях ОС Windows в качестве языка сценариев поддерживался только язык команд MS-DOS. По сравнению с новыми языками VBScript и JScript, язык команд MS-DOS обладает ограниченным набором средств, хотя и является более компактным и быстрым. В частности, в MS-DOS нет средств для управления процессом выполнения программы, в то время как сервер сценариев ОС Windows, основанный на мощном языке VBScript (JScript), позволяет воспользоваться подобными преимуществами и при этом поддержка языка команд MS-DOS по-прежнему остается доступной.

Для разработки сценариев ОС Windows следует использовать редакторы сценариев JScript или VBScript (в составе Visual Basic Scripting Edition). При запуске сценария из командной строки, сервер сценария читает и передает содержимое указанного файла зарегистрированному обработчику сценариев. Для определения языка сценария используется расширение имени файла (.vbs для VBScript, .js для JScript). Благодаря этому, разработчик сценария не обязан знать точные программные идентификаторы (ProgID) различных обработчиков сценариев. Сопоставление расширения имени файла сценария с программным идентификатором и запуск конкретного обработчика сценариев осуществляется непосредственно сервером сценариев ОС Windows.

В рамках настоящей лабораторной работы не предполагается использование среды Visual Basic для написания сценариев, поскольку, с одной стороны, эта среда изучается в рамках отдельного курса, а с другой стороны, зная общие принципы построения и организации пакетных файлов (сценариев) и имея достаточно обширную базу примеров, доступных на сайте Microsoft, без труда можно исследовать их работу и, в случае необходимости, внести в необходимый скрипт изменения, отражающие специфику поставленной задачи.

Простейшим сценарием, не требующим применения среды Visual Basic, является сценарий входа в систему, представляющий собой файл, связываемый с одной или несколькими учетными записями пользователей. Обычно сценарий входа является пакетным файлом, который автоматически выполняется при каждом входе пользователя в систему. Сценарии входа используются для настройки рабочей среды пользователя при входе и позволяют администратору задавать основные параметры рабочей среды пользователя без непосредственного его участия.

3.2. Подготовка к выполнению лабораторной работы

Поскольку пакетные файлы могут включать в себя любые команды, их конвейеры и «каналы», при большом количестве условий и циклов последствия некорректной работы пакетного файла могут быть непредсказуемыми для ОС, и возможно как следствие разрушительными. Поэтому для организации пакетного файла разработчику необходимо четко представлять себе, что именно и каким образом должно происходить в системе при работе этого файла, какая последовательность действий реализуется в результате выполнения задуманного сценария и как на эти действия реагирует ОС.

Помимо рассмотренных в предыдущих лабораторных работах команд, которые могут быть использованы при организации пакетного файла, существует ряд дополнительных, функционал которых напоминает операторы языков программирования высокого уровня. К их числу относятся: **At, Call, Doskey, Echo, Endlocal, For, Goto, If, Pause, Rem, Set, Setlocal** и **Shift.**

В настоящей лабораторной работе предполагается ознакомление с основными командами, используемыми в качестве инструментов организации пакетных файлов, создание командного файла в формате ASCII, реализующего определенный сценарий работы системы, а также оценка возможности использования его в качестве сценария входа в систему.

Перед началом выполнения лабораторной работы в среде ОС Windows необходимо выполнить следующее:

- загрузить ОС Windows и активировать справочное меню **(Пуск | Справка и поддержка)**;
- ознакомиться с описанием и синтаксисом ввода командного интерпретатора **Cmd.exe**;

- ознакомиться с описанием и синтаксисом ввода приведенных команд и служебных утилит.

3.3. Порядок выполнения лабораторной работы

Лабораторная работа выполняется последовательно в соответствии с определенным порядком и включает в себя два учебных задания.

3.3.1. Учебное задание №1. Изучение основных команд, предназначенных для организации пакетного файла в ОС Windows.

Примечание. Для выполнения данного учебного задания при себе необходимо иметь USB – Flash - накопитель, отформатированный в среде ОС Windows.

Порядок выполнения:

I. Загрузить командную оболочку:
- нажмите **Пуск | Выполнить**,
- наберите в появившемся окне **Cmd.exe** (или **cmd**),
- нажмите **Enter** для ввода.

II. Одной из первых команд, имеющей первостепенное значение и предназначенной для отображения и установки переменных среды в ОС Windows, является команда **Set**.

Синтаксис команды **Set**:

Set [*переменная*=[*строка*]],

где параметр:

переменная — задает имя переменной, значение которой требуется присвоить или изменить.

строка — задает строковое значение для указанной переменной.

/p — позволяет установить значение переменной среды для входной строки.

/a *выражение* — указывает, что строка справа от знака равенства является числовым выражением, значение которого вычисляется. При этом обработчик выражений поддерживает операции, список которых приведен в табл. 3.3.

В пакетных файлах при использовании любых логических или двоичных операторов необходимо заключить строку *выражения* в кавычки. Любые нечисловые строки в *выражении* рассматриваются как имена переменных среды, значения которых преобразуются в числовой вид перед их использованием. Если переменная с указанным именем не определена в системе, вместо нее подставляется нулевое значение, что позволяет выполнять арифметические операции со значениями переменных среды.

Если команда с ключом **/a** вызывается из командной строки, а не из пакетного файла, она выводит окончательное значение выражения.

Числовые значения рассматриваются как десятичные, если перед ними не стоит префикс 0x для шестнадцатеричных чисел, и 0 для восьмеричных чисел. Например, числа $0x12_h$, и 022_8 обозначают десятичное число 18. Обратите внимание на запись восьмеричных чисел: 08 и 09 не являются допустимыми числами, так как в восьмеричной системе исчисления цифры 8 и 9 не используются.

Таблица 3.3. Операции обработчика выражений команды **Set**

Операция	Описание
( )	Группировка
! ~ -	унарные операторы
* / % + -	арифметические операторы
<< >>	двоичный сдвиг
&	двоичное И
^	двоичное исключающее ИЛИ
|	двоичное ИЛИ
= *= /= %= += -= &= ^= |= <<= >>=	Присвоение
,	разделитель операторов

При включенной расширенной обработке команд доступны несколько переменных среды, которые не отображаются в списке при стандартном вызове с помощью команды **Set**. Значения этих переменных вычисляются динамически каждый раз при их вызове. Если подобная переменная среды задается явным образом, то ее значение перекрывает соответствующее динамическое значение, описанное ниже (табл. 3.4).

Таблица 3.4. Динамические значения команды **Set**

Значение	Описание действия
%Cd%	раскрывается в строку текущей директории
%Date%	раскрывается в текущую дату
%Time%	раскрывается в текущее время
%Random%	раскрывается в случайное десятичное число в диапазоне от 0 до 32767
%Errorlevel%	раскрывается в текущее значение ErrorLevel
%Cmdextversion%	раскрывается в текущее значение версии расширенной обработки команд
%Cmdcmdline%	раскрывается в исходную командную строку, которая вызвала текущее окно командной оболочки

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС **(Пуск | Справка и поддержка)** в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Set /?** (или просто **Set**) и нажав **Enter** для ввода.

Задача №3.3.1а. Исследовать способы применения команды присвоения переменной среды **Set** на конкретных примерах.

1. Отобразите переменные среды двумя способами: из командной оболочки и окна свойств системы (**Пуск | Панель управления | Система)**.
2. Задайте переменную среды, содержащую определенный путь к месту назначения, выбранный самостоятельно.

3. Проверьте наличие в системе переменной среды, заданной в предыдущем пункте задания.

4. Выведите значение выражения, определенного в соответствии с вариантом задания (см. подраздел 3.5 настоящей лабораторной работы), в качестве переменной среды **Result**.

5. Задайте переменную среды с различными вариантами динамически формируемых значений (табл. 3.4). Варианты динамических значений выберете самостоятельно.

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.5.

Таблица 3.5. Результаты выполнения команды **Set**

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		
4.		
5.		

III. Следующая команда **Rem**, наиболее часто встречающаяся в пакетных файлах, добавляет в них комментарии. При обработке пакетной программы строки, начинающиеся с **Rem**, пропускаются.

Синтаксис команды **Rem**:

Rem [*текст*],

где параметр:

текст — задает строку символов, используемую в качестве комментария.

Команда **Rem** не выводит комментарии на экран. Для их вывода в пакетных файлах необходимо использовать команду **Echo**.

IV. Вывод на экран сообщения или задание режима вывода на экран сообщений команд осуществляется с помощью команды **Echo**.

Синтаксис команды **Echo**:

Echo [{on | off}] [*сообщение*],

где параметр:

{on | off} — включает или отключает режим отображения на экране информации о работе команд.

сообщение — задает текст для вывода на экран.

Для вывода сообщений из нескольких строк без вывода дополнительных команд между ними следует использовать несколько последовательных команд **Echo** после команды **Echo off**:

```
Cls
@Echo off
Echo.
Rem  ****** Эта пакетная программа ******
Rem  ***** иллюстрирует возможности *****
Rem  ********** команды Echo **********
Echo.
Echo ******** This batch program ********
Echo *** illustrates possibilities of ***
Echo ********* the Echo command *********
Echo.
Pause
```

Дополнительная информация по данным командам, а также примеры их использования доступны в справке ОС **(Пуск | Справка и поддержка)** в соответствующих разделах. Справку также можно получить, набрав в окне командной оболочки строку **Rem /?** (**Echo /?**) и нажав **Enter** для ввода.

Задача №3.3.1б. Исследовать способы применения команды отображения текста **Echo** на конкретных примерах.

1. Создайте пакетный файл, воспользовавшись любым текстовым редактором. Имя пакетного файла выберете самостоятельно.
2. Введите в созданный пакетный файл текст, приведенного выше примера.
3. Сохраните текст пакетного файла.

При выполнении задания используйте следующие инструкции:

- воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,
- изучите пример и полученный с его помощью результат, обратив внимание на то, что команда **Echo** с точкой (.) в конце выводит на экран пустую строку, а символ «коммерческое И» (@) перед командой **Echo** отключает режим отображения команд.
- сделайте соответствующий вывод и запишите его в отчет.

Контрольный вопрос:

Что произойдет при обработке пакетного файла командным интерпретатором, если в пакетном файле будут встречаться пустые строки, не закомментированные с помощью команды **Rem**?

V. Утилита **For** командной среды по своему смыслу аналогична одноименной команде известных языков высокого уровня и предназначена для выполнения циклических операций для заданного множества операндов.

Синтаксис команды **For**:

For {% | %%} *переменная* **in (***множество***) do** *команда* [*ПараметрыКоманднойСтроки*]

где параметр:

{%переменная | %%переменная} — обязательный замещаемый параметр. %*Переменная* используется для выполнения команды из строки в окне командной оболочки, в то время как %%*переменная* используется для выполнения команды в пакетном файле. *Переменные* должны учитывать регистр и могут быть представлены в виде, например, %A, %B или %C. При этом можно использовать любые символы, кроме цифр 0–9, a, чтобы не было конфликта, эти цифры применяются с параметрами пакетных файлов, то есть %0–%9. Для простых пакетных файлов вполне достаточно обозначений с одним символом, например, %%f; в сложных пакетных файлах могут быть использованы также другие обозначения для параметра *переменная*.

(множество) — обязательный параметр, задающий один или группу файлов, каталогов, диапазон значений или текстовых строк, подлежащих обработке заданной *командой*. Для задания групп файлов можно использовать подстановочные знаки (* и ?). Например, допустимыми являются следующие варианты: (*.doc), (*.doc *.txt *.me), (jan*.doc jan*.rpt feb*.doc feb*.rpt), (ar??1991.* ap??1991.*). Скобки являются обязательными для обозначения *множества*.

команда — обязательный параметр, задающий команду, которая будет выполнена для каждого операнда *множества*.

ПараметрыКоманднойСтроки — задает параметры командной строки, которые используются с указанной *командой*.

Если расширения командного интерпретатора разрешены, что делается по умолчанию, то существуют дополнительные формы команды **For**.

For /D {% | %%} *переменная* **in (***множество***) do** *команда* [*ПараметрыКоманднойСтроки*]

Команда с ключом **/D** выполняется для каждого каталога, не принимая во внимание файлы в указанном каталоге из *множества*.

For /R [[*диск***:**]*путь*] {% | %%} *переменная* **in (***множество***) do** *команда* [*ПараметрыКоманднойСтроки*]

Рекурсивная форма команды с ключом **/R** организует проход по дереву каталогов с корнем в [[*диск***:**]*путь*], выполняя инструкцию **For** для каждого каталога в дереве. Если параметр *множество* задан одной точкой (.), то команда просто перечислит каталоги в дереве.

For /L {% | %%} *переменная* **in (***НачальноеЗначение#,шаг#,КонечноеЗначение#***) do** *команда* [*ПараметрыКоманднойСтроки*]

Форма с ключом **/L** осуществляет итерации по диапазону значений. Диапазон значений задается *Начальным* и *КонечнымЗначением#.* Кроме этого, задается *шаг#*, который может быть, как положительным, так и отрицательным. Например, (1,1,5) реализует следующую последовательность «1, 2, 3, 4, 5».

Следующая форма команды, задаваемая ключом **/F**, используется для обработки вывода команды, строк и содержимого файла методом «разбора», заключающегося в чтении

результатов вывода, строки или содержимого файла, разбиении его на отдельные строки текста и разборе каждой строки на маркеры.

For /F ["*КлючевыеСловаРазбора*"] {% | %%} *переменная* **in (***МножествоИменФайлов***) do** *команда* [*ПараметрыКоманднойСтроки*]

For /F ["*КлючевыеСловаРазбора*"] {% | %%} *переменная* **in (**"*СимвольнаяСтрока*"**) do** *команда* [*ПараметрыКоманднойСтроки*]

For /F ["*КлючевыеСловаРазбора*"] {% | %%} *переменная* **in (**'*команда*'**) do** *команда* [*ПараметрыКоманднойСтроки*]

Параметр *КлючевыеСловаРазбора* это возможные маркеры, представляющие собой любые несокращаемые текстовые элементы анализируемых данных (табл. 3.6). Маркеры разделяются пробелами и воспринимаются как переменные итерации. В частности, если используется параметр **usebackq**, синтаксис указанной выше формы модифицируется следующим образом:

…*переменная* **in (**"*МножествоИменФайлов*"**) do** *команда*…

…*переменная* **in (***'СимвольнаяСтрока'***) do** *команда*…

…*переменная* **in (**`*команда*`**) do** *команда*…

Аргумент *МножествоИменФайлов* задает одно или несколько имен файлов, каждый из которых открывается, считывается и обрабатывается до перехода к следующему файлу в аргументе.

Таблица 3.6. *КлючевыеСловаРазбора*

Ключевое слово	Описание
eol=c	Задает только один символ конца строки.
skip=n	Задает число строк, пропускаемых в начале файла.
delims=xxx	Задает набор разделителей. Заменяет набор разделителей по умолчанию, состоящий из пробела и символа табуляции.
tokens=x, y, m-n	Задает элементы, передаваемые из каждой строки в тело цикла **For** при каждой итерации. В результате размещаются дополнительные имена переменных. Форма m-n задает диапазон, указывающий элементы с m-го по n-ый. Если последним символом строки tokens= является звездочка (*), то размещается дополнительная переменная, в которую помещается остаток строки после разбора последнего элемента.
Usebackq	Задает возможность использования кавычек для имен файлов в параметре *МножествоИменФайлов*. Задает исполнение строки, заключенной в обратные кавычки, как команды, а строки в одиночных кавычках — как команды в символьной строке.

Инструкция **For /F** может быть использована применительно к отдельной строке. Для этого необходимо поместить параметр *МножествоИменФайлов* между скобками в одиночные кавычки, то есть ('*МножествоИменФайлов*'). При разборе параметр будет воспринят как одиночная строка ввода из пакетного файла.

Кроме того, **For /F** можно использовать для разбора вывода другой команды. Для этого необходимо поместить параметр *МножествоИменФайлов* между скобками в обратные кавычки, то есть (`*МножествоИменФайлов*`). Он будет воспринят как командная строка, которая передается дочернему командному интерпретатору **Cmd.exe**, а результаты работы команды помещаются в память и разбираются, как если бы они являлись файлом.

Следует отдельно отметить, что в рассматриваемых формах команды **For** может быть реализована подстановка переменных на основе модификаторов (см. подраздел 3.1 настоящей лабораторной работы). Разрешаются все варианты синтаксических конструкций, приведенных в табл. 3.1 и 3.2.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС **(Пуск | Справка и поддержка)** в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **For /?** и нажав **Enter** для ввода.

Задача №3.3.1в. Исследовать способы применения команды циклической обработки данных **For** на конкретных примерах.

1. Скопируйте файлы каталога, путь к которому задайте самостоятельно, в точку назначения, заданную путем D:\Temp\. При копировании воспользуйтесь любым методом, изученным ранее.

2. К каждому из файлов, местоположение которых определено путем D:\Temp\, добавьте символ «!» в начале имени, воспользовавшись командой циклической обработки данных.

3. Подсчитать количество каталогов на локальном диске, воспользовавшись командой циклической обработки данных, в процессе выполнения выводя результат в переменную среды, выбранную самостоятельно. Проверьте полученный результат в файловом диспетчере **Total Commander** (**Файл | Подсчитать занимаемое место**), предварительно выделив содержимое локального диска.

4. Модифицируйте пакетный файл, полученный в предыдущем задании, воспользовавшись командой циклической обработки данных таким образом, чтобы в процессе его выполнения отображалось определенное количество раз выражение «********* the For command **********».

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в окне командной оболочки наберите соответствующую команду с необходимыми ключами,
- нажмите **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет, заполнив табл. 3.7.

Таблица 3.7. Результаты выполнения команды **For**

№ п/п.	Команда с ключами	Результат и вывод по способу применения команды
1.		

2.		
3.		
4.		

Контрольный вопрос:

Как Вы думаете, почему в варианте задания №3 **задачи №3.3.1в** количество подсчитанных каталогов с помощью команды **For** отличается от результата, полученного в файловом диспетчере **Total Commander**?

VI. Обработка условий в пакетных файлах осуществляется командой **If**.

Синтаксис команды **If**:

If [**not**] **errorlevel** *число команда* [**else** *выражение*],
If [**not**] *строка1==строка2 команда* [**else** *выражение*],
If [**not**] **exist** *имя_файла команда* [**else** *выражение*],

Если расширения командного интерпретатора разрешены, следует использовать следующий синтаксис:

If [**/i**] *строка1 оп_сравнения строка2 команда* [**else** *выражение*],
If cmdextversion *число команда* [**else** *выражение*],
If defined *переменная команда* [**else** *выражение*],

где параметр:

not — задает выполнение команды в случае невыполнения условия.
errorlevel *число* — условие выполняется, если предыдущая команда завершилась с кодом, равным или большим *числа*. С помощью этого параметра коды завершения можно использовать в качестве условий.
команда обрабатывается в случае выполнения условия.
строка1==строка2 — условие выполняется, если *строки1* и *2* совпадают. Строки могут быть заданы явно или могут быть пакетными переменными.
exist *имя_файла* — условие выполняется, если существует файл с именем *имя_файла*. Команда **If** не может применяться непосредственно для проверки существования каталога, но в каждом каталоге существует устройство **Nul**, которое может быть использовано для этой цели: **If exist** c:\Mydir\Nul **Echo** «Каталог существует» **.**
оп_сравнения — трехзначный оператор сравнения, допустимые значения которого приведены в табл. 3.8. Пример: **If** %**errorlevel**% LEQ 1 **Goto** Okay.
/i — сравнивает строки. При использовании **/i** применительно к конструкции *строка1==строка2*, где *строки* состоят из цифр, последние преобразуются в числа, с которыми, в свою очередь, выполняется сравнение.
cmdextversion *число* — условие выполняется, если номер внутренней версии, связанный с расширениями командного интерпретатора, равен или больше *числа*.
defined *переменная* — условие выполняется в случае, если *переменная* определена.
выражение — определяет команду с ключами, выполняемую в случае условия **Else**.

Таблица 3.8. Допустимые значения *оп_сравнения*

Оператор	Описание	Оператор	Описание
EQU	Равно	LEQ	меньше или равно
NEQ	не равно	GTR	Больше
LSS	Меньше	GEQ	больше или равно

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС **(Пуск | Справка и поддержка)** в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **If /?** и нажав **Enter** для ввода.

Задача №3.3.1г. Исследовать способы применения команды обработки условия **If** на конкретных примерах.

Модифицируйте пакетный файл, полученный в предыдущем задании таким образом, чтобы выполнялись следующие условия:

1. Если не существует каталог D:\Temp\MyFont\, создайте его любым способом, изученным ранее. В противном случае выведите сообщение «Folder exists» (Каталог существует).
2. Если в каталоге D:\Temp\MyFont\ не существует файлов-шрифтов, скопируйте любые три одним из методов, изученных ранее, из системного каталога c:\Windows\Fonts\. В противном случае выведите сообщение «Fonts exist» (Шрифты присутствуют).
3. Если в каталоге D:\Temp\MyFont\ существует файлы, удалите каталог вместе с его содержимым, изученным ранее способом и выведите сообщение «Folder deleted». В противном случае выведите сообщение «Folder is empty. Deleting is senseless» (Каталог пуст. Удаление бессмысленно).

При выполнении задания используйте следующие инструкции:

- по каждому из пунктов задания в командном файле наберите соответствующий код из команд с необходимыми ключами,
- сохраните модифицированный пакетный файл,
- воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет (табл. 3.9).

Таблица 3.9. Результаты выполнения команды **If**

№ п/п.	Код из команд с ключами	Результат и вывод по способу применения команды
1.		
2.		
3.		

VII. Следующая команда **Goto**, эквивалентная безусловному переходу в языке высокого уровня, в пакетной программе реализует передачу управления ОС Windows в строку,

определенную символьной меткой. Когда метка найдена, выполнение продолжается со следующей за ней строки. Команда **Goto** неразрывно связана с описанной выше командой **If**, в совокупности обеспечивающей возможность сложных программных конструкций с переходами.

Синтаксис команды **Goto**:

Goto *метка*,

где параметр:

Метка — это строка в пакетной программе, на которую выполняется переход. В пакетном файле она должна начинаться с двоеточия (**:**). Если строка начинается с двоеточия, все присутствующие в ней команды обработаны не будут. Синтаксически она может включать пробелы, но не может включать другие разделители, такие как точка с запятой или знак равенства. При этом используются только первые восемь знаков *метки* (метки: hithere0, hithere01 и hithere02 воспринимаются интерпретатором команд как эквивалентные).

Если расширения командного интерпретатора разрешены и в команде **Goto** используется метка **:Eof**, управление будет передано в конец текущего пакетного файла для выхода из него без назначения метки. В синтаксис команды обязательно должно быть включено двоеточие (**:**), то есть **Goto :Eof**.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС **(Пуск | Справка и поддержка)** в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Goto /?** и нажав **Enter** для ввода.

Задача №3.3.1д. Исследовать способы применения команды перехода **Goto** на конкретных примерах.

1. Модифицируйте существующий пакетный файл, введя в него следующий текст:

Pause
Echo.
Format A:
If not Errorlevel 1 **Goto** End
Echo.
Echo *** Error of formatting ***
Rem *** Ошибка форматирования ***
:End
Echo.
Echo *** The end of batch program ***
Rem *** Конец пакетной программы ***
Echo.
Pause

2. Сохраните текст пакетного файла.

При выполнении задания используйте следующие инструкции:

- воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,
- изучите пример и полученный с его помощью результат,
- сделайте вывод о проделанной работе и запишите его в отчет.

Контрольный вопрос:

Каким образом нужно модифицировать текст пакетного файла, чтобы в случае ошибки выводилось сообщение «*** Your disk has errors or no disk in drive E. Insert a new disk! ***» (Ваш диск содержит ошибки или отсутствует в дисководе E. Вставьте новый диск!) и управление передавалось бы в начало процедуры форматирования? Запишите в отчет модифицированный текст.

VIII. Вызов одного пакетного файла из другого без завершения его выполнения осуществляется командой **Call**. Эта команда эквивалентна вызову процедуры из основного тела программы. Она принимает метки в качестве объекта вызова и используется только в сценариях или пакетных файлах; при вызове из командной строки команда **Call** игнорируется.

Синтаксис команды **Call**:

Call [[*диск***:**][*путь*] *имя_файла* [*пакетные_параметры*]] [**:***метка* [*аргументы*]],

где параметр:

[*диск*:][*путь*] *имя_файла* — задает имя и расположение пакетного файла.

пакетные_параметры — задает данные командной строки, используемые программой пакетной обработки, включая параметры командной строки, имена файлов, пакетные параметры (%0-%9) или переменные (например, *%baud%*).

:метка — указывает метку, на которую должно быть передано управление программы пакетной обработки. При использовании с этим параметром создается новый контекст пакетного файла, а управление передается инструкции, следующей за указанной меткой.

аргументы — задает данные командной строки, которые передаются в новую программу пакетной обработки, начинающуюся с **:***метки*, включая параметры командной строки, имена файлов, пакетные параметры или переменные.

Необходимо отметить, что при использовании команды **Call** символы перенаправления ввода-вывода и «каналы» не допускаются. Кроме того, может быть реализована подстановка переменных на основе модификаторов (см. подраздел 3.1). При этом разрешаются все варианты синтаксических конструкций, приведенных в табл. 3.1 и 3.2.

Дополнительная информация по данной команде, а также примеры ее использования доступны в справке ОС **(Пуск | Справка и поддержка)** в соответствующем разделе. Справку также можно получить, набрав в окне командной оболочки строку **Call /?** и нажав **Enter** для ввода.

Задача №3.3.1е. Исследовать способы применения команды вызова пакетного файла **Call** на конкретных примерах.

1. Создайте новый (дочерний) пакетный файл, воспользовавшись любым текстовым редактором. Имя пакетного файла выберите самостоятельно.
2. Введите в дочерний пакетный файл процедуру форматирования гибкого диска, учитывающую переход в начало процедуры в случае ошибки, из приведенного выше примера.
3. Модифицируйте родительский пакетный файл, удалив из него лишние команды и добавив ссылку на дочерний пакетный файл для его вызова.
4. Сохраните тексты обоих пакетных файлов.

При выполнении пунктов 1-4 задания используйте следующие инструкции:

- воспользовавшись командой **Start** и указав путь к родительскому файлу, запустите его на выполнение, нажав **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- запишите полученную информацию в отчет.

5. Вспомните команду форматирования **Format** и ее параметры.
6. Модифицируйте родительский и дочерний файлы таким образом, чтобы осуществилась передача из родительского файла двух значений параметров (*%переменная*) команды **Format** (см. подраздел 3.5 настоящей лабораторной работы), находящейся внутри дочернего файла. Обратите внимание на то, что в таблице подраздела 3.5 передаваемые параметры команды **Format** имеют числовое (%0-%9), а не символьное представление.
7. Сохраните тексты обоих пакетных файлов.

При выполнении пунктов 5-7 задания используйте следующие инструкции:

- воспользовавшись командой **Start** и указав путь к родительскому файлу с параметрами для команды **Format**, запустите его на выполнение, нажав **Enter** для ввода,
- изучите полученный результат и сделайте вывод о проделанной работе,
- перенесите тексты модифицированных пакетных файлов, а также значения используемых пакетных параметров в отчет.

Контрольный вопрос:

Что такое рекурсивный вызов пакетного файла?
Сколько звеньев рекурсии может быть при рекурсивном вызове пакетных файлов?

IX. Следующие две команды и последние из основного набора, предназначены для задания начала и конца области определения локальных переменных среды в пакетном файле. Изменения среды являются локальными для пакетного файла и задаются командой **Setlocal**. Локальное окружение используется до тех пор, пока не встретится команда **Endlocal** или не будет достигнут конец пакетного файла (**Eof**), при этом командный интерпретатор восстанавливает первоначальные параметры.

Синтаксис команд **Setlocal** и **Endlocal**:

Setlocal {**enableextension | disableextensions**} {**enabledelayedexpansion | disabledelayedexpansion**},
:
Endlocal,

где параметр:

enableextension — включает расширения командного интерпретатора до появления соответствующей команды **Endlocal**, вне зависимости от состояния расширений командного интерпретатора перед командой **Setlocal**.

disableextensions — выключает расширения командного интерпретатора до появления соответствующей команды **Endlocal**, вне зависимости от состояния расширений командного интерпретатора перед командой **Setlocal**.

enabledelayedexpansion — включает расширения переменной среды с задержкой до появления соответствующей команды **Endlocal**, вне зависимости от состояния расширений командного интерпретатора перед командой **Setlocal**.

disabledelayedexpansion — выключает расширения переменных среды с задержкой до появления соответствующей команды **Endlocal**, вне зависимости от состояния расширений командного интерпретатора перед командой **Setlocal**.

Вертикальное двоеточие (:) иллюстрирует последовательность определенных команд, расположенных между **Setlocal** и **Endlocal**, ограничивающие область локальных переменных среды**.** Кроме того, что допускается использование нескольких вложенных пар команд **Setlocal** и **Endlocal**.

Дополнительная информация по данным командам, а также примеры их использования доступны в справке ОС **(Пуск | Справка и поддержка)** в соответствующих разделах. Справку также можно получить, набрав в окне командной оболочки строку **Setlocal /?** или **Endlocal /?** и нажав **Enter** для ввода.

Задача №3.3.1ж. Исследовать применение команд локализации переменных среды **Setlocal** и **Endlocal** на конкретном примере.

1. Модифицируйте существующий пакетный файл, введя в него следующий текст, иллюстрирующий локальное изменение переменных среды:

```
@Echo off
Echo.
Echo *Local changing the environment variables*
Rem *Локальное изменение переменных среды*
Setlocal
Path=C:\Windows\system32\help;%path%
Call help>C:\help.out
Endlocal
Start notepad C:\help.out
Pause
```

2. Сохраните текст пакетного файла.

При выполнении задания используйте следующие инструкции:

- воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,
- изучите пример и полученный с его помощью результат,
- сделайте вывод о проделанной работе и запишите его в отчет.

Контрольный вопрос:

Что происходит в рассмотренном примере между командами **Setlocal** и **Endlocal** в процессе выполнения пакетного файла?

Каким образом меняется локальная переменная среды **Path** после выполнения команды **Endlocal?**

3.3.2. Учебное задание №2. Создание пакетного файла, реализующего определенную последовательность действий в ОС Windows.

Порядок выполнения:

I. Создайте новый пакетный файл, воспользовавшись любым текстовым редактором. Имя пакетного файла выберете самостоятельно.

II. Изучите выбранный вариант задания подраздела 3.5.

III. Синтезируйте алгоритм работы пакетного файла.

IV. Выберите необходимый набор команд для реализации алгоритма.

V. С помощью выбранного набора команд запрограммируйте сценарий в виде пакетного файла, реализующего определенную последовательность действий в среде ОС Windows.

VI. Сохраните текст пакетного файла.

При выполнении задания используйте следующие инструкции:

- воспользовавшись командой **Start** и указав путь к пакетному файлу, запустите его на выполнение, нажав **Enter** для ввода,
- изучите полученный результат,
- перенесите алгоритм, блок-схему и текст разработанного пакетного файла в отчет.
- сделайте вывод о проделанной работе и запишите его в отчет.

Дополнительную информацию по возможностям командной оболочки, а также все множество команд доступных при работе с ней наряду с параметрами и примерами применения можно получить в справке ОС **(Пуск | Справка и поддержка)** в разделах **«Общие сведения о командной оболочке», «Справочник по параметрам командной строки»** и **«Новые средства командной строки».**

3.4. Содержание отчета по лабораторной работе

Отчет по лабораторной работе оформляется в соответствии с требованиями государственного стандарта и должен содержать:

1) титульный лист (**Приложение 6**);

2) описание и цель работы;

3) краткое описание служебных команд и утилит командной оболочки, предназначенных для построения и организации пакетных файлов и сценариев в среде ОС Windows (см. подраздел 3.1 настоящей лабораторной работы);

4) результаты исследований работы служебных команд и утилит в соответствии с учебными заданиями лабораторной работы;

5) заполненные таблицы учебных заданий лабораторной работы;

6) письменные ответы на контрольные вопросы, размещенные в соответствующих учебных заданиях лабораторной работы;

7) алгоритмы, блок-схемы и тексты пакетных файлов;

8) выводы о проделанной работе.

3.5. Варианты заданий к лабораторной работе

Варианты заданий для выполнения лабораторной работы представлены в табл. 3.10 и 3.11, имеющие столбцы с заглавиями, указывающими на их причастность к определенной задаче (например, «Задача №3.3.1a»). Для того чтобы выбрать требуемый вариант для выполнения задания лабораторной работы, необходимо из столбца с номером текущего задания выбрать строку номера варианта, определяемую порядковым номером обучающегося в списке группы.

Таблица 3.10. Варианты заданий к лабораторной работе

Учебное задание №1						
Вар. №	**Задача №3.3.1a**				**Задача №3.3.1e**	
	Переменная среды				**Параметры команды Format**	
	a	**b**	**C**	**Result**	**%1**	**%2**
1.	24	11	35	(a + b – c) * 10	/v:System	/a:512
2.	AA	01	C1	a * 5 – b / 5 + c	/a:512	/q
3.	12	33	10	a * 4 / c – b * 2	/v:IICT	/a:1024
4.	25	A3	B4	a – b * 3 – c / 2	/a:1024	/q
5.	49	02	65	a – b * b + c / 5	/v:VMgroup	/a:2048
6.	21	99	12	(b *(a + c)) / 3	/a:2048	/q
7.	BC	BC	CB	10 + (a – b) * c	/v:MyCore	/a:4096
8.	01	94	04	(b – c) / (a * 9)	/a:4096	/q
9.	84	D2	2A	a / 10 – (b * c)	/v:Useless	/a:8192
10.	10	39	92	a * a + b – c / 2	/q	/v:MyDisk
11.	D1	CC	1C	a * b * c – b / c	/v:Temp	/q
12.	FF	00	F1	(a – c) * b + 25	/q	/v:Apps
13.	CA	DA	FA	a * b * (FA – c)	/v:Double	/q
14.	45	78	87	((c – b) *a) / 8	/q	/v:Home Use

15.	88	88	00	(a – b) / (1 – c)	/v:MyDocs	/q
16.	75	93	02	(b + c) * 2 –a	/q	/v:VMComp
17.	A1	CD	0E	a*a*b*b*c*c	/v:MyList	/a:512
18.	C4	EA	E3	(a + b + c) * a	/a:512	/q
19.	44	55	33	c *(b – a) / 10	/v:Admin	/a:1024
20.	B3	E5	DE	(a * b * c) / 25	/a:1024	/q
21.	BB	ED	AE	(a – b) * 2 – c	/v:SysCore	/a:2048
22.	55	56	31	(b – a) * 31 – c	/a:2048	/q
23.	D1	EC	EE	(c – b) * 5 / a	/v:Kernel	/a:4096
24.	D6	E6	FE	(a + b) * c – 11	/a:4096	/q
25.	71	65	32	(a – b) *c / 32	/v:User	/a:8192
26.	84	32	10	(a * (b + c)) / 5	/a:8192	/q

Таблица 3.11. Варианты заданий к лабораторной работе

Учебное задание №2	
Вар. №	**Описание пакетного файла**
1.	Пакетный файл, предназначенный для резервного копирования файлов с определенным расширением из разных каталогов с возможностью создания резервного каталога, в случае его отсутствия в системе. Расширение файлов для копирования задается в качестве пакетного параметра. Резервное копирование осуществляется каждый четверг в 22:00. В течение 3 минут после копирования выводится сообщение «Резервное копирование в каталог <путь> завершено» и далее происходит автоматическое выключение системы с принудительным закрытием всех работающих приложений.
2.	Пакетный файл, предназначенный для организации процесса поиска и отображения текстового файла. Поиск осуществляется по всем локальным дискам. Имя текстового файла задается пакетным параметром. После того как необходимый файл найден, в автоматическом режиме осуществляется его отображение в текстовом процессоре «Блокнот».
3.	Пакетный файл, предназначенный для копирования каталога с его содержимым в заданное место назначения. Копируемый каталог и место назначения задаются в качестве пакетных параметров. После копирования каталога файл-отчет, содержащий информацию о количестве скопированных файлов и их месте расположения, в автоматическом режиме загружается в текстовый процессор «Блокнот».
4.	Пакетный файл, предназначенный для перемещения каталога с его содержимым в заданное место назначения с запросом на удаление, перемещаемого каталога. Перемещаемый каталог и место назначения задаются в качестве пакетных пара-

	метров. После перемещения каталога в отдельный файл выводится отчет, содержащий два дерева каталогов тех мест, откуда и куда было осуществлено перемещение. В конце выводится сообщение вида «Отчет о перемещении находится в каталоге <путь>».
5.	Пакетный файл, предназначенный для копирования каталога и включенных в него файлов, расположенных в месте, заданном определенным путем. Полный путь расположения и маска копируемых файлов задаются в качестве пакетных параметров. Если в результирующем каталоге уже находятся копирующиеся файлы, то повторное копирование должно сопровождаться выдачей предупреждающего сообщения о существовании файлов. В конце выводится сообщение вида «Копирование файлов из каталога <путь> в каталог <путь> завершено».
6.	Пакетный файл, предназначенный для создания отчета, содержащего «Software part» (программная часть), включающую информацию о содержимом корневых каталогов всех логических дисков в системе и «Hardware part» (аппаратная часть), включающую сведения о конфигурации компьютера и ОС, сведения о безопасности, параметры оборудования, такие как ОЗУ, дисковое пространство, сетевые карты и другие. Файл-отчет копируется в некоторый сетевой каталог, задаваемый пакетным параметром, под именем, отражающим имя компьютера, с которого получен этот отчет. В конце выводится сообщение вида «Отчет находится в сетевом каталоге <путь>».
7.	Пакетный файл, предназначенный для углубленной проверки жесткого диска с созданием файла отчета, путь к которому задается в качестве пакетного параметра. Проверка жесткого диска осуществляется ежедневно в 21:00. В течение 20 секунд по окончании проверки диска выводится сообщение «Проверка диска завершена. Файл-отчет находится в каталоге <путь >» и далее осуществляться автоматическая перезагрузка системы.
8.	Пакетный файл, предназначенный для резервного копирования файлов системной папки Windows с возможностью создания резервного каталога, в случае его отсутствия в системе. Путь к резервному каталогу задается в качестве пакетного параметра. Резервное копирование осуществляется ежедневно в 23:00. В течение 2 минут после копирования выводится «Резервное копирование в каталог <путь> завершено» и далее происходит автоматическое выключение системы с принудительным закрытием всех работающих приложений.
9.	Пакетный файл, предназначенный для архивирования и шифрования указанного каталога с его содержимым. Архивируется каталог-источник с помощью существующего в системе архиватора (например, WinRar), вызов которого осуществляется непосредственно из пакетного файла. Архив в дальнейшем шифруется и сохраняется в определенном месте на жестком диске. Пути к каталогу-источнику и месту назначения задаются в качестве пакетных параметров. В конце выводится сообщение вида «Шифрованный архив сохранен в каталог <путь>».
10.	Пакетный файл, предназначенный для удаления файлов по маске, расположенных в месте, заданном определенным путем. Полный путь расположения и маска

	удаляемых файлов задаются в качестве пакетных параметров. В процессе необходимо осуществлять запрос на подтверждение удаления. В конце выводится сообщение вида «Стерто файлов: <количество> из каталога <путь>».
11.	Пакетный файл, предназначенный для организации процесса поиска и сравнения оригинального и резервной копии (.bak) одного и того же файла. Если оригинальный файл найден, то осуществляется его сравнение с резервной копией. Отличия, найденные при сравнении, передаются в отчет, который сохраняется в определенном месте. Имя резервной копии файла и путь к месту назначения, где сохраняется отчет о сравнении, задаются в качестве пакетных параметров. В конце выводится сообщение вида «Отчет сохранен в каталог <путь>».
12.	Пакетный файл, предназначенный для создания отчета, содержащего «Software part» (программная часть), включающую информацию об присутствующих в системе загруженных драйверах и «Hardware part» (аппаратная часть), включающую сведения о конфигурации компьютера и ОС, сведения о безопасности, параметры оборудования, такие как ОЗУ, дисковое пространство и другие. Файл-отчет копируется в некоторый сетевой каталог, задаваемый пакетным параметром, под именем, отражающим IP-адрес компьютера, с которого получен этот отчет. В конце выводится сообщение вида «Отчет находится в каталоге <путь>».
13.	Пакетный файл, предназначенный для подсчета файлов в каталоге, заданном определенным путем. Полный путь расположения и расширение подсчитываемых файлов задаются в качестве пакетных параметров. Организовать отчет с возможностью дописывания в него информации вида «Каталог <путь> содержит <количество> файлов с <расширение> расширением»
14.	Пакетный файл, предназначенный для резервного копирования каталога, заданного определенным путем, и содержащихся в нем файлов с возможностью создания резервного каталога, в случае его отсутствия в системе. Пути к каталогу-источнику и месту назначения задаются в качестве пакетных параметров. Резервное копирование осуществляется ежедневно в 23:59. В течение 5 секунд после копирования выводится сообщение «Резервное копирование в каталог <путь> завершено» и далее происходит автоматическая перезагрузка системы.
15.	Пакетный файл, предназначенный для отражения статистики по атрибутам файлов в каталоге, заданном определенным путем. В каталоге файлы с определенным атрибутом подсчитываются, а их количество передается в текстовый файл статистики с дописыванием в него информации вида «Файлов с атрибутом <атрибут>: <количество>». Пути к каталогу и месту назначения, где сохраняется файл статистики, задаются в качестве пакетных параметров. В конце выводится сообщение вида «Отчет сохранен в каталог <путь>».
16.	Пакетный файл, предназначенный для копирования файлов с определенным расширением и путем в заданное место назначения с их последующим шифрованием. Расширение файлов для копирования, полный путь расположения и путь к месту назначения задаются в качестве пакетных параметров. В конце выводится сообщение вида «Копирование файлов из каталога <путь> в каталог <путь> завершено. Шифрование скопированных данных завершено».
17.	Пакетный файл, предназначенный для организации процесса поиска и сравнения оригинального и резервной копии (.bak) одного и того же файла. Имя резервной копии файла передается в качестве пакетного параметра. Если оригинальный файл найден, то осуществляется его сравнение с резервной копией. Отличия,

	найденные при сравнении, передаются в отчет, который, в автоматическом режиме загружается в текстовый процессор «Блокнот».
18.	Пакетный файл, предназначенный для создания отчета, включающего информацию о присутствующих в системе загруженных драйверах и отображающего список приложений и служб, выполняющихся на компьютере. Файл-отчет копируется в некоторый сетевой каталог, задаваемый пакетным параметром, под именем, отражающим MAC-адрес компьютера, с которого получен этот отчет. В конце выводится сообщение вида «Отчет находится в сетевом каталоге <путь>».
19.	Пакетный файл, предназначенный для копирования файлов, определяемых маской и путем, в заданное место назначения с их последующим архивированием. Архивирование осуществляется с помощью доступного в системе архиватора (например, WinRar), вызов которого осуществляется непосредственно из пакетного файла. Маска файлов для копирования, полный путь расположения и путь к месту назначения задаются в качестве пакетных параметров. В конце выводится сообщение вида «Копирование файлов из каталога <путь> в каталог <путь> завершено. Архивирование скопированных данных завершено».
20.	Пакетный файл, предназначенный для удаления файлов с определенным расширением, расположенных в месте, заданном определенным путем. Полный путь расположения и расширение удаляемых файлов задаются в качестве пакетных параметров. В процессе необходимо осуществлять запрос на подтверждение удаления. После удаления в отдельный файл выводится отчет, содержащий список удаленных файлов с их полным путем, который, в свою очередь, в автоматическом режиме загружается в текстовый процессор «Блокнот».
21.	Пакетный файл, предназначенный для создания отчета событий и их свойств из журнала событий на локальном компьютере. Журналы событий: Application, System и Security. Файл-отчет копируется в некоторый сетевой каталог, задаваемый пакетным параметром, под именем, отражающим IP-адрес компьютера, с которого получен этот отчет. В конце выводится сообщение вида «Отчет находится в сетевом каталоге <путь>». Примечание: если необходимо, смените сервер сценариев ОС Windows.
22.	Пакетный файл, предназначенный для перемещения файлов с определенным расширением из каталога-источника в заданное место назначения. Расширение файлов, каталог-источник и место назначения задаются в качестве пакетных параметров. После перемещения отчет, содержащий список перемещенных файлов с путем, загружается в текстовый процессор «Блокнот».
23.	Пакетный файл, предназначенный для резервного копирования каталога, заданного определенным путем, и содержащихся в нем файлов с возможностью создания резервного каталога, в случае его отсутствия в системе. Пути к каталогу-источнику и месту назначения задаются в качестве пакетных параметров. Резервное копирование осуществляется ежедневно в 23:30. В течение 10 секунд после копирования выводится сообщение «Резервное копирование в каталог <путь> завершено» и далее происходит автоматическая перезагрузка системы.
24.	Пакетный файл, предназначенный для копирования дерева каталогов из каталога-источника в заданное место назначения. Копируемый каталог-источник и место назначения задаются в качестве пакетных параметров. После копирования отчет, содержащий дерево каталогов с местом его расположения, в автоматическом режиме загружается в текстовый процессор «Блокнот».

25.	Пакетный файл, предназначенный для организации процесса поиска и копирования файлов с определенным расширением. Поиск осуществляется по всем локальным дискам. Расширение файлов и место назначения, куда необходимо копировать файлы, задаются в качестве пакетных параметров. В конце выводится сообщение вида «Файлы найдены и скопированы в каталог <путь>».