Основы работы в интерпретаторе команд MS-DOS

ИНТЕРФЕЙС КОМАНДНОЙ СТРОКИ MS-DOS

Операционная система MS-DOS исторически не обладает графическим интерфейсом пользователя, в современном понимании этого термина. Доминирующее место в процессе взаимодействия с пользователем занимает текстовый режим, при помощи которого пользователь вводит команды и получает текстовые сообщения от операционной системы о результатах их исполнения. Аббревиатура DOS (ДОС) расшифровывается как Disk Operation System (Дисковая Операционная Система). Соответственно, основными операциями системы являются действия над файловой системой и ее компонентами (файлами, каталогами, дисками). Команды операционной системы MS-DOS разделяются на четыре категории: интерфейсные, основные (связанные с операциями над файлами и каталогами), потоковые, пакетные. Задачи, которые решаются командами каждой из групп, могут выполняться сообща благодаря потоковым командам, то есть несколько команд могут быть объединены специальными операторами-связками в форме единой конструкции, которая интерпретируется и выполняется операционной системой MS-DOS в определенном порядке.

Начнем с рассмотрения интерфейсной группы команд. Первая из них позволяет определить системное окружение, то есть параметры даты и времени, относительно которых выполняются операции над файлами. Команда **`DATE`** выполняет установку текущей даты, **`TIME`** – устанавливает текущее время на компьютере. Эти команды, так же как и остальные команды в операционной системе MS-DOS, поддерживают встроенную справку по использованию. Для вызова дополнительной информации необходимо после названия команды через пробел указать символы **`/?`**. Например, **`DATE /?`**. Знак **`/`** очень часто используется в комбинациях с другими буквами и символами – называется такое сочетание дополнительным параметром или ключом, который указывается через пробел. Подобных параметров у команды может быть несколько, точное их количество зависит от конкретной инструкции. Любые дополнительные ключи отделяются друг от друга, как и от команды, знаком пробела. При запуске команды **`DATE`** или **`TIME`** и нажатии клавиши [Enter] от пользователя потребуется указать с клавиатуры дату или время в соответствующем формате – для даты **`ДД.ММ.ГГ`** (ДД – дата, ММ – месяц, ГГ – две последние цифры года), для времени **`ЧЧ:ММ:СС`** (ЧЧ – часы, ММ – минуты, СС – секунды).

Системное окружение определяется не только датой и временем, но и таким понятием как *текущая директория*. Важность этого термина определяется его назначением. Все операции, выполняемые над файлами и каталогами в операционной системе MS-DOS, осуществляются в рамках одного каталога, который является текущим (в отдельных случаях, при указании точного расположения, текущая директория операционной системой не берется во внимание).

При введении команд пользователя в командной строке, слева от текстового курсора расположена подсказка, содержащая полный путь к текущему каталогу. Изменить значение текущего каталога можно с помощью команды **CD ПУТЬ**, указав в качестве пути *полный* или *относительный путь* к требуемому каталогу. Полный путь к директории определяется указанием буквы логического диска, промежуточных каталогов, находящихся перед целевой директорией, и наименованием необходимой папки. Например, **C:\DIR1\DIR2\CURDIR**, обозначает папку CURDIR, расположенную на диске C: в подкаталоге DIR2 каталога DIR1. Полный путь к файлу формируется аналогичным образом. Например, **C:\DIR1\DIR2\FILE.TXT**, обозначает расположение текстового файла FILE.TXT в подкаталоге DIR2 каталога DIR1 на диске C:. Установив в качестве текущей директории какой-либо каталог, пользователь может обращаться к нему или к каталогу, расположенному на уровень выше (предыдущая папка) с применением специального краткого обозначения. Полный путь к текущей директории указывается с помощью знака точка «**.**». Полный путь к предыдущей директории устанавливается через запись двух точек «**..**». Подобная форма может использоваться наряду с записью полного пути, а также совместно, если есть необходимость указать подобным образом расположение папки или файла относительно текущей директории. Относительный путь – это достаточно гибкое средство указания расположения папок и файлов. Зачастую место размещения пользовательских файлов определяется лишь наименованием каталога, в котором они размещены. А вот сам путь размещения этой папки индивидуален для каждого случая. Поэтому, если пользовательские файлы взаимосвязаны внутренними вызовами или гиперссылками между собой, возникает необходимость определения системного окружения, в котором эти файлы будут использоваться – текущая директория является той самой отправной точкой, относительно которой строятся взаимосвязи между файлами пользователя. Необходимо отметить, что текущая директория у каждого логического диска своя, и операционная система следит за переходами пользователя от одного диска к другому, восстанавливая полный путь в качестве значения. Переключение диска в командной строке операционной системы MS-DOS выполняется непосредственно через указание наименования буквы диска и знака двоеточия «**:**». При этом в командной строке подсказка (по-другому она называется строкой приглашения MS-DOS) изменит свое значение.

Установить содержимое строки приглашения можно при помощи команды **PROMPT**, для которой указывается шаблон подсказки из переданных в качестве аргумента текстовых символов и/или знаков-параметров, именуемых ESC-последовательностью. Кроме информации о текущей директории можно указать системное время или дату, а также какие-либо дополнительные символы, набранные с клавиатуры. Вместе с тем можно использовать и так называемые *переменные окружения*, которые хранят дополнительную полезную информацию о среде выполнения операционной системы MS-DOS. Например, команда с использованием аргументов **PROMPT**

($D#$THH$H)$P$G** изменит вид строки приглашения MS-DOS на **(01.02.2010#12:00)C:\TEMP>**. А команда **PROMPT (%COMPUTERNAME%#%USERNAME%)PG приведет строку приглашения к виду **(TERMINAL_PC1#STUDENT)C:\TEMP>**.

Для определения и установки значений переменным окружения используется команда **SET**. По сути, имена для переменных окружения определяются использующими их программами, то есть каждая в отдельности такая переменная используется либо отдельной программой, либо группой программ, и имеет строго назначенное имя, иначе трудно будет отделить нужный параметр среды выполнения от прочих и передать необходимое значение другим программам. Например, такие переменные окружения как TEMP или TMP определяют путь к каталогу для временных файлов. Обычно значением таких переменных окружения выступают выражения вида **C:\TEMP**. При обращении к переменным окружения их наименования заключаются в знак процента «**%**». То есть, команда **CD %TEMP%** установит в качестве текущей директории каталог **C:\TEMP**. Чтобы ознакомиться со всеми предопределенными при загрузке операционной системы переменными окружения достаточно набрать команду **SET** без параметров. При установлении значений переменным окружения используется, в качестве аргумента команды **SET**, знак равенства, слева от которого указывается наименование переменной, а справа ее значение. При этом значение, в отличие от наименования, может содержать пробелы. Вообще пробелами разделяют аргументы, переданные команде или программе, между собой. Такое отступление допустимо нескольким командам MS-DOS (cd, echo, set), в числе которых находится и команда **SET**. При этом дополнительные аргументы лучше всего указывать до определения значений переменной окружения, то есть слева от присвоения значения и справа от команды. Например, **SET /P TEMP=Input new path:** позволяет определить новое значение переменной окружения **%TEMP%** посредством запроса этого значения с клавиатуры. Для очистки значения переменной окружения и последующего ее удаления используется команда **SET TEMP=**.

ОСНОВНЫЕ ОПЕРАЦИИ В КОМАНДНОЙ СТРОКЕ MS-DOS

Как упоминалось ранее, операционная система MS-DOS и, особенно, набор команд изначально проектировались для обеспечения интерфейса между программной составляющей рабочего места пользователя и его аппаратным обеспечением. Любая из программ, до момента ее загрузки и исполнения, расположена на каком-либо носителе информации, представляющем на логическом уровне файловую систему. Поэтому основное внимание в интерпретаторе команд MS-DOS было сконцентрировано на использовании файловой системы.

Для вывода содержимого файла используется команда **TYPE**, в качестве параметра которой указывается путь к файлу с текстовой информацией. Обычно, речь идет о текстовых файлах с расширением **TXT**.

Вывод содержимого каталога осуществляется по команде **DIR**. Если не указывать аргументы, то будет выведен список файлов и каталогов текущей директории. С этой командой используются дополнительные аргументы **/P**, а также **/W**. Первый параметр позволяет пролистывать список содержимого директории в поэкранном режиме с нажатием клавиш подтверждения для продолжения. Второй параметр уплотняет выводимый список в форме нескольких колонок, скрывая дополнительную информацию о файлах и каталогах. Порядок следования параметров, как и во всех командах операционной системы MS-DOS не имеет значения.

Необходимо отметить, что использование команды **DIR**, как и других команд, рассматриваемых далее, очень тесно связано с таким понятием как *маска*. Под этим термином понимают обозначение некоторого шаблона, в котором указываются расположения известных пользователю символов в определенных позициях наименования файла или каталога, а также некоторых символов, которые являются искомыми при выполнении той или иной операции. При этом неизвестные символы отмечаются в маске при помощи специальных знаков звездочка «*****» и вопрос «**?**». Использование данных специальных знаков определяет не только позицию искомых знаков, но и их количество. Так, например, под знаком вопрос понимается ровно один неизвестный символ. Если есть необходимость указать в шаблоне три неизвестных символа, следующих друг за другом, знак вопроса вносят в маску три раза подряд. Для обозначения любого возможного количества знаков используют символ звездочка. Маска, обычно, применяется для отсечения некоторого подмножества файлов из выводимого или обрабатываемого командой списка. Например, маска **T*.?X?** при использовании в команде **DIR** будет означать необходимость вывести список содержимого текущей директории, а именно файлы, начинающиеся с символа T и имеющие в расширении символ X на второй позиции. Это могут быть такие файлы как **text.txt**, **tools.ext**, **tree.exe**, **two.fx**. Последнее наименование файла из примера попало в список обработки команды по причине того, что специальные знаки в конце маски (то есть в окончании наименования или расширения файла) приобретают смысл «не более», то есть может быть и отсутствие каких-либо символов в этой позиции. Четыре наименования файлов, рассмотренные как результат применения маски, может быть получен и с помощью маски **T????.?X?**.

Копирование файлов и каталогов выполняется при помощи команды **COPY**. Первый параметр указывает путь к файлу источнику, второй параметр указывает путь к каталогу приемнику или путь к новому файлу, если есть необходимость при копировании выполнить переименование. В параметрах, указывающих путь расположения файлов, можно использовать маску. Например, команда **COPY *.INI %TEMP%*.TXT** выполнит копирование всех конфигурационных фалов, находящихся в текущей директории, во временную директорию, определяемую переменной окружения **%TEMP%**, с изменением у конечных файлов расширения с **INI** на **TXT**. Использование

маски в команде **COPY** позволяет выполнить слияние файлов в один результирующий. Вообще склеивание файлов выполняется перечислением в качестве первого параметра команды тех файлов, которые необходимо объединить, при этом между наименованиями файлов записывается знак плюс «+». Например, команда **COPY text?.txt+file.txt result.txt** выполнит склеивание файлов **text.txt**, **text0.txt**, **text1.txt**, **text2.txt**, **texta.txt** и **file.txt**.

При задании пути расположения файла, а также в его наименовании, могут быть использованы имена, которые не подходят под формат 8.3 (до 8 знаков под задание имени и до 3 знаков для расширения). Причиной этому служит то, что операционная система MS-DOS ушла в лета, и о ней лишь напоминает интерпретатор командной строки MS-DOS, являющийся функциональной частью операционной системы Windows, в которой соглашение о длине имен для файлов позволяет использовать очень длинные наименования каталогов и файлов, в общей сложности, не превышающие по количеству 255 знаков, в том числе и пробелы. В этом случае, путь расположения или наименование файлов или каталогов, в том числе и в случае использования масок, заключается в двойные текстовые кавычки. Например, команда **COPY "C:\Мои документы\Опер.сист., среды и оболочки\Лабораторные работы\Lab1.doc" "D:\Отчеты на завтра\"**.

Переименование файлов также можно выполнить при помощи команды **REN**. Так, используя маску в качестве аргументов команды, можно выполнить изменение расширений у файлов в текущей директории – **REN *.INI *.TXT**.

Для удаления файлов используется команда **DEL**, которая в качестве параметров может принимать не только наименование конкретного удаляемого файла, но и маску.

Создание каталога выполняется при помощи команды **MD**, имя создаваемой папки, а также, при необходимости, полный или относительный путь ее расположения, задается через аргумент.

Удаление пустого каталога, не содержащего ни файлов, ни других каталогов, выполняет команда **RD**. Для очистки каталога необходимо позаботится об удалении всех содержащихся в нем элементов. Однако, при использовании дополнительного параметра **/S**, команда **RD** выполнит очистку структуры удаляемого каталога самостоятельно.

Отдельно нужно сказать о команде **ATTRIB**, которая используется для назначения атрибутов файлам и каталогам, присваивание кроме имен и расширений дополнительных свойств которым позволяет обособить отдельную группу, независимо от расположения на диске. Отмеченные таким образом файлы и каталоги будут находиться под особым вниманием операционной системы MS-DOS, и, в случае некорректных манипуляций с ними, пользователь будет об этом предупрежден, либо ограничен в своих полномочиях до тех пор, пока атрибут не будет изменен.

РАБОТА С ПОТОКАМИ В ИНТЕРПРЕТАТОРЕ КОМАНД MS-DOS

Выполняя действия в операционной системе MS-DOS, пользователь использует для их инициирования командную строку, в которой посредством клавиатуры вносит наименования команд, программ, файлов и каталогов. В результате обработки запросов, весь получаемый информационный поток в виде текста выводится на экран монитора. Использование стандартных устройств ввода-вывода является тем минимумом, который должна обеспечить операционная система MS-DOS своих пользователей. Однако, вместо экрана дисплея, пользователю, возможно, понадобится использование текстового файла, или принтера. В этом случае операционная система MS-DOS предлагает ряд устройств. Рассмотрим их по порядку.

CON – устройство, соединяющее поток при вводе с клавиатурой, а при выводе с монитором.

LPT1-LPT3 – устройства, присоединяемые к параллельным портам (обычно старые модели принтеров и сканеров).

COM1-COM3 – устройства, присоединяемые к последовательным портам (обычно старые модели модемов, манипуляторов «мышь»).

AUX – дополнительное устройство, присоединяемое к асинхронному последовательному порту.

PRN – устройство, ассоциированное в операционной системе MS-DOS с принтером.

NUL – устройство, которое отсутствует, а предназначаемые ему операции игнорируются операционной системой MS-DOS.

Ввиду строго определенных наименований для устройств ввода-вывода, рассмотренные сочетания символов считаются зарезервированными в операционной системе MS-DOS. Поэтому, использование их в качестве наименований для файлов запрещается. Однако использовать эти же обозначения устройств ввода-вывода допускается в качестве расширений для файлов.

Указание в качестве устройства ввода или вывода конкретного устройства или файла называется *переопределением* или *перенаправлением потока*. Для выполнения этой операции используются специальные знаки меньше «<», больше «>», удвоенный знак больше «>>» и вертикальная палочка «|», обособленные пробелами с обеих сторон. Данные символы располагают справа от полностью записанной команды со всеми необходимыми аргументами. После знака операции перенаправления потока указывается устройство, которое будет использоваться вместо стандартного потока ввода-вывода. При этом необходимо отметить, что не все команды используют потоки ввода или вывода. Указав новый поток ввода или вывода для команды, которая его не использует, не стоит ожидать, что операционная система организует обработку введенной Вами информации или выведет какое-либо сообщение.

Подробнее о каждой из операций перенаправления потока. Знак меньше используется для указания устройства ввода, которым может выступать файл, в

случае указания его наименования или пути расположения вместо имени устройства. Знак больше используется для указания устройства вывода информации, и, в случае, если в качестве устройства задано наименование файла или путь, вся информация будет выводиться именно туда. При этом повторное выполнение команды с выводом на указанное устройство стирает ранее находящуюся там информацию. Для того чтобы такой ситуации не происходило, используется двойной знак больше, который позволяет дописывать информацию, выдаваемую командой.

Рассмотрим пример с уже известными Вам командами. Операция **DIR /W *.TXT > DIRINFO.TXT** выдаст перечень текстовых файлов текущей директории, записав информацию в текстовый файл DIRINFO.TXT в этом же каталоге. Причем полученный список файлов будет помимо прочего содержать и наименование созданного в результате операции файла DIRINFO.TXT. Объясняется это тем, что операционная система MS-DOS выполняет связывание потоков ввода вывода заблаговременно, чтобы команда при выполнении не оказалась в состоянии ожидания на предоставления аппаратного или программного ресурса.

Стоит отметить, что команды, предполагающие наличие обязательных аргументов с указанием источника и/или приемника информационных потоков могут использовать непосредственно наименования устройств ввода-вывода. Так, команда **COPY** в качестве потока вывода может использовать устройство **CON**, и тогда запись **COPY TEXT.TXT CON** будет аналогична вызову команды **TYPE TEXT.TXT**. И обратная ситуация, которая может случиться при необходимости набрать текстовый файл с клавиатуры и сохранить его. Существует два пути решения этой задачи. Первый способ – это команда **COPY CON TEXT.TXT**. Второй способ – команда **TYPE CON > TEXT.TXT**. При этом второй подход более гибок в отношении пополнения текстового файла, в отличие от первого, где необходимо для этих целей проделать ряд предварительных, а потом последующих операций. В режиме работы с клавиатуры через устройство **CON** завершение ввода осуществляется нажатием клавиш [Ctrl]+[Z] и [Enter] или [F6] и [Enter]. Для отмены действий ввода используйте комбинацию клавиш [Ctrl]+[C] или [Ctrl]+[Break].

Пример переназначения потока ввода обычно сводится к формированию отдельного файла ответов для команды, поддерживающей диалоговый способ работы, когда на каждом этапе выполнения операции запрашивается подтверждение пользователя. При этом перечень вопросов, на которые предстоит ответить пользователю, заранее известен. Например, команда форматирования диска может выполняться в подобном режиме – **FORMAT A: < ANSWER.TXT**.

Рассмотрим другой пример. Имеется некоторый конфигурационный текстовый файл TEMP.INI, содержащий полный путь к каталогу для временных файлов. Для того чтобы определить переменную окружения и задать ей значение, которое хранится в файле, используем уже знакомую Вам команду **SET**. Выглядеть целиком команда будет так **SET /P TEMP= < TEMP.INI**.

Для вывода на экран монитора и проверки значения переменной окружения после выполнения команды нужно выполнить операцию **SET TEMP** или **ECHO TEMP=%TEMP%**. Последняя команда будет рассмотрена в следующем подразделе. Обе команды произведут вывод на экран значения переменной окружения TEMP.

Использование знака вертикальной палочки связано с таким режимом выполнения команд как *пакетная обработка*. В этом режиме информация передается из потока вывода одной команды или программы в поток ввода другой. Такая цепочка может содержать две, и более операции, которые постепенно преобразуют исходную информацию согласно используемым командам. В связи с этим необходимо рассмотреть специальные *команды-фильтры*, которые работают с обоими потоками и используются для промежуточной обработки данных.

Команда **MORE** используется для поэкранного вывода содержимого текстового файла или информационного потока вывода другой команды. Пример вызова – **MORE TEXT.TXT** или **TYPE TEXT.TXT | MORE**. При этом хочется отметить аналогичную, на первый взгляд, команду **COPY TEXT.TXT CON | MORE**, которая выполнит вывод содержимого текстового файла, но не в поэкранном режиме, а сразу пролистает его до конца. Причина в особенности команды, которая на уровне программной реализации изолированно работает с устройствами источника и приемника информации, а поток вывода касается лишь сервисных сообщений о результатах операции копирования, которые можно вывести в отдельный текстовый файл, например, так – **COPY TEXT.TXT CON > RESULT.INF**, или не выводить вовсе – **COPY TEXT.TXT CON > NUL**.

Команда **SORT** выполняет упорядочивание строк. Может использоваться как с указанием устройства или файла источника текстовых строк, так и с указанием результирующего файла или устройства вывода информации. При размещении в цепочке пакетной обработки может комбинировать перенаправление потоков и аргументы. Например, **SORT TEXT.TXT | MORE** или **TYPE CON | SORT > SORTED.TXT**.

Выборку строк, удовлетворяющих условию, выполняет команда-фильтр **FIND**. Под условием понимается текстовое выражение параметра, которое является частью строки текста, проходящего фильтрацию одна за другой. При этом использование аргумента **/V** позволяет включить обратный режим функционирования, исключающий из выборки строки, содержащие указанный в качестве параметра текст. Если условие фильтрации содержит несколько значений, то их необходимо реализовывать каждой новой командой **FIND**, выполняя их соединение в цепочку пакетной обработки с использованием знака вертикальной палочки. Например, команда **TYPE TEXT.TXT | FIND /V "Scarlett" | FIND "Hanson" | SORT > SELECTED.TXT** выполнит отбор в списке тех строк текстового файла TEXT.TXT, где встречается фамилия Hanson, за исключением имен Scarlett, после чего выполнит сортировку, а результат поместит в файле SELECTED.TXT.

ФОРМИРОВАНИЕ ПАКЕТНЫХ ФАЙЛОВ

Возможности, которые открываются перед пользователем при работе с потоками, не всегда решают задачи автоматизированной обработки в полном объеме. Во многих случаях объединение в одной строке нескольких команд нагромождает и усложняет понимание происходящих в командах процессов. Поэтому операционная система MS-DOS поддерживает обработку последовательности команд, которые объединены в одном текстовом файле. Расширение такого файла должно быть BAT. По умолчанию, при указании наименования файла без расширения, операционная система MS-DOS рассматривает пакетные файлы в первую очередь, потом, в случае отсутствия одноименного пакетного файла, выполняется попытка запуска файла с расширением COM, и, в последнюю очередь, рассматривается EXE файл. Если ни один из файлов с этими расширениями не найден, операционная система MS-DOS выдаст сообщение об отсутствии указанного файла. Если одноименные файлы с расширением BAT, COM и EXE присутствуют в текущем каталоге, то желательно указать при запуске полностью имя с расширением, чтобы исключить неоднозначнось. Как уже упоминалось, запуск пакетного файла или программы выполняется из текущей директории, но если файл с таким именем отсутствует, операционная система MS-DOS выполнит поиск этого файла в поисковых директориях, список которых определяется переменной окружения **%PATH%**.

При запуске пакетных файлов, также как и при запуске программ и команд, в операционной системе MS-DOS поддерживается передача параметров, определяющих режим их работы. В самом пакетном файле обращение к аргументам, переданным после указания наименования пакетного файла, осуществляется через указание порядкового номера аргумента. Так под номером 0 воспринимается собственно наименование пакетного файла, в том виде, в котором его указали при запуске в командной строке, под номером 1 – первый параметр и так далее. Перед порядковым номером параметра необходимо указывать знак процента «**%**». Например, пакетный файл RUN.BAT содержит строку следующего содержания **ECHO This File Name – %0**. При запуске этого пакетного файла, на экране мы получим сообщение **This File Name – RUN.BAT**. Для очистки экрана монитора от ранее выданных сообщений используется команда **CLS**.

Так как пакетный файл похож по структуре на некоторое подобие программы на языке программирования, то для решения задач автоматизации интерпретатор командной строки MS-DOS расширен дополнительными командами, цель которых организовать автоматизированный процесс выполнения задач пользователя. Как в любой программе, часть строк могут содержать комментарии, поясняющие смысл той или иной операции. Для комментирования используется команда **REM**, которая при интерпретации пакетного файла игнорируется.

Каждая команда, входящая в пакетный файл при исполнении дублируется в командной строке как есть. Для подавления вывода на экран монитора

исполняемых в пакетном файле команд можно использовать перед командой знак собака «**@**». Вообще режим дублирования в интерпретаторе команд MS-DOS регулирует операция **ECHO**, которая не только выводит сообщения на экран монитора, что является основной ее задачей, но и принимает особые аргументы – **ON** и **OFF**. **ECHO ON** включает режим дублирования команд, а **ECHO OFF** отключает его. Данную инструкцию лучше расположить в самой первой строке Вашего пакетного файла, чтобы на экране монитора появлялись только нужные Вам сообщения.

Любая строка в пакетном файле может быть обозначена особым маркером – *меткой*, который позволяет при необходимости пропустить определенные команды и приступить к обработке операции, расположенной после метки. Обозначается метка текстовыми символами, которые следуют неразрывно после знака двоеточие «**:**». Например, метка **:END**. Для перехода на определенную метку достаточно указать ее наименование в качестве параметра команды **GOTO**. Например, **GOTO END**. Необходимо отметить, что наименование метки может быть значением переменной окружения. Например, **GOTO %METKA%** или **GOTO METKA%HOMEP%**.

Вернемся к рассмотрению аргументов, переданных пакетному файлу. Зачастую параметры могут следовать в произвольном порядке, и четко ожидать под первым параметром определенного значения не стоит. Весь набор возможных значений можно проверить при помощи команды **IF**, которая в качестве первого аргумента принимает логическое условие, а в качестве второго операцию для выполнения в случае логической истины первого. Так, проверяя соответствие первого аргумента пакетного файла каждому из возможных значений, мы можем выполнить соответствующую операцию, а потом забыть про этот самый аргумент, перейдя к следующему параметру, и так далее, пока последний переданный параметр не окажется пустым. Команда **SHIFT** выполняет сдвиг параметров пакетного файла: первый аргумент перемещается на место нулевого, второй на место первого и так далее. В итоге, после нескольких сдвигов, параметры закончатся, и, обращаясь к первому аргументу, мы получим пустое значение.

```
:METKA
IF _%1==_/A GOTO METKA1
IF _%1==_/B GOTO METKA2
...
SHIFT
IF NOT _%1==_ GOTO METKA
```

В предлагаемом примере в операции сравнения используется дополнительный знак подчеркивания «_», который не изменяет логику происходящего, и необходим для корректной обработки логического выражения, так как для интерпретатора командной строки не существует значения "пусто".

Кроме проверки аргументов и переменных окружения на соответствие какому-либо значению, команда **IF** позволяет определять наличие по

указанному пути файла или каталога, для чего в выражении условия используется дополнение **EXIST**. Например, определить наличие файла TEXT.TXT в текущей директории можно с помощью команды **IF EXIST TEXT.TXT ECHO File found**. При этом совместно с выражением можно использовать приставку **NOT** для изменения логического смысла выражения.

Для организации циклической обработки некоторого множества, например, файлов, используется команда **FOR**. При этом каждый из экземпляров множества предстает в описании действия итерации цикла в виде переменной, имя которой определяет пользователь. Например, команда **FOR %i IN (*.TXT) DO TYPE %i | SORT > %i** выполнит раздельную сортировку всех текстовых файлов, расположенных в текущей директории, в отличие от команды **TYPE *.TXT | SORT > SORTED.TXT**, которая выполнит объединение всех текстовых файлов текущей директории в один, а после отсортирует его.

В следующем примере пакетный файл обрабатывает текстовые файлы, указанные в аргументах начиная со второго, построчно, обрамляя каждую текстовую строку в квадратные скобки. Результат работы записывается в текстовый файл, указанный в первом аргументе. Необходимо отметить, что при использовании команды **FOR** в пакетном файле, переменные, объявленные внутри цикла должны записываться с удвоенным знаком процента. Данное обстоятельство связано с тем, что интерпретатор команд MS-DOS должен различать обращения к локальным переменным цикла и аргументам, которые переданы в пакетный файл. Расширенный режим команды **FOR** с параметром **/F** включает дополнительные возможности обработки, в частности возможность построчной обработки информационного потока, выдаваемого при выполнении команды MS-DOS, которая указана в одинарных текстовых кавычках.

```
@ECHO OFF
IF _%1==_ GOTO END
IF _%2==_ GOTO END
:METKA
FOR /F "tokens=*" %%i IN ('type %2') DO ECHO [%%i] >> %1
SHIFT /2
IF NOT _%2==_ GOTO METKA
:END
```

При написании пакетного файла может понадобиться более развитая структура процедур, по причине того, что очень неудобно выделять в блоки часть пакетного файла и переходить к ним через метки. Отдельно процедуру можно оформить как самостоятельный пакетный файл, принимающий определенные аргументы, так как связность пакетных файлов при таком подходе должна соблюдаться (использование переменных окружения в этом смысле не оправдано, поскольку обычно их предназначение заключается в формировании параметризованного пространства ресурсов, как программных, так и аппаратных).

Для обращения из пакетного файла к другому пакетному файлу необходимо использовать команду **CALL**, для которой в качестве параметра передается наименование запускаемого пакетного файла (или программы), с указанием необходимых для его работы аргументов. По окончанию работы пакетный файл может передать числовой код завершения – это, так называемая, обратная связь на те параметры, которые были переданы при запуске. Выполняется это командой **EXIT /B 0**. По коду завершения вызвавший процедуру пакетный файл может провести анализ успешности операции. В качестве кода может быть представлено любое числовое значение, и оно не обязательно будет свидетельствовать об ошибке выполнения. Все зависит от реализации процедуры и того, что ожидает вызвавший ее пакетный файл. Если обратиться к пакетному файлу без вызова **CALL**, то анализ кода завершения будет невозможен, так как инициативу перехватит операционная система MS-DOS и вернет код, соответствующий ее ранжированию успешности выполнения программы.

Анализ кода завершения выполняется командой **IF** с использование специального условия **ERRORLEVEL 0**. Вообще, получить код ошибки можно, обратившись к переменной окружения **%ERRORLEVEL%**, в которой в качестве значения сохраняется последний код завершения, полученный при выполнении пакетного файла, при этом не важно каким образом был произведен вызов, с использованием команды **CALL** или без. При использовании команды **IF** с выражением **ERRORLEVEL** необходимо выстроить последовательность проверок от большего значения к меньшему значению, так как условие интерпретируется как код завершения "не больше проверяемого" числового значения.

Для организации диалога с пользователем используется команда **CHOICE**, которая выводит на экран монитора некоторое сообщение и ожидает в ответ на него выбор и ввод с клавиатуры одного из предложенных значений. В соответствии с введенным пользователем значением устанавливается код завершения команды, который необходимо обработать в пакетном файле с помощью блока команд **IF ERRORLEVEL**. При использовании команды **CHOICE** в качестве аргументов **/C:** передаются друг за другом варианты символов для ввода с клавиатуры, **/T:** параметры выбора по умолчанию и после запятой время в секундах на принятие пользователем самостоятельного решения, сообщение с вопросом пользователю. Код завершения выставляется в соответствии с порядковым номером символа, который выбрал пользователь из строки вариантов. Пример использования команды **CHOICE** представлен ниже. Первый вариант выбора не обрабатывается условием **IF**, так как предшествующий блок условий отсекает большие по коду завершения пункты.

```
CHOICE /C:ABCD /T:A,5 "What you'll choose"
IF ERRORLEVEL 4 GOTO PROCD
IF ERRORLEVEL 3 GOTO PROCC
IF ERRORLEVEL 2 GOTO PROCB
GOTO PROCA
```

Основы программирования на скриптовых языках

ТИПЫ ДАННЫХ И ИНТЕРФЕЙСНЫЕ ОБЪЕКТЫ

Пакетные файлы, рассмотренные в предыдущем разделе, расширяют функциональные возможности пользователя, работающего в операционной системе. Однако современные приложения и документы имеют более сложную структуру, нежели текстовый редактор "Блокнот" и документ с расширением TXT. По этой причине решение задач автоматизирования с помощью BAT-файлов оказывается не столь эффективным, сколько требует от этого и позволяет пользователю операционная система Windows. Речь в данном разделе пойдет об интерпретационных языках программирования, по-другому называемых *скриптовыми языками*. В состав операционной системы Windows входит платформа Windows Script Host, обеспечивающая интерпретацию команд, составляющих мини-программы или, по-другому, *скрипта* (*сценария*), с распространенных языков программирования, таких как Visual Basic и Java. Название у скриптовых языков, синтаксис которых базируется на языках программирования, соответственно VBScript и JScript. Скрипт-файл имеет отличное от BAT расширение – vbs и js соответственно. Для запуска скрипта используется программный модуль интерпретатора, который реализован в виде двух программ – **`cscript.exe`** для консольного выполнения (в черном текстовом окне, наподобие командного режима MS-DOS) и **`wscript.exe`** для графического выполнения (режим выполнения с использованием графических средств операционной системы Windows). По умолчанию, расширения скрипт-файлов зарегистрированы и ассоциированы (по двойному щелчку мыши) с запускающим приложением **`wscript.exe`**. Приступим к рассмотрению типов данных и базовых операций над элементами.

Первым в этом списке будет строковый (текстовый) тип данных. Набор знаков этого типа данных заключается в двойные текстовые кавычки, а также для JScript в одинарные текстовые кавычки. Как таковое определение переменных не производится, назначение типа выполняется в момент присваивания значения при помощи операции равенства «=». Поэтому в одной части программы одна и та же переменная может принимать значения разных типов. Вернемся к строковому типу. В интерпретационных языках VBScript и JScript имеются различия при внесении так называемых непечатаемых знаков, то есть символов, которые управляют текстовым курсором, а также знаков, которые используются в языке программирования для указания границ строки. К таким знакам относятся символы перевода на новую строку, символы перемещения каретки, удаления знаков, кавычки. Различия заключаются в обозначении этих знаков при формировании значения текстовой строки. В VBScript используются константы, значения которых соответствуют ASCII кодам. Использование констант выполняется при помощи операции *конкатенации* (соединения строк). В VBScript – это оператор «**&**», а в JScript – оператор «+». Однако непечатаемые символы в JScript записываются в виде *ESC-последовательностей*, которые размещаются внутри текстовых кавычек наряду с обыкновенными символами текста.

Константы VBScript		ESC-последовательности JScript	
vbCr	Возврат каретки	\b	Удаление символа слева
vbLf	Перевод строки	\n	Перевод строки
vbCrLf	vbCr & vbLf	\r	Возврат каретки
vbFormFeed	Перевод станицы	\t	Символ табуляции
vbNullChar	Символ с нулевым кодом	\'	Одинарная кавычка
vbNullString	Нулевая строка	\"	Двойная кавычка
vbTab	Символ табуляции	\\	Косая черта

Для удобочитаемости скрипта, возможно записывать текстовые значения в несколько строк, при этом каждая подстрока должна быть оформлена отдельным значением и объединена с другими подстроками операцией конкатенации. При этом в VBScript в конце продолжаемой строки записываемого оператора присваивания необходимо указывать знак подчеркивания «_». Завершением записи оператора в JScript является знак точка с запятой «**;**».

```
StringVB="┌──┬──────────────┬──────────┐"&vbCrLf&_
         "│N│     ФИО      │  ОЦЕНКА  │"&vbCrLf&_
         "├──┼──────────────┼──────────┤"&vbCrLf
StringJ="┌──┬──────────────┬──────────┐\r\n"+
        '│N│     ФИО      │  ОЦЕНКА  │\r\n'+
        "├──┼──────────────┼──────────┤\r\n ";
```

Необходимо отметить, что переход на новую строку может быть осуществлен в текстовом значении одним из непечатаемых знаков только при выводе сообщения на экран. При записи в файл операционная система Windows автоматически выполнит замену одного непечатаемого знака на соответствующую пару, поэтому при чтении информации из файла необходимо ожидать в качестве разделителя строк именно пару специальных символов. Символы с определенными ASCII кодами могут быть внесены в строку при помощи функции. В VBScript этой функцией является **Chr(КОД)**. Если текстовый символ входит в состав кодировочной системы Unicode (в которой удобнее пользоваться шестнадцатеричной системой обозначения кодов символов), тогда необходимо использовать функцию **ChrW(&НШКОД)**. В отличие от VBScript, являющегося регистронезависимым интерпретатором, JScript требует четкого разделения строчных и прописных букв в наименованиях, это касается как переменных, так и функций. В этом месте необходимо ввести новый термин – *объект*. В VBScript при вызове так называемых встроенных функций используется единое пространство имен, когда независимо от того, в какой части скрипта мы делаем вызов, будет использована именно та функция, имя которой мы указали и подразумевали в программе. Объектом же является некоторый компонент операционной системы, имеющий программную реализацию и, обычно один, *интерфейс*, посредством которого происходит взаимодействие между скриптом и функциональной составляющей объекта. Сервисные объекты, или объекты-утилиты, содержат набор статичных функций, которые никак не привязаны к какому-либо отдельному вызову и обычно при первом же обращении

полностью загружаются в оперативную память на весь период работы обратившегося к нему скрипта. По-другому, такие объекты называются *пакетами*. При этом интерфейс пакета является с одной стороны описанием всех включенных в него функций, а с другой – набором точек входа в программные процедуры. В JScript такие пакеты назначены в качестве поддерживающего функционала для каждого из типов данных. Обращение через интерфейс к функциям пакета осуществляется через знак точка «**.**», следуемого после указания наименования типа данных. Так строковый тип данных имеет соответствующий пакет, именуемый **`String`**. Для добавления символа по Unicode коду в JScript используется функция соответствующего пакета **`String.formCharCode(0xШКОД)`**. Также необходимо отметить, что двойные текстовые кавычки в VBScript можно указать в тексте через удвоение самого знака, так как ESC-последовательности используются только в JScript. При этом в JScript для указания одинарных кавычек внутри текстовой строки, обрамленной двойными кавычками, соответствующая ESC-последовательность не нужна, как и в обратной ситуации. Одинарная кавычка в VBScript внутри текстовой строки может использоваться любое количество раз, но если этот знак используется в других частях скрипта, тогда он воспринимается как оператор комментирования, то есть все, что располагается справа от него, игнорируется интерпретатором. Также для комментирования используется оператор **`Rem`**. Для комментирования в JScript используется двойная косая черта «//», а в случае необходимости указать многострочный комментарий можно воспользоваться комментирующими скобками «/*» и «*/».

Отдельно нужно сказать о преобразовании типов. В JScript такое преобразование в общих случаях не требуется, по причине того, что интерпретатор самостоятельно выполняет приведение типов к результирующему, либо к первому стоящему справа от знака равенства. В VBScript преобразование необходимо выполнять в обязательном порядке, иначе в момент достижения отдельной некорректной строки сценарий прервется с сообщением об ошибке. Числовое значение преобразуется к текстовому виду при помощи функции **`CStr(ЧИСЛО)`**.

```
Rem Добавим новые строки в таблицу
DividerVB=vbCrLf&"├─┼──────────────┼──────────┤"&vbCrLf
'Иванов
StringVB=StringVB&ChrW(&H2502)&CStr(1)&ChrW(&H2502)&_
"Иванов       "&ChrW(&H2502)&"""отлично"" "&_
ChrW(&H2502)&DividerVB
'Петров
StringVB=StringVB&ChrW(&H2502)&CStr(2)&ChrW(&H2502)&_
"Петров       "&ChrW(&H2502)&"""хорошо""  "&_
ChrW(&H2502)&DividerVB
'Сидоров
StringVB=StringVB&ChrW(&H2502)&CStr(3)&ChrW(&H2502)&_
"Сидоров      "&ChrW(&H2502)&"""удовлетв"""&_
ChrW(&H2502)&DividerVB
```

```
/*Добавим новые
строки в таблицу*/
DividerJ="\r\n\├─┼──────────────┼──────────┤\r\n";
//Иванов
StringJ=StringJ+String.fromCharCode(0x2502)+1+
String.fromCharCode(0x2502)+"Иванов       "+
String.fromCharCode(0x2502)+'"отлично" '+
String.fromCharCode(0x2502)+DividerJ;
//Петров
StringJ=StringJ+String.fromCharCode(0x2502)+2+
String.fromCharCode(0x2502)+"Петров       "+
String.fromCharCode(0x2502)+'"хорошо"  '+
String.fromCharCode(0x2502)+DividerJ;
//Сидоров
StringJ=StringJ+String.fromCharCode(0x2502)+3+
String.fromCharCode(0x2502)+"Сидоров      "+
String.fromCharCode(0x2502)+'"удовлетв"'+
String.fromCharCode(0x2502)+DividerJ;
```

Если в текстовых значениях переменных Вы используете символы псевдографики, а скрипт набирается при помощи программы «Блокнот», то желательно использовать кодировку Unicode, а в качестве шрифта выбрать Lucida Console. Также следует позаботиться о выборе этого же шрифта в опциях оформления рабочего стола, в частности, для элемента «Окно сообщения».

Числовые типы данных в VBScript представлены, относительно точности и диапазона хранимой в переменных информации, в шести видах:

Byte	целочисленные значения в диапазоне [0; 255]
Integer	целочисленные значения в диапазоне [-32768; 32767]
Currency	специальный формат для денежных величин
Long	целочисленные значения в диапазоне [-2147483648; 2147483647]
Single	числа с плавающей точкой одинарной точности
Double	числа с плавающей точкой двойной точности

Вообще, когда речь идет о типе данных, в VBScript следует понимать этот термин как некоторый подтип множества типа данных Variant, который включает в себя все используемые типы данных. Сам подтип определяется в момент операции присваивания переменной конкретного значения. Если необходимо указать точно, какой подтип должен использоваться именно для этой переменной, прибегают к помощи функций явного приведения: **`CByte(ЧИСЛО)`**, **`CInt(ЧИСЛО)`**, **`CCur(ЧИСЛО)`**, **`CLng(ЧИСЛО)`**, **`CSngl(ЧИСЛО)`**, **`CDbl(ЧИСЛО)`**, **`CDate(СТРОКА)`**, **`CBool(ЧИСЛО)`**, **`CStr(ЧИСЛО)`**. Типы Bool и Date/Time необходимы для работы с логическими выражениями и для операций с датой/временем. Кроме типизированных элементов переменные могут принимать значения **`Empty`** (для вновь объявляемых переменных, когда конкретное значение еще не задано), **`Null`** (указывает на то, что переменная не содержит каких-либо допустимых для

данного типа значений). Дополнительно используются подтипы данных Error (для хранения значений кодов ошибок) и Object (представляет собой ссылку на объект – сложно структурированный тип данных). Кроме этого, в качестве значения переменной, может использоваться **Nothing**, для освобождения ссылки на объект. При этом, переменную, которая будет содержать ссылку на объект, необходимо в обязательном порядке объявить оператором **Dim**, а в момент присвоения, кроме операции «=», нужно использовать оператор **Set**. Также оператор **Dim** применяется при объявлении массивов (индексация элементов при обращении к массиву начинается с нуля), объявление же прочих переменных необязательно.

В JScript числовой набор типов представлен классом целочисленных подтипов в диапазоне [-999999999999999; 999999999999999] и вещественных с двойной точностью $4{,}9*10^{-324}$ для диапазона [$-1{,}8*10^{308}$; $1{,}8*10^{308}$]. Кроме этого, наряду с пакетом для обработки текстовых строк, присутствуют пакеты для работы с типом данных **Date** (операции с датой/временем), а также со структурами данных **Array** (массивы, с началом индексации ноль) и **Enumerator** (коллекции). При работе с массивами необходимо выполнить объявление соответствующей переменной при помощи оператора **var**. Также в объявлении нуждаются переменные, которым в дальнейшем будут присваиваться значения сложных структур данных, таких как **Date**, **Array**, инициируемых оператором **new**, ссылок на объекты, в том числе и коллекции (обычно элементом коллекции является объект). В момент первого определения переменной она не содержит конкретного значения – в JScript используется специальный элемент **undefined**. Когда переменная освобождается от значения, например в виду удаления объекта из контекста выполняемой программы, ей присваивается специальное значение **null**.

Массивы в VBScript и JScript определяются по-разному. Так для определения массива, состоящего из двух элементов, используются следующие записи:

```
'Фрагмент на VBScript
Dim ArrVB(1)
ArrVB(0)="Первый элемент"
ArrVB(1)=2'явно указывается целочисленное значение
//Фрагмент на JScript
var ArrJ;
ArrJ=new Array(2);
ArrJ[0]="Первый элемент";
ArrJ[1]=2.0;//явно указывается вещественное число
```

Если размерность массива должна динамически расширяться, причем с сохранением прежних значений элементов, в VBScript и JScript также используются различные подходы. Необходимо напомнить, что для работы с массивами в JScript используется пакет методов для типа **Array** (по сути, с массивом обращаются как с объектом, имеющим как собственные методы, так и свойства, например, **length** – определяет количество элементов массива).

Для определения динамического массива в VBScript изначально размерность не указывается. Первоначальная размерность вводится оператором **ReDim**, который расширяет размерность массива и может вызываться неоднократно. В момент расширения границ массива, для обеспечения сохранности ранее внесенных в массив значений элементов, нужно использовать ключевое слово **Preserve**. Особенность JScript при изменении размерности массива заключается в том, что для этого достаточно изменить свойство массива **length** до требуемой величины.

```
'Фрагмент на VBScript
Dim DArrVB()
ReDim DArrVB(1)
DArrVB(0)="Первый элемент"
DArrVB(1)=2.5
ReDim Preserve DArrVB(3)
DArrVB(2)=2
DArrVB(3)=DArrVB(1)*DArrVB(2)'Результатом будет
'вещественное числовое значение 5.0
//Фрагмент на JScript
var DArrJ;
DArrJ=new Array(2);
DArrJ[0]="Первый элемент";
DArrJ[1]=2;
DArrJ.length=4;
DArrJ[2]=3.0;
DArrJ[3]=DArrJ[0]+DArrJ[1]*DArrJ[2];//Результатом будет
//текстовое значение
```

Организация вывода сообщений пользователю может быть реализована любым из предлагаемых Windows Script Host способов. Для работы со встроенными объектами системы используется обращение к пакету **WScript**. Объект может быть либо создан по желанию пользователя, либо предложен из пакета **WScript**. В консольном режиме скрипт может взаимодействовать с потоками ввода-вывода. Для работы с потоком ввода используется встроенный объект **WScript.StdIn**, который содержит функцию для прочтения текстовой строки с клавиатуры **ReadLine()**. Поток вывода представлен встроенным объектом **WScript.StdOut**, содержащим функцию вывода текстовой строки на экран монитора **WriteLine("ТЕКСТ")**. Однако, скрипт обычно запускается пользователем по двойному щелчку мыши, и работает в графическом режиме. Поэтому консольный режим ввода-вывода может понадобиться в исключительных случаях, когда функциональное назначение скрипта относится к классу утилит. Для вывода сообщений пользователю наиболее часто прибегают к использованию функции **Echo**, пакета **WScript**. При этом сообщения выводятся в консольном окне, в случае запуска через интерпретатор **cscript.exe**, а также в диалоговом окне с кнопкой [OK] при использовании интерпретатора **wscript.exe**.

Более широкие возможности по выводу текстовых сообщений пользователю предоставляет метод **`Popup`** объекта **`WScript.Shell`**, который позволяет определить не только текстовое сообщение, но и заголовок окна, а кроме этого выбрать наборы кнопок для ответной реакции пользователя на предоставленную информацию. Метод возвращает числовое значение, соответствующее определенному выбору пользователя.

Значение	**Константа VBScript**	**Описание**
-1		Пользователь не нажал ни на одну из кнопок в течение отведенного времени
1	vbOk	Нажата кнопка **OK**
2	vbCancel	Нажата кнопка **Отмена** (Cancel)
3	vbAbort	Нажата кнопка **Стоп** (Abort)
4	vbRetry	Нажата кнопка **Повтор** (Retry)
5	vbIgnore	Нажата кнопка **Пропустить** (Ignore)
6	vbYes	Нажата кнопка **Да** (Yes)
7	vbNo	Нажата кнопка **Нет** (No)

Метод может принимать как один параметр – текст сообщения, так и набор параметров в порядке: текст сообщения, время ожидания нажатия кнопки пользователем в секундах (0 – ожидание до момента нажатия кнопки), текст заголовка окна, тип набора кнопок в окне.

Значение	**Константа VBScript**	**Описание**
0	vbOkOnly	Выводится кнопка **OK**
1	vbOkCancel	Выводятся кнопки **OK** и **Отмена** (Cancel)
2	vbAbortRetryIgnore	Выводятся кнопки **Стоп** (Abort), **Повтор** (Retry) и **Пропустить** (Ignore)
3	vbYesNoCancel	Выводятся кнопки **Да** (Yes), **Нет** (No) и **Отмена** (Cancel)
4	vbYesNo	Выводятся кнопки **Да** (Yes) и **Нет** (No)
5	vbRetryCancel	Выводятся кнопки **Повтор** (Retry) и **Отмена** (Cancel)
16	vbCritical	Выводится значок 
32	vbQuestion	Выводится значок 
48	vbExclamation	Выводится значок 
64	vbInformation	Выводится значок 

Проанализировать результат, возвращаемый диалоговым окном при нажатии кнопки пользователем можно при помощи оператора условия. Конструкции, используемые в VBScript и JScript, различаются между собой.

В VBScript оператор условия выглядит так: **`If УСЛОВИЕ Then ВЫРАЖЕНИЕ1 Else ВЫРАЖЕНИЕ2`**. При этом если полная запись оператора условия не умещается в одну строку (например, одно из выражений конструкции выполняется в несколько операторов), тогда необходимо по окончанию описания конструкции указать операторную скобку **`End`** с

указанием наименования завершаемой команды, то есть **End If**. Такое правило применяется также и в других случаях. При выполнении логического условия (которое может быть истинной **true** или ложью **false**) оператор **If** запускает на исполнение выражение под номером 1, в противном случае – выражение под номером 2. Знаки логических соотношений и операций приведены ниже.

Оператор	**Описание оператора VBScript**
>	Левый операнд больше правого
>=	Правый операнд не больше левого
<	Левый операнд меньше правого
<=	Правый операнд не меньше левого
=	Левый и правый операнды равны
<>	Левый и правый операнды не равны
Not	Оператор отрицания
Or	Оператор отношения «ИЛИ», который возвращает истину при условии истинности хотя бы одного из операндов
Xor	Оператор отношения «ИСЛЮЧАЮЩЕЕ ИЛИ», который возвращает ложь при совпадении логических значений обоих операндов
And	Оператор отношения «И», который возвращает истину при истинности значений обоих операндов

При необходимости проверить множество сопоставлений различным значениям одной переменной используют условный оператор **Select Case**, после которого указывается наименование переменной, сопоставление значений которой будет производить условный оператор. Перечисление сопоставляемого значения осуществляется с новой строки после ключевого слова **Case**. Далее следует операция, выполняемая в случае истинности. Если есть необходимость указать действие по умолчанию (когда ни один из вариантов сопоставления не сработал), оно записывается после ключевых слов **Case Else**. По окончанию конструкции **Select Case** следует операторная скобка **End Select**.

В JScript оператор условия выглядит так: **if УСЛОВИЕ ВЫРАЖЕНИЕ1 else ВЫРАЖЕНИЕ2;**. Следует отметить, что в случае нескольких операций в одном из выражений (например, при использовании вложенных условий разветвленного дерева) оно заключается в операторные скобки, которые записываются знаками фигурных скобок «{» и «}», а каждая операция в выражении завершается знаком «**;**». Знаки логических соотношений и операций приведены ниже.

Аналогично конструкции **Select Case**, в JScript имеется оператор выбора **switch**, в котором указывается наименование сопоставляемой значениям переменной, после чего следуют операторные скобки. В них при помощи ключевых слов **case** указываются проверяемые значения. После знака двоеточие «**:**» записываются операторы, разделяемые знаком точка с запятой «**;**». Также имеется способ записать альтернативное действие при помощи ключевого слова **default:**.

Использование диалогового окна **Popup** требует предварительного инициирования объекта **Shell** из пакета **WScript**. Для его создания необходимо использовать метод **CreateObject("ПАКЕТНОЕ РАСПОЛОЖЕНИЕ ОБЪЕКТА")**, входящий в тот же самый пакет **WScript**. Только после этого можно обратиться к методам и свойствам созданного объекта через соответствующую переменную. Вообще, создаваемые объекты операционной системы могут быть представлены как программными модулями операционной системы и отдельными приложениями, зарегистрированными в системе, так и COM-объектами, в качестве которых могу выступать специально подготовленные скрипты на VBScript и JScript.

Оператор	**Описание оператора JScript**
>	Левый операнд больше правого
>=	Правый операнд не больше левого
<	Левый операнд меньше правого
<=	Правый операнд не меньше левого
==	Левый и правый операнды равны
!=	Левый и правый операнды не равны
!	Оператор отрицания
||	Оператор отношения «ИЛИ», который возвращает истину при условии истинности хотя бы одного из операндов
&&	Оператор отношения «И», который возвращает истину при истинности значений обоих операндов

Еще одним методом информирования пользователя является системный журнал событий, куда из скрипта тоже можно отправить уведомление. Для этого используется метод **LogEvent** объекта **Shell**. Первый параметр передает код типа сообщения (0 – сообщение об успешном завершении процесса, 1 – в процессе выполнения возникла ошибка, 2 – предупреждение, 4 – информирующее сообщение, и прочие специальные коды), а второй – непосредственно текстовую строку.

```
'Фрагмент на VBScript
WScript.Echo StringVB
Dim vbShell
Set vbShell=WScript.CreateObject("WScript.Shell")
vbPU=vbShell.Popup("Вы желаете записать таблицу в"&_
"системный журнал?", 5, "Вопрос", vbYesNo+vbQuestion)
Select Case vbPU
Case vbYes
vbShell.LogEvent 4, StringVB
vbShell.Popup "Действие выполнено!", 0, "Сообщение",_
vbInformation
Case vbNo
WScript.Echo "Вы отказались от действия!"
Case Else
WScript.Echo "Вы не сделали выбор!"
End Select
```

```
//Фрагмент на JScript
WScript.Echo(StringJ);
var jShell;
jShell=WScript.CreateObject("WScript.Shell");
jPU=jShell.Popup("Вы желаете записать таблицу в "+
"системный журнал?", 5, "Вопрос", 36);
switch (jPU){
case 6:
vbShell.LogEvent(4, StringJ);
vbShell.Popup("Действие выполнено!", 0, "Сообщение", 64);
case 7:
WScript.Echo("Вы отказались от действия!");
default:
WScript.Echo("Вы не сделали выбор!");}
```

ОРГАНИЗАЦИЯ ЦИКЛОВ, ФУНКЦИЙ И ПРОЦЕДУР

Конструкции для организации циклов в VBScript и JScript имеют много общего, за исключением операторных скобок. Начнем с рассмотрения цикла **while**. Итерации выполняются при истинности логического условия, то есть данная конструкция является циклом с предусловием. Однако выйти из любого цикла до завершения итераций на JScript можно при помощи оператора **break;**, а при помощи оператора **continue;** – прервать текущую итерацию.

```
'Фрагмент на VBScript
While (УСЛОВИЕ)
...
WEnd
```

```
//Фрагмент на JScript
while (УСЛОВИЕ){
...
}
```

Другая конструкция, **do**, относится к разновидности цикла с постусловием. В случае с VBScript условие может менять не только расположение в начале или в конце цикла, но и логический смысл, для чего используются ключевые слова **While** или **Until**. Первое устанавливает границу выполнения итераций в цикле на истинность условия, второе – на ложность. Выйти из цикла можно при помощи оператора **Exit Do**.

```
'Фрагмент на VBScript
Do Until (УСЛОВИЕ)
...
Loop
Do
...
Loop While (УСЛОВИЕ)
```

```
Do
...
Loop Until (УСЛОВИЕ)
//Фрагмент на JScript
do{
...
}while (УСЛОВИЕ);
```

Еще одна конструкция, **for**, организует цикл в жестко заданных границах, которые могут регламентироваться определенными числовыми значениями начала и окончания выполнения цикла, а также порядка приращения значения, кроме этого допускается более мягкое определение границ выполнения – использование коллекции элементов (динамического массива объектов), количество которых не определено на момент начала

выполнения цикла. Для этого используются ключевые слова **Each ПЕРЕМЕННАЯ In КОЛЛЕКЦИЯ** в VBScript. Выход из цикла до завершения итераций выполняется при помощи оператора **Exit For**. Обработка коллекции элементов в JScript связана с предварительной операцией инициирования соответствующего объекта. Коллекцией обычно выступает некоторое свойство какого-либо другого объекта, ее содержащего в качестве значения. Для перемещения по коллекции используются соответствующие методы **moveNext()** для перехода к последующему элементу и **atEnd()** для проверки момента достижения окончания коллекции.

```
'Фрагмент на VBScript
For vbItem=1 To 5 Step 0.2
...
Next
For Each vbCollection In КОЛЛЕКЦИЯ
...
Next
//Фрагмент на JScript
for (jItem=1; jItem<=5; jItem=jItem+0.2){
...
}
var jCollection=new Enumerator(КОЛЛЕКЦИЯ);
for (; !jCollection.atEnd(); jCollection.moveNext()){
...
}
```

Описывая в скрипте некоторые повторяющиеся последовательности операций, особенно связанные с алгоритмическими действиями каких-либо методов вычислений, нередко формируют отдельный программный модуль с отдельным именем, к которому, при необходимости, неоднократно обращаются из других частей скрипта. Подобное структурирование программы помогает впоследствии легко читать код и вносить коррективы, улучшающие его функциональные характеристики. Примером применения такого подхода являются системные библиотеки функций операционной системы, сконцентрированные в DLL-файлах, программы-утилиты, выполняющие любое количество раз выбранную пользователем однотипную операцию. Например, обратившись к системной библиотеке из командной строки: **RunDll32.EXE Shell32.DLL,LockWorkStation** , можно вызвать функцию блокирования компьютера. В данном примере программой-утилитой является **RunDll32.EXE**, а системной библиотекой – **Shell32.DLL**. Описывая функции или процедуры, при условии соответствующего декларирования, можно использовать их за рамками программного модуля. Отличительной особенностью функции от процедуры является предоставление пользователю или программе, ее вызвавшей, какого-либо результирующего значения. Для написания процедуры в VBScript используется конструкция **Sub ... End Sub**, в JScript эта функция возложена на декларирующий оператор **function**,

который используется и для определения функции. Если процедура принимает параметры, то они указываются через запятую в круглых скобках после наименования процедуры. Если параметры не используются, то круглые скобки тоже записываются, но пустые. При этом все локальные переменные нужно объявлять в начале процедуры – это скорее требование этикета программирования, причем обязательное. Действие локальных переменных заканчивается, как только программа выходит из процедуры. При совпадении наименований переменных происходит локальная подмена на момент выполнения подпрограммы. Если необходимо обеспечить переменную видимостью во всех процедурах и функциях скрипта, то, по требованию этикета программирования, необходимо объявить глобальную переменную в самом начале скрипта. При этом видимость переменной за пределами файла исключается. Для этих целей используются аргументы вызова подпрограмм.

```
'Фрагмент на VBScript
Sub ПРОЦЕДУРА(ПАРАМЕТРЫ)
Dim ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ
...
End Sub
```

```
//Фрагмент на JScript
function ПРОЦЕДУРА(){
var ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ;
...
}
```

При описании функций используется оператор **function**. Возвращаемая величина указывается в VBScript через присвоение имени функции этого самого значения, а в JScript при помощи оператора **return**.

```
'Фрагмент на VBScript
Function ФУНКЦИЯ()
Dim ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ
...
ФУНКЦИЯ=РЕЗУЛЬТАТ
End Function
```

```
//Фрагмент на JScript
function ФУНКЦИЯ(){
var ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ;
...
return РЕЗУЛЬТАТ;
}
```