ЛАБОРАТОРНАЯ РАБОТА №2 ИНТЕРПРЕТАТОР КОМАНДНОЙ СТРОКИ ОС MS WINDOWS

Часть 2. Язык интерпретатора и командные файлы

Цель работы – знакомство с языком интерпретатора командной строки ОС MS Windows и командными файлами

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Язык интерпретатора Cmd.exe. Командные файлы

Язык оболочки командной строки (shell language) в Windows реализован в виде командных (или пакетных) файлов. **Командный файл** в Windows — это обычный текстовый файл с расширением bat или cmd, в котором записаны допустимые команды ОС (как внешние, так и внутренние), а также некоторые дополнительные инструкции и ключевые слова, придающие командным файлам некоторое сходство с программами, написанными на языке программирования. Например, если записать в файл deltmp.bat следующие команды:

```
C:\
CD %TEMP%
DEL /F *.tmp
```

и запустить его на выполнение (аналогично исполняемым файлам с расширением com или exe), то мы удалим все файлы во временной директории Windows. Таким образом, исполнение командного файла приводит к тому же результату, что и последовательный ввод записанных в нем команд. При этом не проводится никакой предварительной компиляции или проверки синтаксиса кода; если встречается строка с ошибочной командой, то она игнорируется. Очевидно, что если приходится часто выполнять одни и те же действия, то использование командных файлов может сэкономить много времени.

1.1.1 Вывод сообщений и дублирование команд

По умолчанию команды пакетного файла перед исполнением выводятся на экран, что выглядит не очень эстетично. С помощью команды ECHO OFF можно отключить дублирование команд, идущих после нее (сама команда ECHO OFF при этом все же дублируется). Например,

```
REM Следующие две команды будут дублироваться на экране …
:: эта строка – такой же комментарий, как и предыдущая
DIR C:\
ECHO OFF
:: А остальные уже не будут
```

DIR D:\

Для восстановления режима дублирования используется команда ECHO ON. Кроме этого, можно отключить дублирование любой отдельной строки в командном файле, написав в начале этой строки символ @, например:

```
ECHO ON
:: Команда DIR C:\ дублируется на экране
DIR C:\
:: А команда DIR D:\ — нет
@DIR D:\
```

Таким образом, если поставить в самое начало файла команду @ECHO OFF,то это решит все проблемы с дублированием команд.

В пакетном файле можно выводить на экран строки с сообщениями. Делается это с помощью команды ECHO сообщение

Например:

```
 @ECHO OFF
ECHO Привет!
```

Команда ECHO. (точка должна следовать непосредственно за словом "ECHO") выводит на экран **пустую строку**. Например:

```
@ECHO OFF
ECHO Привет!
ECHO.
ECHO Пока!
```

Часто бывает удобно для просмотра сообщений, выводимых из пакетного файла, предварительно полностью очистить экран командой CLS.

Используя механизм *перенаправления ввода/вывода* (символы > и >>), можно направить сообщения, выводимые командой ECHO, в определенный текстовый файл. Например:

```
@ECHO OFF
ECHO Привет! > hi.txt
ECHO Пока! >> hi.txt
```

С помощью такого метода можно, скажем, заполнять файлы-протоколы с отчетом о произведенных действиях. Например:

```
@ECHO OFF
REM Попытка копирования
XCOPY C:\PROGRAMS D:\PROGRAMS /s
:: Добавление сообщения в файл report.txt в случае удачного завершения копирования
IF NOT ERRORLEVEL 1 ECHO Успешное копирование >> report.txt
```

1.1.2 Использование параметров командной строки

При запуске пакетных файлов в командной строке можно указывать произвольное число параметров, значения которых можно использовать внутри файла. Это позволяет, например, применять один и тот же командный файл для выполнения команд с различными параметрами.

Для доступа из командного файла к параметрам командной строки применяются символы %0, %1, …, %9 или %*. При этом вместо %0 подставляется имя выполняемого пакетного файла, вместо %1, %2, …, %9 — значения первых девяти параметров командной строки соответственно, а вместо %* — все аргументы. Если в командной строке при вызове пакетного файла задано меньше девяти параметров, то "лишние" переменные из %1 – %9 замещаются пустыми строками. Рассмотрим следующий пример. Пусть имеется командный файл copier.bat следующего содержания:

```
@ECHO OFF
CLS
ECHO Файл %0 копирует каталог %1 в %2
XCOPY %1 %2 /S
```

Если запустить его из командной строки с двумя параметрами, например

```
copier.bat C:\Programs D:\Backup
```

то на экран выведется сообщение

```
Файл copier.bat копирует каталог C:\Programs в D:\Backup
```

и произойдет копирование каталога C:\Programs со всеми его подкаталогами в D:\Backup.

При необходимости можно использовать более девяти параметров командной строки. Это достигается с помощью команды `SHIFT`, которая изменяет значения замещаемых параметров с `%0` по `%9`, копируя каждый параметр в предыдущий, то есть значение `%1` копируется в `%0`, значение `%2` – в `%1` и т.д. Замещаемому параметру `%9` присваивается значение параметра, следующего в командной строке за старым значением `%9`. Если же такой параметр не задан, то новое значение `%9` — пустая строка.

Пример 1. Пусть командный файл my.bat вызван из командной строки следующим образом:

```
my.bat p1 p2 p3
```

Тогда %0=my.bat, %1=p1, %2=p2, %3=p3, параметры %4 – %9 являются пустыми строками. После выполнения команды SHIFT значения замещаемых параметров изменятся следующим образом: %0=p1, %1=p2, %2=p3, параметры %3 – %9 – пустые строки.

При включении расширенной обработки команд SHIFT поддерживает ключ /n, задающий начало сдвига параметров с номера n, где n может быть числом от 0 до 9.

Например, в следующей команде SHIFT /2 параметр %2 заменяется на %3,

%3 на %4 и т.д., а параметры %0 и %1 остаются без изменений.

Команда, обратная SHIFT (обратный сдвиг), отсутствует. После выполнения SHIFT уже нельзя восстановить параметр (%0), который был первым перед сдвигом. Если в командной строке задано больше десяти параметров, то команду SHIFT можно использовать несколько раз.

В командных файлах имеются некоторые возможности синтаксического анализа заменяемых параметров. Для параметра с номером n (%n) допустимы синтаксические конструкции (операторы), представленные в (табл. 1).

Таблица 1.

Операторы для заменяемых параметров

Операторы	Описание
`%~Fn`	Переменная `%n` расширяется до полного имени файла
`%~Dn`	Из переменной `%n` выделяется только имя диска
`%~Pn`	Из переменной `%n` выделяется только путь к файлу
`%~Nn`	Из переменной `%n` выделяется только имя файла
`%~Xn`	Из переменной `%n` выделяется расширение имени файла
`%~Tn`	Возвращается дата и время создания (модификации) файла
`%~Zn`	Возвращается размер файла в байтах
`%~$PATH:n`	Проводится поиск по каталогам, заданным в переменной среды `PATH`, и переменная `%n` заменяется на полное имя первого найденного файла. Если переменная `PATH` не определена или в результате поиска не найден ни один файл, эта конструкция заменяется на пустую строку. Естественно, здесь переменную `PATH` можно заменить на любое другое допустимое значение

Данные синтаксические конструкции можно объединять друг с другом, например:

`%~DPn` — из переменной `%n` выделяется имя диска и путь,

`%~NXn` — из переменной `%n` выделяется имя файла и расширение.

Пример 2. Пусть мы находимся в каталоге C:\TEXT и запускаем пакетный файл с параметром `Рассказ.doc` (`%1=Рассказ.doc`). Размер файла 2150 байт, дата создания 12.12.2015, время -12:55. Тогда применение операторов, описанных в табл. 1, к параметру `%1` даст следующие результаты:

%~F1=C:\TEXT\Рассказ.doc

%~D1=C:

```
%~P1=\TEXT\
%~N1=Рассказ
%~X1=.doc
%~DP1=C:\TEXT\
%~NX1=Рассказ.doc
%~T1=12.12.2009 12:55
%~Z1=2150
```

1.1.3 Работа с переменными среды

Внутри командных файлов можно использовать так называемые **переменными среды** (или переменными окружения), каждая из которых хранится в оперативной памяти, имеет свое уникальное имя, а ее значением является **строка**. Стандартные переменные среды автоматически инициализируются в процессе загрузки операционной системы. Такими переменными являются:

- WINDIR, которая определяет расположение каталога Windows,
- TEMP, которая определяет путь к каталогу для хранения временных файлов Windows
- PATH, в которой хранится системный путь (путь поиска), то есть список каталогов, в которых система должна искать выполняемые файлы или файлы совместного доступа (например, динамические библиотеки).

Кроме того, в командных файлах с помощью команды `SET` можно объявлять собственные переменные среды.

1.1.3.1 Получение значения переменной

Для получения значения определенной переменной среды нужно заключить имя этой переменной в символы `%`. Например:

```
@ECHO OFF
CLS
:: Создание переменной MyVar
SET MyVar=Привет
:: Изменение переменной
SET MyVar=%MyVar%
ECHO Значение переменной MyVar: %MyVar%
:: Удаление переменной MyVar
SET MyVar=
ECHO Значение переменной WinDir: %WinDir%
```

При запуске такого командного файла на экран выведется строка

```
Значение переменной MyVar: Привет!
Значение переменной WinDir: C:\WINDOWS
```

1.1.4 Преобразования переменных как строк

С переменными среды в командных файлах можно производить некоторые манипуляции. *Во-первых*, над ними можно производить операцию конкатенации (соединения). Для этого нужно в команде `SET` просто написать рядом значения соединяемых переменных. Например,

```
SET A=Раз
SET B=Два
SET C=%A%%B%
```

После выполнения в файле этих команд значением переменной C будет являться строка `'РазДва'`. Не следует для конкатенации использовать знак `+`, так как он будет воспринят просто в качестве символа. Например, после запуска файл следующего содержания

```
SET A=Раз
SET B=Два
SET C=A+B
ECHO Переменная C=%C%
SET D=%A%+%B%
ECHO Переменная D=%D%
```

на экран выведутся две строки:

```
Переменная C=A+B
Переменная D=Раз+Два
```

Во-вторых, из переменной среды можно **выделять подстроки** с помощью конструкции `%имя_переменной:~n1,n2%`, где число `n1` определяет смещение (количество пропускаемых символов) от начала (если `n1` положительно) или от конца (если `n1` отрицательно) соответствующей переменной среды, а число `n2` – количество выделяемых символов (если `n2` положительно) или количество последних символов в переменной, которые не войдут в выделяемую подстроку (если `n2` отрицательно). Если указан только один отрицательный параметр `-n`, то будут извлечены последние `n` символов. Например, если в переменной хранится строка `"21.12.2015"` (символьное представление текущей даты при определенных региональных настройках), то после выполнения следующих команд

```
SET dd1=%DATE:~0,2%
SET dd2=%DATE:~0,-8%
SET mm=%DATE:~-7,2%
SET yyyy=%DATE:~-4%
```

новые переменные будут иметь такие значения: `%dd1%=21, %dd2%=21, %mm%=12, %yyyy%=2015.`

В-третьих, можно выполнять процедуру замены подстрок с помощью конструкции `%имя_переменной:s1=s2%` (в результате будет возвращена строка, в которой каждое вхождение подстроки `s1` в

соответствующей переменной среды заменено на s2). Например, после выполнения команд
SET a=123456
SET b=%a:23=99%
в переменной b будет храниться строка "199456". Если параметр s2 не указан, то подстрока s1 будет удалена из выводимой строки, т.е. после выполнения команды
SET a=123456
SET b=%a:23=%
в переменной b будет храниться строка "1456".

1.1.5 Операции с переменными как с числами

При включенной расширенной обработке команд (этот режим в Windows используется по умолчанию) имеется **возможность рассматривать значения переменных среды как числа** и производить с ними арифметические вычисления (используются ТОЛЬКО целые числа). Для этого используется команда SET с ключом /A. Ниже приведен пример пакетного файла add.bat, складывающего два числа, заданных в качестве параметров командной строки, и выводящего полученную сумму на экран:

```
@ECHO OFF
:: В переменной M будет храниться сумма
SET /A M=%1+%2
ECHO Сумма %1 и %2 равна %M%
:: Удалим переменную M
SET M=
```

В команде SET с ключом /A могут использоваться операции – (вычитание), * (умножение), / (деление нацело), % (остаток от деления). При использовании знака % в качестве знака операции в **командных файлах** он должен быть записан ДВАЖДЫ.
Рекомендуется при инициализации числовых переменных использовать ключ /A
SET /A col=0

1.1.6 Ввод значения переменной с клавиатуры

Ввод значения переменной при выполнении командного файла выполняется командой SET с ключом /P. Например, для ввода значения переменной M следует использовать команду
SET /P M=[введите M]

Текст подсказки [введите M] будет выведен на экран.

1.1.7 Локальные изменения переменных

Все изменения, производимые с помощью команды SET над переменными среды в командном файле, сохраняются и после завершения

работы этого файла, но действуют только внутри текущего командного окна. Также имеется возможность локализовать изменения переменных среды внутри пакетного файла, то есть автоматически восстанавливать значения всех переменных в том виде, в каком они были **до начала запуска** этого файла. Для этого используются две команды: SETLOCAL и ENDLOCAL. Команда SETLOCAL определяет начало области локальных установок переменных среды. Другими словами, изменения среды, внесенные после выполнения SETLOCAL, будут являться локальными относительно текущего пакетного файла. Каждая команда SETLOCAL должна иметь соответствующую команду ENDLOCAL для восстановления прежних значений переменных среды. Изменения среды, внесенные после выполнения команды ENDLOCAL, уже не являются локальными относительно текущего пакетного файла; их прежние значения не будут восстановлены по завершении выполнения этого файла.

1.1.8 Связывание времени выполнения для переменных

При работе с составными выражениями (группы команд, заключенных в круглые скобки) нужно учитывать, что переменные среды в командных файлах используются в режиме раннего связывания. С точки зрения логики выполнения командного файла это может привести к ошибкам. Например, рассмотрим командный файл 1.bat со следующим содержимым:

```
SET a=1
ECHO a=%a%
SET a=2
ECHO a=%a%
```

и командный файл 2.bat:

```
SET a=1
ECHO a=%a%
(SET a=2
ECHO a=%a% )
```

Казалось бы, результат выполнения этих двух файлов должен быть одинаковым: на экран выведутся две строки: "a=1" и "a=2". На самом же деле таким образом сработает только файл 1.bat, а файл 2.bat два раза выведет строку "a=1".

Данную ошибку можно обойти, если для получения значения переменной вместо знаков процента (%) использовать восклицательный знак (!) и предварительно включить режим связывания времени выполнения командой SETLOCAL ENABLEDELAYEDEXPANSION. Таким образом, для корректной работы файл 2.bat должен иметь следующий вид:

```
SETLOCAL ENABLEDELAYEDEXPANSION
SET a=1
```

```
ECHO a=%a%
(SET a=2
ECHO a=!a!)
```

ВНИМАНИЕ! Приведенный материал необходим для правильной работы команды цикла FOR и будет использован в командных файлах!

1.1.9 Приостановка выполнения командных файлов

Для того, чтобы вручную **прервать выполнение** запущенного bat-файла, нужно нажать клавиши <Ctrl>+<C> или <Ctrl>+<Break>. Однако часто бывает необходимо программно приостановить выполнение командного файла в определенной строке с выдачей запроса на нажатие любой клавиши. Это делается с помощью команды PAUSE. Перед запуском этой команды полезно с помощью команды ECHO информировать пользователя о действиях, которые он должен произвести. Например:

```
ECHO Вставьте дискету в дисковод A: и нажмите любую клавишу
PAUSE
```

Команду PAUSE обязательно нужно использовать при выполнении потенциально опасных действий (удаление файлов, форматирование дисков и т.п.). Например,

```
ECHO Сейчас будут удалены все файлы в C:\Мои документы!
ECHO Для отмены нажмите Ctrl-C
PAUSE
DEL "C:\Мои документы\*.*"
```

1.1.10 Вызов внешних командных файлов

Из одного командного файла можно вызвать другой, просто указав его имя. Например:

```
@ECHO OFF
CLS
REM Вывод списка log-файлов
DIR C:\*.log
:: Передача выполнения файлу f.bat
f.bat
COPY A:\*.* C:\
PAUSE
```

Однако в этом случае после выполнения вызванного файла управление в вызывающий файл не передается, то есть в приведенном примере команда

```
COPY A:\*.* C:\
```

(и все следующие за ней команды) никогда не будет выполнена.

Для того, чтобы вызвать внешний командный файл с последующим возвратом в первоначальный файл, нужно использовать специальную

команду CALL файл
Например:
```
@ECHO OFF
CLS
:: Вывод списка log-файлов
DIR C:\*.log
:: Передача выполнения файлу f.bat
CALL f.bat
COPY A:\*.* C:\
PAUSE
```

В этом случае после завершения работы файла f.bat управление вернется в первоначальный файл на строку, следующую за командой `CALL` (в нашем примере это команда `COPY A:\*.* C:\`).

1.1.11 Операторы перехода GOTO и вызова CALL

Командный файл может содержать метки и команды `GOTO` перехода к этим меткам. Любая строка, начинающаяся с двоеточия :, воспринимается при обработке командного файла как метка. Имя метки задается набором символов, следующих за двоеточием до первого пробела или конца строки.

Пример 3. Пусть имеется командный файл следующего содержания:

```
@ECHO OFF
COPY %1 %2
GOTO Label1
ECHO Эта строка никогда не выполнится
:Label1
:: Продолжение выполнения
DIR %2
```

После того, как в этом файле мы доходим до команды

```
GOTO Label1
```

его выполнение продолжается со строки

```
:: Продолжение выполнения
```

В команде перехода внутри файла `GOTO` можно задавать в качестве метки перехода строку `:EOF`, которая передает управление в конец текущего пакетного файла (это позволяет легко выйти из пакетного файла без определения каких-либо меток в самом его конце).

Для перехода к метке внутри текущего командного файла кроме команды `GOTO` можно использовать и рассмотренную выше команду `CALL`:

```
CALL :метка аргументы
```

При вызове такой команды создается новый контекст текущего пакетного файла с заданными аргументами, и управление передается на

инструкцию, расположенную сразу после метки. Для выхода из такого пакетного файла необходимо два раза достичь его конца. Первый выход возвращает управление на инструкцию, расположенную сразу после строки CALL, а второй выход завершает выполнение пакетного файла. Например, если запустить с параметром "Копия-1" командный файл следующего содержания:

```
@ECHO OFF
ECHO %1
CALL :2 Копия-2
:2
ECHO %1
```

то на экран выведутся три строки:

```
Копия-1
Копия-2
Копия-1
```

Таким образом, подобное использование команды CALL похоже на вызов подпрограмм в языках высокого уровня.

1.1.12 Оператор проверки условия IF

С помощью команды IF … ELSE (ключевое слово ELSE может отсутствовать) в пакетных файлах можно выполнять обработку условий нескольких типов. При этом если заданное после IF условие принимает истинное значение, система выполняет следующую за условием команду (или несколько команд, заключенных в круглые скобки), в противном случае выполняется команда (или несколько команд в скобках), следующие за ключевым словом ELSE.

1.1.12.1 Проверка значения переменной

Первый тип условия используется обычно для проверки значения переменной. Для этого применяются два варианта синтаксиса команды IF:

```
IF [NOT] строка1==строка2 команда1 [ELSE команда2]
```

(квадратные скобки указывают на необязательность заключенных в них параметров) или

```
IF [/I] [NOT] строка1 оператор_сравнения строка2 команда
```

Рассмотрим сначала первый вариант. Условие строка1==строка2 (здесь необходимо писать именно два знака равенства – как и в программах на C/C++) считается истинным при точном совпадении обеих строк. Параметр NOT указывает на то, что заданная команда выполняется лишь в том случае, когда сравниваемые строки не совпадают.

Для *группировки команд* могут использоваться круглые скобки. Иногда использование круглых скобок необходимо для правильной работы команды if…else – например для вывода на экран наибольшего из двух

параметров, с которыми запущен командный файл, следует использовать оператор
if %1 GTR %2 (echo %1) else (echo %2)

Строки могут быть литеральными или представлять собой значения переменных (например, `%1` или `%TEMP%`). Кавычки для литеральных строк **не требуются**. Например,
IF %1==%2 ECHO Параметры совпадают!
IF %1==windows ECHO значение первого параметра - windows

Отметим, что при сравнении строк, заданных переменными, следует проявлять определенную осторожность. Дело в том, что значение переменной может оказаться пустой строкой, и тогда может возникнуть ситуация, при которой выполнение командного файла аварийно завершится. Например, если вы не определили с помощью команды `SET` переменную `MyVar`, а в файле имеется условный оператор типа
IF %MyVar%==C:\ ECHO Ура!!!, то в процессе выполнения вместо `%MyVar%` подставится пустая строка и возникнет синтаксическая ошибка. Такая же ситуация может возникнуть, если одна из сравниваемых строк является значением параметра командной строки, так как этот параметр может быть не указан при запуске командного файла. Поэтому при сравнении строк нужно приписывать к ним в начале какой-нибудь символ, например:
IF -%MyVar%==-C:\ ECHO Ура!!!

С помощью команд `IF` и `SHIFT` можно в цикле обрабатывать все параметры командной строки файла, даже не зная заранее их количества. Например, следующий командный файл (назовем его primer.bat) выводит на экран имя запускаемого файла и все параметры командной строки:

```
@ECHO OFF
ECHO Выполняется файл: %0
ECHO.
ECHO Файл запущен со следующими параметрами...
:: Начало цикла
:BegLoop
IF -%1==- GOTO ExitLoop
ECHO %1
:: Сдвиг параметров
SHIFT
:: Переход на начало цикла
GOTO BegLoop
:ExitLoop
:: Выход из цикла
ECHO.
ECHO Все.
```

Если запустить primer.bat с четырьмя параметрами:
primer.bat A B C D то в результате выполнения на экран выведется следующая информация:
Выполняется файл: primer.bat
Файл запущен со следующими параметрами:
A
B
C
D
Все.

Рассмотрим теперь оператор IF в следующем виде:
IF [/I] строка1 оператор_сравнения строка2 команда

Синтаксис и значение операторов_сравнения представлены в (табл. 2).

Таблица 2.

Операторы сравнения в IF

Оператор	Значение
EQL	Равно
NEQ	Не равно
LSS	Меньше
LEQ	Меньше или равно
GTR	Больше
GEQ	Больше или равно

Пример 4. использования операторов сравнения:
@ECHO OFF
CLS
IF -%1 EQL –Вася ECHO Привет, Вася!
IF -%1 NEQ –Вася ECHO Привет, но Вы не Вася!

Ключ /I, если он указан, задает сравнение текстовых строк **без учета регистра**. Ключ /I можно также использовать и в форме строка1==строка2 команды IF. Например, условие
IF /I DOS==dos …
будет истинным.

1.1.12.2 Проверка существования заданного файла

Второй способ использования команды IF — это проверка существования заданного файла. Синтаксис для этого случая имеет вид:

IF [NOT] EXIST файл команда1 [ELSE команда2]

Условие считается истинным, если указанный файл существует. Кавычки для имени файла не требуются. Приведем пример командного файла, в котором с помощью такого варианта команды IF проверяется наличие файла, указанного в качестве параметра командной строки.

```
@ECHO OFF
IF -%1==- GOTO NoFileSpecified
IF NOT EXIST %1 GOTO FileNotExist

:: Вывод сообщения о найденном файле
ECHO Файл '%1' найден.
GOTO :EOF

:NoFileSpecified
:: Файл запущен без параметров
ECHO В командной строке не указано имя файла.
GOTO :EOF

:FileNotExist
:: Параметр командной строки задан, но файл не найден
ECHO Файл '%1' не найден.
```

1.1.12.3 Проверка наличия переменной среды

Аналогично файлам команда `IF` позволяет проверить наличие в системе определенной переменной среды:

IF DEFINED переменная команда1 [ELSE команда2]

Здесь условие `DEFINED` применяется подобно условию `EXISTS` наличия заданного файла, но принимает в качестве аргумента имя переменной среды и возвращает истинное значение, если эта переменная определена. Например:

```
@ECHO OFF
CLS
IF DEFINED MyVar GOTO :VarExists
ECHO Переменная MyVar не определена
GOTO :EOF
:VarExists
ECHO Переменная MyVar определена,
ECHO ее значение равно %MyVar%
```

1.1.12.4 Проверка кода завершения предыдущей команды

Еще один способ использования команды `IF` — это проверка кода завершения (кода выхода) предыдущей команды. Синтаксис для `IF` в этом

случае имеет следующий вид:
IF [NOT] ERRORLEVEL число команда1 [ELSE команда2]

Здесь условие считается истинным, если последняя запущенная команда или программа завершилась с кодом возврата, равным либо превышающим указанное число.

Рассмотрим командный файл, который копирует файл my.txt на диск C: без вывода на экран сообщений о копировании, а в случае возникновения какой-либо ошибки выдает предупреждение:

```
@ECHO OFF
XCOPY my.txt C:\ > NUL
:: Проверка кода завершения копирования
IF ERRORLEVEL 1 GOTO ErrOccurred
ECHO Копирование выполнено без ошибок.
GOTO :EOF
:ErrOccurred
ECHO При выполнении команды XCOPY возникла ошибка!
```

В операторе `IF ERRORLEVEL ...` можно также применять операторы сравнения чисел, приведенные в табл. 2. Например:
IF ERRORLEVEL LEQ 1 GOTO Case1

Замечание. Иногда более удобным для работы с кодами завершения программ может оказаться использование переменной `%ERRORLEVEL%`. (строковое представление текущего значения кода ошибки `ERRORLEVEL`).

1.1.13 Организация циклов

В командных файлах для организации циклов используются несколько разновидностей оператора `FOR`, которые обеспечивают следующие функции:

- выполнение заданной команды для всех элементов указанного множества;
- выполнение заданной команды для всех подходящих имен файлов;
- выполнение заданной команды для всех подходящих имен каталогов;
- выполнение заданной команды для определенного каталога, а также всех его подкаталогов;
- получение последовательности чисел с заданными началом, концом и шагом приращения;
- чтение и обработка строк из текстового файла;
- обработка строк вывода определенной команды.

1.1.13.1 Цикл FOR ... IN ... DO ...

Самый простой вариант синтаксиса команды `FOR` для командных файлов имеет следующий вид:

FOR %%переменная IN (множество)
DO команда [параметры]

Внимание!

Перед названием переменной должны стоять именно два знака процента (%%), а не один, как это было при использовании команды FOR непосредственно из командной строки!

Пример 5. Если в командном файле заданы строки

```
@ECHO OFF
FOR %%i IN (Раз, Два, Три) DO ECHO %%i
```

то в результате его выполнения на экран будет выведено следующее:

```
Раз
Два
Три
```

Параметр множество в команде FOR задает одну или более текстовых строк, разделенных запятыми, которые необходимо обработать с помощью заданной команды. Скобки здесь **обязательны**. Параметр команда [параметры] задает команду, выполняемую для каждого элемента множества, при этом вложенность команд FOR на одной строке **не допускается**. Если в строке, входящей во множество, используется запятая, то значение этой строки нужно заключить в кавычки. Например, в результате выполнения файла с командами

```
@ECHO OFF
FOR %%i IN ("Раз,Два",Три) DO ECHO %%i
```

на экран будет выведено

```
Раз,Два
Три
```

Параметр %%переменная представляет подставляемую переменную (счетчик цикла), причем здесь могут использоваться только имена переменных, **состоящие из одной буквы**. При выполнении команда FOR заменяет подставляемую переменную текстом каждой строки в заданном множестве, пока команда, стоящая после ключевого слова DO, не обработает все такие строки.

Замечание. Чтобы избежать путаницы с параметрами командного файла %0 — %9, для переменных следует использовать любые символы кроме 0 – 9.

Параметр множество в команде FOR может также представлять одну или несколько групп файлов. Например, чтобы вывести в файл список всех файлов с расширениями txt и prn, находящихся в каталоге C:\TEXT, без использования команды DIR, можно использовать командный файл следующего содержания:

```
@ECHO OFF
```

FOR %%f IN (C:\TEXT*.txt C:\TEXT*.prn) DO ECHO %%f >> list.txt

При таком использовании команды FOR процесс обработки продолжается, пока не обработаются все файлы (или группы файлов), указанные во множестве.

1.1.13.2 Цикл FOR /D ... IN ... DO ...

Следующий вариант команды `FOR` реализуется с помощью ключа /D (directory – каталог):

FOR /D %переменная IN (набор) DO команда [параметры]

В случае, если набор содержит подстановочные знаки, то команда выполняется для всех подходящих имен каталогов, а не имен файлов. Скажем, выполнив следующий командный файл:

```
@ECHO OFF
CLS
FOR /D %%f IN (C:\*.*) DO ECHO %%f
```

мы получим список всех каталогов на диске C:, например:

```
C:\Arc
C:\CYR
C:\MSCAN
C:\Program Files
C:\TEMP
C:\WINNT
```

1.1.13.3 Цикл FOR /R ... IN ... DO ...

С помощью ключа `/R` можно задать **рекурсию** в команде `FOR`:

FOR /R [[диск:]путь] %переменная IN (набор)
DO команда [параметры]

В этом случае заданная команда выполняется для каталога `[диск:]путь`, а также для **всех подкаталогов** этого пути. Если после ключа `R` не указано имя каталога, то выполнение команды начинается с текущего каталога.

Пример 6. Для распечатки всех файлов с расширением txt в текущем каталоге и всех его подкаталогах можно использовать следующий пакетный файл:

```
@ECHO OFF
CLS
FOR /R %%f IN (*.txt) DO PRINT %%f
```

Если вместо набора указана только точка (.), то команда проверяет все подкаталоги текущего каталога. Например, если мы находимся в каталоге C:\TEXT с двумя подкаталогами BOOKS и ARTICLES, то в результате выполнения файла:

```
@ECHO OFF
CLS
```

```
FOR /R %%f IN (.) DO ECHO %%f
```
на экран выведутся три строки:
```
C:\TEXT\.
C:\TEXT\BOOKS\.
C:\TEXT\ARTICLES\.
```

1.1.13.4 Цикл FOR /L ... IN ... DO ...

Ключ `/L` позволяет реализовать с помощью команды `FOR` цикл со счетчиком, в этом случае синтаксис имеет следующий: вид:
```
FOR /L %переменная IN (начало,шаг,конец) DO команда [параметры]
```
Здесь заданная после ключевого слова `IN` тройка (начало, шаг, конец) задает последовательность чисел с заданными началом, концом и шагом приращения. Например, тройка (1, 1, 5) порождает последовательность (1 2 3 4 5), а тройка (5, -1, 1) - последовательность (5 4 3 2 1). Например, в результате выполнения следующего командного файла:
```
@ECHO OFF
CLS
FOR /L %%f IN (1,1,5) DO ECHO %%f
```
переменная цикла `%%f` получит значения от 1 до 5, и на экран будут выведены пять чисел:
```
1
2
3
4
5
```
Числа, получаемые в результате выполнения цикла `FOR /L`, можно использовать в **арифметических вычислениях**. Рассмотрим командный файл my.bat следующего содержания:
```
@ECHO OFF
CLS
FOR /L %%f IN (1,1,5) DO CALL :2 %%f
GOTO :EOF
:2
SET /A M=10*%1
ECHO 10*%1=%M%
```
В третьей строке в цикле происходит вызов нового контекста файла my.bat с текущим значением переменной цикла `%%f` в качестве параметра командной строки, причем управление передается на метку `:2` (см. описание `CALL` в разделе "Изменения в командах перехода"). В шестой строке переменная цикла умножается на десять, и результат записывается в переменную `M`. Таким образом, в результате выполнения этого файла выведется следующая информация:

```
10*1=10
10*2=20
10*3=30
10*4=40
10*5=50
```

1.1.13.5 Цикл FOR /F ... IN ... DO ...

Самые широкие возможности имеет команда `FOR` с ключом `/F`:

```
FOR /F ["ключи"] %переменная IN (набор) DO команда [параметры]
```

Здесь параметр набор содержит имена одного или нескольких файлов, которые по очереди открываются, читаются и обрабатываются. Обработка состоит в чтении файла, разбиении его на отдельные строки текста и выделении из каждой строки заданного числа подстрок. Затем найденная подстрока используется в качестве значения переменной при выполнении основного тела цикла (заданной команды).

По умолчанию ключ `/F` выделяет из каждой строки файла первое слово, очищенное от окружающих его пробелов. Пустые строки в файле пропускаются. Необязательный параметр `"ключи"` служит для переопределения заданных по умолчанию правил обработки строк. Ключи представляют собой заключенную в кавычки строку, содержащую приведенные в (табл. 3) ключевые слова:

Таблица 3.

Ключи в команде FOR /F

Ключ	Описание
`EOL=C`	Определение символа комментариев в начале строки (допускается задание только одного символа)
`SKIP=N`	Число пропускаемых при обработке строк в начале файла
`DELIMS=XXX`	Определение набора разделителей для замены заданных по умолчанию пробела и знака табуляции
`TOKENS=X,Y,M-N`	Определение номеров подстрок, выделяемых из каждой строки файла и передаваемых для выполнения в тело цикла

При использовании ключа `TOKENS=X,Y,M-N` создаются дополнительные переменные. Формат `M-N` представляет собой диапазон подстрок с номерами от `M` до `N`. Если последний символ в строке `TOKENS=` является звездочкой, то создается дополнительная переменная, значением которой будет весь текст, оставшийся в строке после обработки последней подстроки.

Разберем применение этой команды на примере пакетного файла parser.bat, который производит разбор файла myfile.txt:

```
@ECHO OFF
IF NOT EXIST myfile.txt GOTO :NoFile
FOR /F "EOL=; TOKENS=2,3* DELIMS=, " %%i IN (myfile.txt) DO @ECHO %%i %%j %%k
GOTO :EOF
:NoFile
ECHO Не найден файл myfile.txt!
```

Здесь во второй строке производится проверка наличия файла myfile.txt; в случае отсутствия этого файла выводится предупреждающее сообщение. Команда `FOR` в третьей строке обрабатывает файл myfile.txt следующим образом:

Пропускаются все строки, которые начинаются с символа точки с запятой (`EOL=;`).

Вторая и третья подстроки из каждой строки передаются в тело цикла, причем подстроки разделяются пробелами (по умолчанию) и/или запятыми (`DELIMS=,`).

В теле цикла переменная `%%i` используется для второй подстроки, `%%j` — для третьей, а `%%k` получает все оставшиеся подстроки после третьей.

Замечание.Имена переменных i, j, k должны следовать в алфавитном порядке.

В нашем примере переменная `%%i` явно описана в инструкции `FOR`, а переменные `%%j` и `%%k` описываются **неявно** с помощью ключа `TOKENS=`. Например, если в файле myfile.txt были записаны следующие три строки:

```
AAA BBBB CCCC,GGGG DDDD
EEEEE,JJJJ KKKK
;TTTT LLLL MMMMM
```

то в результате выполнения пакетного файла parser.bat на экран выведется следующее:

```
BBBB CCCC GGGG DDDD
JJJJ KKKK
```

Замечание. Ключ `TOKENS=` позволяет извлечь из одной строки файла до 26 подстрок, поэтому запрещено использовать имена переменных, начинающиеся не с букв английского алфавита (a–z). Следует помнить, что имена переменных `FOR` являются **глобальными**, поэтому одновременно не может быть активно более 26 переменных.

Команда `FOR /F` также позволяет обработать отдельную строку. Для этого следует ввести нужную строку в кавычках вместо набора имен файлов в скобках. Строка будет обработана так, как будто она взята из файла. Например, файл следующего содержания:

```
@ECHO OFF
FOR /F "EOL=; TOKENS=2,3* DELIMS=, " %%i IN ("AA CC BB,GG DD")
DO @ECHO %%i %%j %%k
```
при своем выполнении напечатает
```
CC BB GG DD
```
Вместо явного задания строки для разбора можно пользоваться переменными среды, например:
```
@ECHO OFF
SET M=AAA БББББ ВВВВ,ГГГГГ ДДДД
FOR /F "EOL=; TOKENS=2,3* DELIMS=,
  " %%i IN ("%M%") DO @ECHO %%i %%j %%k
```

Наконец, команда `FOR /F` позволяет обработать **строку вывода другой команды**. Для этого следует вместо набора имен файлов в скобках ввести строку вызова команды в апострофах (не в кавычках!). Строка передается для выполнения интерпретатору команд cmd.exe, а вывод этой команды записывается в память и обрабатывается так, как будто строка вывода взята из файла. Например, следующий командный файл:
```
@ECHO OFF
CLS
ECHO Имена переменных среды:
ECHO.
FOR /F "DELIMS==" %%i IN ('SET') DO ECHO %%i
```
выведет перечень имен всех переменных среды, определенных в настоящее время в системе.

В цикле `FOR` допускается применение тех же синтаксических конструкций (операторов), что и для заменяемых параметров – (табл 4).

Таблица 4.

Операторы для переменных команды FOR

Операторы	**Описание**
`%~Fi`	Переменная `%i` расширяется до полного имени файла
`%~Di`	Из переменной `%i` выделяется только имя диска
`%~Pi`	Из переменной `%i` выделяется только путь к файлу
`%~Ni`	Из переменной `%i` выделяется только имя файла
`%~Xi`	Из переменной `%i` выделяется расширение имени файла
`%~Si`	Значение операторов `N` и `X` для переменной `%i` изменяется так, что они работают с кратким именем файла
`%~Zi`	Определяется длина (размер) файла с указанным именем

Замечание. Если планируется использовать расширения подстановки значений в команде FOR, то следует внимательно подбирать имена переменных, чтобы они не пересекались с обозначениями формата.

Например, если мы находимся в каталоге C:\Program Files\Far и запустим командный файл следующего содержания:

```
@ECHO OFF
CLS
FOR %%i IN (*.txt) DO ECHO %%~Fi
```

то на экран выведутся полные имена всех файлов с расширением txt:

```
C:\Program Files\Far\Contacts.txt
C:\Program Files\Far\ReadMe.txt
C:\Program Files\Far\register.txt
C:\Program Files\Far\WhatsNew.txt
```

Вычисление суммарной длины всех файлов в заданном подкаталоге

```
@ECHO OFF
SETLOCAL ENABLEDELAYEDEXPANSION
Set /a Size = 0
For %%I in (%1\*.*) do set /a Size= Size + %%~zI
Echo %Size%
```

1.1.13.6 Циклы и связывание времени выполнения для переменных

Как и в рассмотренном выше примере с составными выражениями, при обработке переменных среды внутри цикла могут возникать труднообъяснимые ошибки, связанные с ранними связыванием переменных. Рассмотрим пример. Пусть имеется командный файл следующего содержания:

```
SET a=
FOR %%i IN (Раз,Два,Три) DO SET a=%a%%%i
ECHO a=%a%
```

В результате его выполнения на экран будет выведена строка `"a=Три"`, то есть фактически команда

```
FOR %%i IN (Раз,Два,Три) DO SET a=%a%%%i
```

равносильна команде

```
FOR %%i IN (Раз,Два,Три) DO SET a=%%i
```

Для исправления ситуации нужно, как и в случае с составными выражениями, вместо знаков процента (%) использовать восклицательные знаки и предварительно включить режим связывания времени выполнения командой SETLOCAL ENABLEDELAYEDEXPANSION. Таким образом, наш пример следует переписать следующим образом:

```
SETLOCAL ENABLEDELAYEDEXPANSION
SET a=
FOR %%i IN (One,Two,Three) DO SET a=!a!%%i
ECHO a=%a%
```

В этом случае на экран будет выведена строка `"a=OneTwoThree"`.

1.1.13.7 Команда Findstr и ее использование в цикле

Назначение команды - поиск строк в текстовых файлах.

FINDSTR [/B] [/E] [/L] [/R] [/S] [/I] [/X] [/V] [/N] [/M] [/O] [/P] [/F:файл]
[/C:строка] [/G:файл] [/D:список_папок] [/A:цвета] [/OFF[LINE]]
строки [[диск:][путь]имя_файла[...]]

/L-Поиск строк дословно.
/R-Поиск строк как регулярных выражений.
/S-Поиск файлов в текущей папке и всех ее подпапках.
/I-Определяет, что поиск будет вестись без учета регистра.
/X-Печатает строки, которые совпадают точно.
/V-Печатает строки, не содержащие совпадений с искомыми.
/N-Печатает номер строки, в которой найдено совпадение, и ее содержимое.
/M-Печатает только имя файла, в которой найдено совпадение.
/O-Печатает найденный строки через пустую строку.
/P-Пропускает строки, содержащие непечатаемые символы.
/F:файл-Читает список файлов из заданного файла (/ для консоли).
/C:строка-Использует заданную строку как искомую фразу поиска.
/D:список_папок-Поиск в списке папок (разделяются точкой с запятой).
строка Искомый текст.
[диск:][путь]имя_файла - задает имя файла или файлов.

Использовать пробелы для разделения нескольких искомых строк, если аргумент не имеет префикса /C. Например, 'FINDSTR "Привет мир" a.b' ищет "Привет" или "мир" в файле a.b, а команда 'FINDSTR /C:"Привет мир" a.b' ищет строку
"Привет мир" в файле a.b.

Краткая сводка по синтаксису регулярных выражений:

. Любой символ.
* Повтор: ноль или более вхождений предыдущего символа или класса
^ Позиция в строке: начало строки
$ Позиция в строке: конец строки
[класс] Класс символов: любой единичный символ из множества
[^класс] Обратный класс символов: любой единичный символ из дополнения
[x-y] Диапазон: любые символы из указанного диапазона
\x Служебный символ: символьное обозначение служебного символа x
\<xyz Позиция в слове: в начале слова
xyz\> Позиция в слове: в конце слова

Пример командного файла для поиска в файле num.txt по образцу строк, в которых присутствует хотя бы одна двоичная цифра.

```
@echo off
set /a kol=0
for /f %%b in ('findstr /rc:"[0-1]" num.txt') do set /a kol=kol+1
echo %kol%
```

2 МЕТОДИКА ВЫПОЛНЕНИЯ

1. Неформально ознакомиться с теоретическими сведениями.
2. Для подготовки текстов командных файлов рекомендуется использовать блокнот (Notepad). При этом следует избегать использования в выводимых на экран результатах работы командного файла букв русского алфавита.
3. Разработать и выполнить командные файлы (КФ), выполняющие следующие функции:
4. Вывод на экран имен всех файлов с указанным расширением, находящихся в каталоге, имя которого задается при запуске командного файла первым параметром. Расширение файлов задается вторым параметром.
5. Среди введенных с клавиатуры целых чисел (использовать SET /P) найти наибольшее и наименьшее. Признак конца ввода – знак -.
6. В заданном каталоге и его подкаталогах найти общее количество подкаталогов. На экран вывести только требуемый результат.
7. В каталогах, имена которых заданы первым и вторым параметрами командного файла, найти и вывести на экран имена файлов (расширения могут быть любые), присутствующие как в первом, так и во втором каталоге. Следует использовать только один оператор FOR.
8. Вычисление и вывод на экран значения факториала целого числа, задаваемого при запуске КФ. Предусмотреть проверку заданного значения и при задании отрицательного значения или значения, превышающего максимально возможную величину, выводить соответствующие сообщения. Для проверки правильности вычислений использовать калькулятор.
9. Разработать и выполнить КФ в соответствии с табл. 5 (индивидуальные задания для студентов).

Таблица 5.

Индивидуальные задания для бригад и студентов

Номер бригады	Действия, выполняемые КФ
1	1.Подсчет количества целых чисел в текстовом файле. Считать, что слова в файле записаны в формате ОДНО СЛОВО В СТРОКЕ. Слово – это целое число (состоящее из десятичных цифр) или последовательность букв латинского алфавита (начинающаяся с буквы). Имя файла задается первым параметром КФ. 2. Вывод на экран списка файлов, хранящихся в указанном первым параметром каталоге и созданных в первом полугодии (месяцы 1-6) года, указанного вторым параметром КФ.
2	В каталоге, указанном первым параметром КФ, (и его подкаталогах) найти файл наибольшего размера с расширением, указанным вторым параметром КФ. В каталоге, указанном первым параметром КФ, (и его подкаталогах) найти ТРИ файла самого большого размера. Вывести имена файлов, их размеры и даты создания
3	1. Разбиение текстового файла, имя которого задано первым параметром КФ, на три файла с именами 1.txt, 2.txt и 3.txt. Количество строк в каждом из этих файлов задано вторым, третьим и четвертым параметрами КФ. Проверить наличие указанного исходного файла и вывести сообщение о его отсутствии, проверить наличие остальных параметров и их значения на допустимость 2. В каталоге, указанном первым параметром КФ, (и его подкаталогах) найти суммарный объем файлов, имеющих расширение, указанное вторым параметром КФ.
4	1. Удаление из каталога, заданного первым параметром, файлов, которые присутствуют и в каталоге, указанным вторым параметром. Предусмотреть запрос пользователю на подтверждение удаления. 2. В каталоге, указанном первым параметром КФ, и его подкаталогах, найти файлы, созданные во второй половине рабочего дня (после 14 часов) и скопировать их в отдельный подкаталог.
5	Нахождение суммарного объема файлов с атрибутом system, хранящихся в каталоге, имя которого задано первым параметром КФ. Проверить наличие файла Numb.txt в каталоге, указанном первым параметром КФ. Прочитать целые числа из файла, найти среди них простые и вывести результаты на экран. Считать, что все числа не превышают значения 2500.
6	1. Поиск на диске C: (или любом доступном диске) файла с

Номер бригады	Действия, выполняемые КФ
	заданным именем. Если файл не найден – вывод сообщения. Если файл найден – открыть его для редактирования. 2. Проверка наличия на диске в каталоге, указанном первым параметром КФ, файла FNames.txt, содержащего список имен файлов и подкаталогов. Если он есть – проверка наличия перечисленных в списке файлов и вывод имен отсутствующих. Если файла FNames нет, создание его и запись имен файлов и подкаталогов.
7	1. Вывод списка DLL (хранящихся на доступном диске), созданных до 12.2015 размером до 12000 байтов. 2. Проверка наличия на диске в каталоге, указанном первым параметром КФ, файла Numbers.txt, содержащего 2 столбца целых чисел, столбцы располагаются с позиций 2 и 20 и отделены пробелами. Если файла нет – вывод сообщения. Если файл есть, создать новый файл, содержащий три столбца, в третий поместить сумму чисел из двух первых столбцов.
8	1. Просмотр содержимого каталога, указанного первым параметром КФ. Необходимо: 1. создать подкаталоги с именами EXE, TXT, CMD, DOC и OTHER. 2. В каждый подкаталог скопировать файлы с соответствующими расширениями. 3. Пустые подкаталоги удалить. 2. В каталоге, указанном первым параметром КФ, (и его подкаталогах) найти файлы наибольшего и наименьшего размеров. Вывести имена файлов, их размеры и даты создания.
9	1. Проверка наличия трех текстовых файлов на диске и объединения их в один файл. 2. Подсчет количества вещественных чисел и целых чисел в текстовом файле. Вещественные и целые числа подсчитать отдельно. Считать, что слова в файле записаны в формате ОДНО СЛОВО В СТРОКЕ. Слово – это целое число (состоящее из десятичных цифр) или последовательность букв латинского алфавита (начинающаяся с буквы) или последовательность десятичных цифр с точкой (.) внутри строки. Имя файла задается первым параметром КФ.
10	1. Подсчет количества слов в текстовом файле, содержащем целые числа и слова. Считать, что слова в файле записаны в формате ОДНО СЛОВО В СТРОКЕ. Число – это целое число (состоящее из десятичных цифр). Слово - последовательность букв латинского алфавита (начинающаяся с буквы). Имя файла задается первым параметром КФ.

Номер бригады	Действия, выполняемые КФ
	2. Просмотр содержимого каталога, указанного первым параметром КФ. Необходимо: 1. создать подкаталоги с именами 1, 2, ..., 12. 2. В каждый подкаталог скопировать файлы, созданные в соответствующие месяцы. 3. Пустые подкаталоги удалить.
11	1. Подсчет количества строк в текстовом файле, имя которого задано первым параметром КФ. Проверить наличие указанного файла и вывести сообщение о его отсутствии. 2. С помощью команды DIR вывести на экран имена файлов, находящихся в каталоге, имя которого задано первым параметром КФ. Второй и остальные параметры задают расширения файлов, имена которых выводить не следует. Рекомендуется с помощью ATRIB присвоить некоторым файлам атрибут СКРЫТЫЙ – такие файлы DIR не показывает.
12	1.Поиск текстового файла по его содержимому. Считать, что слова в текстовых файлах записаны в формате ОДНО СЛОВО В СТРОКЕ. Искомое слово задается первым параметром КФ.
13	1. Вывод на экран аргументов, с которыми КФ был запущен. Число аргументов от 4 до 11. При неверном числе аргументов ничего не выполнять, сообщить об ошибке. 2. Поиск и вывод на экран минимального и максимального значения аргумента КФ. Предполагается, что все аргументы КФ – целые положительные числа.

Примечание. Для решения задач 1, 9 и 10 рекомендуется использовать команду Findstr

3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Вывод сообщений и дублирование команд.
2. Использование параметров командной строки.
3. Переменные среды, получение и изменение их значений.
4. Операции со строковыми и числовыми переменными.
5. Проверка существования заданного файла и наличия переменной среды.
6. Выполнение заданной команды для всех элементов указанного множества.
7. Выполнение заданной команды для всех подходящих имен файлов.
8. Выполнение заданной команды для всех подходящих имен каталогов.

9. Выполнение заданной команды для определенного каталога, а также всех его подкаталогов.
10. Получение последовательности чисел с заданными началом, концом и шагом приращения.
11. Чтение и обработка строк из текстового файла.
12. Команда Findstr. Назначение. Ключи. Использование регулярных выражений в команде. Задание и использование класса цифр и класса букв через диапазон.
13. Операторы перехода и вызова.
14. Какое минимальное количество строк (включая @echo off) должен иметь командный файл, выводящий на экран минимальное значения двух числовых аргументов?
15. Какое минимальное количество строк (включая @echo off) должен иметь командный файл, выводящий на экран минимальное значения трех числовых аргументов?