МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

Коновалов В.М.

ПОСОБИЕ
к выполнению лабораторных работ
по дисциплине
«Прикладное программное обеспечение»
выпуск 3

для студентов
специальности 0730
дневного обучения

Москва – 2002

МИНИСТЕРСТВО ТРАНСПОРТА РФ
ГОСУДАРСТВЕННАЯ СЛУЖБА ГРАЖДАНСКОЙ АВИАЦИИ
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

Кафедра прикладной математики

Коновалов В.М.

ПОСОБИЕ
к выполнению лабораторных работ
по дисциплине
«Прикладное программное обеспечение»
выпуск 3

для студентов
специальности 0730
дневного обучения

Москва – 2002

Содержание

Основные понятия и определения

Динамический обмен данными (DDE)

Согласованный протокол обмена данными между приложениями, работающими в среде Windows.

Документ

Субъект динамического обмена данными между приложениями по протоколу DDE. Для большинства приложений, использующих файлы, документ представляет имя файла. Кроме того, большинство приложений распознает системный документ «System». Документ «System» позволяет получить сведения о приложении, например, имена других документов, распознаваемых данным приложением.

Канал связи DDE

Активный канал связи между приложениями Microsoft Windows, по которому осуществляется обмен данными.

Номер канала

Целое значение, соответствующее открытому каналу динамического обмена данными. Номера каналов присваиваются Microsoft Windows или создаются функцией DDEInitiate и используются другими функциями и инструкциями DDE, например, DDERequest.

Объект

Комбинация кода и данных, которая может рассматриваться как единое целое, например, элемент управления, форма или компонент приложения. Каждый объект определяется по принадлежности к классу.

Программируемый объект

Объект, который может быть открыт для других приложений и средств программирования через интерфейсы программирования.

Компонент/компонент ActiveX

Приложение, которое может использовать объекты из другого приложения или предоставляет собственные объекты для использования в другом приложении.

Программирование объектов

Стандартное средство для работы с объектами некоторого приложения из другого приложения или среды разработки. Программирование объектов (ранее называемое программированием OLE) является функцией модели COM (Component Object Model).

Раздел данных

Зависящий от приложения элемент данных, который может быть передан по каналу связи DDE.

Функция DDE

Функция DDE позволяет открыть сеанс динамического обмена данными (DDE) с другим приложением, направить требование на прием данных из этого приложения и вывести полученные данные в элементе управления в форме или отчете.

Например, указание функции DDE в свойстве **Данные (ControlSource)** поля в форме позволяет вывести в этом поле содержимое конкретной ячейки электронной таблицы Microsoft Excel.

Синтаксис:

DDE(*приложение, документ, раздел*)

Функция DDE использует следующие аргументы:

приложение - строковое выражение, которое определяет приложение, участвующее в сеансе DDE. Обычно, для приложений, работающих в среде Microsoft Windows, аргумент *приложение* задает имя файла .EXE (без расширения .EXE). Например, для

открытия канала связи DDE с Microsoft Excel следует указать "Excel" в аргументе *приложение*;

документ - строковое выражение, содержащее имя документа, принимаемое *приложением*. В аргументе *документ* обычно указывается документ или файл данных. За списком поддерживаемых приложением имен документов следует обращаться к документации данного *приложения*;

раздел - строковое выражение, содержащее имя раздела данных, принимаемое *приложением*. За списком поддерживаемых приложением имен разделов данных следует обращаться к документации данного *приложения*.

При вызове функции DDE делается попытка открыть сеанс связи DDE с указанным *приложением*, передать соответствующее имя *документа* и требование на прием данных, указанных в аргументе *раздел*. При успешном выполнении функция DDE возвращает строку, содержащую затребованные данные.

В требовании на прием данных из Microsoft Excel аргумент *раздел* должен содержать идентификатор адреса ячейки (строки и столбца, например, "R1C1") или имя диапазона ячеек. В следующем примере функция DDE передает требование на прием содержимого ячейки, находящейся в первой строке и первом столбце электронной таблицы Microsoft Excel. В ячейку **Данные (ControlSource)** в окне свойств поля вводится следующее выражение:

=DDE("Excel", "Лист1", "R1C1")

Функция DDE используется только для указания свойства **Данные (ControlSource)** поля, группы, флажка или поля со списком. Не допускается вызов функции DDE в инструкциях Visual Basic.

После ввода функции DDE элемент управления становится нередактируемым в режиме формы и предварительного просмотра.

Например, после ввода функции DDE в поле это поле становится нередактируемым. Поскольку свойство **Данные (ControlSource)** становится в режимах формы и предварительного просмотра доступным только для чтения, все изменения должны вноситься в элемент управления в режиме конструктора.

Максимальное количество одновременно открываемых сеансов DDE определяется настройками Windows, а также памятью и ресурсами компьютера. Если попытка открыть сеанс оказалась неудачной из-за того, что *приложение* не запущено, не распознается *документ* или превышено максимально допустимое число сеансов, функция DDE возвращает значение **Null**.

Примечание. В конфигурации другого приложения может быть указано, что запросы на открытие сеанса DDE игнорируются. В этом случае функция DDE возвращает **Null**. Аналогично, в настройках Microsoft Access можно указать игнорирование запросов на открытие сеансов из других приложений: для этого выберите в меню **Сервис** команду **Параметры**, выберите вкладку **Другие** и в группе **Операции DDE** установите флажок **Пропуск команд DDE**.

Ниже приводится описание использования функции DDE с разными элементами управления.

Поле. Аргумент *раздел* может представлять текст или числа. Если *раздел* представляет несколько элементов данных, например, имя диапазона, содержащего несколько ячеек электронной таблицы Microsoft Excel, то функция DDE возвращает содержимое первого элемента. Функция DDE позволяет вывести в поле содержимое ячейки электронной таблицы.

Поле со списком. Функция DDE заполняет список данными, указанными в аргументе *раздел*. Не допускается ввод данных в поле.

Функция DDE позволяет вывести в поле со списком список значений, сохраняемых в электронной таблице Microsoft Excel.

Группа параметров. Свойство **Значение параметра (OptionValue)** каждого из переключателей в группе имеет числовое значение. Обычно первый переключатель имеет значение 1, второй - значение 2 и т.д. Числовое значение, возвращаемое функцией DDE, определяет, какой из переключателей будет выбран. Например, если функция DDE возвращает 2, будет включен второй переключатель. Если функция DDE возвращает значение, не соответствующее ни одному из свойств **Значение параметра (OptionValue)**, не будет выбран ни один из переключателей. Если *раздел* представляет несколько элементов данных, например, имя диапазона, содержащего несколько ячеек электронной таблицы Microsoft Excel, то функция DDE возвращает содержимое первого элемента.

Флажок. Если функция DDE возвращает 0, флажок будет снят. Если функция DDE возвращает любое ненулевое значение, например, 1 или –1, флажок будет установлен. Если *раздел* представляет несколько элементов данных, например, имя диапазона, содержащего несколько ячеек электронной таблицы Microsoft Excel, состояние флажка становится неопределенным.

Функция DDEInitiate

Функция DDEInitiate позволяет открыть сеанс динамического обмена данными с другим приложением. Функция DDEInitiate открывает канал связи DDE, обеспечивающий передачу данных между сервером DDE и приложением-клиентом.

Например, для передачи данных из электронной таблицы Microsoft Excel в базу данных Microsoft Access следует открыть канал связи между двумя приложениями с помощью функции DDEInitiate. В

этом случае Microsoft Access будет выполнять роль приложения-клиента, а Excel приложения-сервера.

Синтаксис:

DDEInitiate(*приложение, документ*)

Функция DDEInitiate использует следующие аргументы:

приложение - строковое выражение, которое определяет приложение, участвующее в сеансе DDE;

документ - строковое выражение, содержащее имя документа, принимаемое *приложением*.

При успешном выполнении функция DDEInitiate открывает сеанс связи DDE с указанными *приложением* и *документом* и возвращает значение типа **Long**. Данное число представляет уникальный номер канала, определяющий канал, по которому будет проводиться обмен данными. Этот номер канала будет использоваться в аргументах других функций и инструкций DDE.

Если приложение еще не запущено, а также если запущенное приложение не принимает аргумент *документ* или не поддерживает протокол DDE, функция DDEInitiate возвращает ошибку при выполнении.

Допустимые значения аргумента *документ* определяются *приложением*. В приложениях, использующих документы или файлы данных, допустимыми значениями данного аргумента обычно являются имена этих файлов.

Примечание. Максимально возможное число одновременно открытых каналов определяется настройками Windows, а также системной памятью и ресурсами компьютера. Если канал не используется, то для экономии ресурсов следует закрыть его с помощью инструкций DDETerminate или DDETerminateAll.

Инструкция DDEExecute

Инструкция DDEExecute передает по открытому каналу динамического обмена данными команду из приложения-клиента в приложение-сервер.

Предположим, например, что канал DDE открыт в Microsoft Access для передачи текста из электронной таблицы Microsoft Excel в базу данных Microsoft Access. Инструкция DDEExecute позволяет передать в Microsoft Excel команду **New (Создать)** для создания новой электронной таблицы. В этом случае Microsoft Access является приложением-клиентом, а Microsoft Excel приложением-сервером.

Синтаксис:

DDEExecute *канал, команда*

Инструкция DDEExecute использует следующие аргументы:

канал - номер канала, представленный длинным целым числом, который возвращается функцией DDEInitiate;

команда - строковое выражение, задающее команду, принимаемую приложением-сервером. Список допустимых команд можно определить по документации приложения-сервера.

Допустимые значения аргумента *команда* зависят от выбранного приложения и имени документа, указанного при открытии данного *канала*. Выполнение данной инструкции приводит к ошибке, если значение аргумента *канал* не является целым числом, соответствующим открытому каналу, или если указанная команда не может быть выполнена в другом приложении.

В программах Visual Basic инструкцию DDEExecute можно использовать только для передачи команд в другие приложения. Сведения о передаче команд из других приложений в Microsoft

Access приведены в разделе Использование Microsoft Access в роли сервера DDE.

Инструкция DDEPoke

Инструкция DDEPoke позволяет передать текст из приложения-клиента в приложение-сервер по открытому каналу динамического обмена данными. Например, если открыт канал связи DDE между Microsoft Access и Microsoft Excel, инструкция DDEPoke позволяет передать текст из базы данных Microsoft Access в электронную таблицу Microsoft Excel. В этом случае Microsoft Access будет выполнять роль приложения-клиента, а Excel приложения-сервера.

Синтаксис:

DDEPoke *канал, раздел, данные*

Инструкция DDEPoke использует следующие аргументы:

канал - номер канала, целое значение, возвращаемое функцией DDEInitiate;

раздел - строковое выражение, содержащее имя раздела данных, принимаемое приложением, заданным в функции DDEInitiate;

данные - строка, содержащая данные, передаваемые в приложение.

Допустимые значения аргумента *раздел* определяются именами приложения и документа, указанными при открытии *канала*. Например, *разделом* может быть диапазон ячеек электронной таблицы Microsoft Excel.

Строка *данные* должна передаваться в текстовом формате. Другие форматы при передаче данных не поддерживаются. Например, в аргументе *данные* можно передать числа для заполнения указанного диапазона электронной таблицы Excel.

Аргумент *канал* должен представлять целое число, совпадающее с номером открытого канала. При указании другого значения, а также если приложение-получатель не распознает или не принимает указанные данные, возникает ошибка при выполнении.

Функция DDESend

Функция DDESend позволяет открыть сеанс динамического обмена данными с другим приложением и передать в это приложение данные из элемента управления в форме или отчете.

Например, функция DDESend вводится в ячейку свойства **Данные (ControlSource)** поля, чтобы передать данные из этого поля в ячейку электронной таблицы Microsoft Excel.

Синтаксис:

DDESend(*приложение, документ, раздел, данные*)

Функция DDESend использует следующие аргументы:

приложение - строковое выражение, которое определяет приложение, участвующее в сеансе DDE. Обычно, для приложений, работающих в среде Microsoft Windows, аргумент *приложение* задает имя файла .EXE (без расширения .EXE). Например, для открытия канала связи DDE с Microsoft Excel следует указать "Excel" в аргументе *приложение*;

документ - строковое выражение, содержащее имя документа, принимаемое *приложением*. В аргументе *документ* обычно указывается документ или файл данных. За списком поддерживаемых приложением имен документов следует обращаться к документации данного *приложения*;

раздел - строковое выражение, содержащее имя раздела данных, принимаемое *приложением*. За списком поддерживаемых

приложением имен разделов данных следует обращаться к документации данного приложения.

данные - строка или выражение, содержащие данные, передаваемые в *приложение*.

При вызове функции DDESend делается попытка открыть сеанс связи DDE с указанным *приложением*, передать соответствующее имя *документа* и указать *раздел*, который должен принять *данные*. Например, если аргумент *приложение* определяет Microsoft Excel, *документ* может иметь вид "Лист1", а *раздел* представлять идентификатор адреса ячейки (строки и столбца, например, "R1C1") или имя диапазона ячеек.

Аргумент *данные* определяет передаваемые данные. Этот аргумент может содержать строку, например, "Отчет подготовил В.Сидоров" или выражение, включающее функцию как часть результирующей строки, например, "Отчет подготовлен " & Date(). Если в аргументе *раздел* указывается несколько адресов элементов, например, имя диапазона ячеек электронной таблицы Microsoft Excel, функция DDESend передает *данные* в первый элемент.

В следующем примере функция DDESend передает строку "Некий текст" в первую ячейку электронной таблицы Microsoft Excel. Данное выражение может быть введено в ячейку **Данные (ControlSource)** в окне свойств поля в следующем виде:

=DDESend("Excel", "Лист1", "R1C1", "Некий текст")

Предположим теперь, что требуется передать данные из присоединенного элемента управления, находящегося в форме Microsoft Access, в ячейку электронной таблицы Microsoft Excel. В свойстве **Данные (ControlSource)** связанного элемента управления уже указано имя поля или выражения. В этом случае следует создать еще одно поле или поле со списком, задать для него в свойстве

Данные выражение, включающее функцию DDESend, и указать в аргументе *данные* имя связанного элемента управления. Например, если связанным является поле "Фамилия", то в ячейку **Данные** второго поля следует ввести такое выражение:

=DDESend("Excel", "Лист1", "R1C1", [Фамилия])

В качестве промежуточного элемента управления необходимо использовать поле или поле со списком. Не допускается указание имени связанного элемента управления в аргументе данные для флажка или группы переключателей.

Функция DDESend используется только для указания свойства **Данные (ControlSource)** поля, группы, флажка или поля со списком. Не допускается вызов функции DDESend в инструкциях Visual Basic.

После ввода функции DDESend элемент управления становится нередактируемым в режиме формы и предварительного просмотра. Поскольку свойство **Данные (ControlSource)** становится в режимах формы и предварительного просмотра доступным только для чтения, все изменения должны вноситься в элемент управления в режиме конструктора.

Максимальное количество одновременно открываемых сеансов DDE определяется настройками Windows, а также памятью и ресурсами компьютера. Если попытка открыть сеанс оказалась неудачной из-за того, что *приложение* не запущено, не распознается *документ* или превышено максимально допустимое число сеансов, функция DDESend возвращает значение **Null**.

Примечание. В конфигурации другого приложения может быть указано, что запросы на открытие сеанса DDE игнорируются. В этом случае функция DDESend возвращает **Null**. Аналогично, в настройках Microsoft Access можно указать игнорирование запросов

на открытие сеансов из других приложений: для этого выберите в меню **Сервис** команду **Параметры**, выберите вкладку **Другие** и в группе **Операции DDE** установите флажок **Пропуск команд DDE**.

Ниже описано использование функции DDESend с разными элементами управления.

Поле или поле со списком. Так как в режимах формы и предварительного просмотра поле (промежуточный элемент управления) выводится пустым, для его свойства **Вывод на экран (Visible)** можно задать значение **False**. Аргумент *данные* содержит ссылку на другой элемент управления. В следующем примере демонстрируется передача содержимого поля «Тел./факс» в электронную таблицу Microsoft Excel.

=DDESend("Excel", "Лист2", "R2C10", [Тел./факс])

Группа параметров. Ни один из переключателей в группе не является выбранным в режимах формы и предварительного просмотра, поэтому для свойства **Вывод на экран (Visible)** группы (и содержащихся в ней переключателей) можно задать значение **False**. Аргумент *данные* должен содержать числовое значение, например, "2". Если значение аргумента *данные* не является числовым, функция DDESend не передает информацию и состояние принимающего элемента не изменяется.

Флажок. Состояние флажка в режиме формы и предварительного просмотра является неопределенным, поэтому для его свойства **Вывод на экран (Visible)** можно задать значение **False**. Аргумент *данные* должен содержать числовое значение, например, "2". Если значение аргумента *данные* не является числовым, функция DDESend не передает информацию и состояние принимающего элемента не изменяется.

Функция DDERequest

Функция DDERequest передает в приложение-сервер по открытому каналу динамического обмена данными требование на прием данных из указанного раздела. Например, если открыт канал связи DDE между Microsoft Access и Microsoft Excel, функция DDERequest позволяет передать текст из электронной таблицы Microsoft Excel в базу данных Microsoft Access. В этом случае Microsoft Access будет выполнять роль приложения-клиента, а Excel приложения-сервера.

Синтаксис:

DDERequest(*канал, раздел*)

Функция DDERequest использует следующие аргументы:

канал - номер канала. Целое значение, возвращаемое функцией DDEInitiate;

раздел - строковое выражение, содержащее имя раздела данных, принимаемое приложением, определенным при вызове функции DDEInitiate.

Аргумент *канал* задает номер используемого канала связи DDE, а аргумент *раздел* определяет данные, загружаемые из приложения - сервера. Допустимые значения аргумента *раздел* определяются именами приложения и документа, указанными при открытии *канала*. Например, разделом может быть диапазон ячеек электронной таблицы Microsoft Excel.

При успешном выполнении функция DDERequest возвращает значение типа **Variant**, в виде строки, содержащей затребованные данные.

Допускается прием данных только в обычном текстовом формате. Прием рисунков или текста в другом формате не поддерживается.

Аргумент *канал* должен представлять целое число, совпадающее с номером открытого канала. При указании другого значения, а также при невозможности передачи затребованных данных, возникает ошибка при выполнении.

Инструкция DDETerminate

Инструкция DDETerminate закрывает указанный канал динамического обмена данными. Например, если открыт канал связи DDE между Microsoft Access и Microsoft Excel, инструкция DDETerminate позволяет закрыть канал после завершения передачи данных.

Синтаксис:

DDETerminate *канал*

Аргумент *канал* задает номер канала. Значение номера канала возвращается функцией DDEInitiate при открытии канала. Если указанное в аргументе *канал* значение не представляет номер открытого канала, возникает ошибка при выполнении.

После закрытия канала вызов любой функции или инструкции DDE с номером этого канала приводит к ошибке при выполнении.

Вызов инструкции DDETerminate не оказывает влияния на активизированные связи DDE, задаваемые с помощью выражений для полей в формах или отчетах.

Инструкция DDETerminateAll

Инструкция DDETerminateAll закрывает все открытые каналы динамического обмена данными. Например, если были открыты два канала связи DDE между Microsoft Excel и Microsoft Access, один для загрузки системной информации о Microsoft Excel и второй для

передачи данных, то инструкция DDETerminateAll позволяет закрыть оба канала одновременно.

Синтаксис:

DDETerminateAll

Вызов инструкции DDETerminateAll эквивалентен выполнению инструкции DDETerminate для каждого открытого канала. Аналогично инструкции DDETerminate, инструкция DDETerminateAll не оказывает влияния на активизированные связи DDE, задаваемые с помощью выражений для полей в формах или отчетах. Если открытых каналов связи DDE нет, вызов DDETerminateAll приводит к ошибке при выполнении.

При прерывании процедуры, осуществляющей обмен по протоколу DDE, некоторые каналы могу случайно оказаться открытыми. Для экономии системных ресурсов рекомендуется вызывать инструкцию DDETerminateAll в программе или из панели проверки окна отладки во время отладки программ, в которых выполняются операции DDE.

Программирование объектов

Программирование объектов (ранее программирование OLE) является свойством модели COM (Component Object Model), стандартной технологии, которая используется приложениями, чтобы предоставить свои объекты в распоряжение средств разработки, макроязыков и других приложений, поддерживающих программирование объектов. Например, приложение для работы с электронными таблицами может предоставлять для использования лист, диаграмму, ячейку или диапазон ячеек в качестве различных типов объектов. Текстовый процессор может предоставлять для использования объекты типа приложений, документов, абзацев, предложений, закладок или выделенных фрагментов.

Если приложение поддерживает программирование объектов, предоставляемые им объекты доступны из языка Visual Basic. Visual Basic позволяет проводить обработку этих объектов с помощью методов этих объектов или с помощью чтения или установки свойств этих объектов. Например, если был создан программируемый объект по имени MyObj, для управления этим объектом можно использовать следующие инструкции:

```
MyObj.Insert "Всем привет."          ' Размещает текст.
MyObj.Bold = True                    ' Форматирует текст.
MyObj.SaveAs  "C:\WORDPROC\ TESTOBJ.DOC"      'Сохраняет объект.
```

Следующие функции позволяют получить доступ к программируемому объекту

CreateObject - Создает новый объект указанного типа.

GetObject - Загружает объект из файла.

Сведения о поддерживаемых объектом свойствах и методах содержатся в документации к определенному приложению. Объекты, функции, свойства и методы, поддерживаемые приложением, определяются обычно в библиотеке объектов приложения.

В Microsoft Access объект программирования может быть создан либо с помощью ключевого слова New, использованного при описании объектной переменной, либо с помощью функций CreateObject или GetObject.

Microsoft Access поддерживает два различных способа программирования объектов. Microsoft Access является компонентом ActiveX, т. е. существует возможность доступа к объектам, предоставленным для использования другими приложениями, и работа с ними из Microsoft Access, а объекты

Microsoft Access могут управляться другими приложениями, поддерживающими программирование объектов.

Примечание. Перед использованием Microsoft Access или другого приложения в качестве компонента необходимо определить ссылку на объектную библиотеку этого приложения. Более подробные сведения об определении ссылок можно получить, выполнив в предметном указателе интерактивной справочной системы поиск строки «определение ссылок».

Например, при использовании Microsoft Access в качестве компонента возможна работа с объектами Microsoft Excel. Для этого в Microsoft Access следует создать новый объект Microsoft Excel и присвоить объектной переменной ссылку на него. После этого, как и для любого другого объекта, становится доступным изменение его свойств и вызов его процедур. Новый экземпляр класса Microsoft Excel Application может быть создан двумя способами.

```
Dim appXL As New Excel.Application
' Или
Dim appXL As Excel.Application
Set appXL = CreateObject("Excel.Application")
```

Если Microsoft Excel уже запущен, для получения ссылки на текущий экземпляр класса Application Microsoft Excel можно воспользоваться функцией GetObject.

```
Dim appXL As Excel.Application
Set appXL = GetObject( , "Excel.Application")
```

Более подробные сведения об объекте Application можно получить, выполнив в предметном указателе интерактивной справочной системы поиск строки «Application - объект».

При использовании Microsoft Access в качестве компонента, возможна работа с его объектами из других приложений. Например, в Microsoft Excel можно открыть форму Microsoft Access и

использовать ее для просмотра книги Microsoft Excel. В следующем примере из Microsoft Excel создается новый объект Microsoft Access.

```
Dim appAccess As New Access.Application
```

Если Microsoft Access уже запущен, для получения ссылки на текущий экземпляр класса Application Microsoft Access можно воспользоваться функцией GetObject.

Пример динамического обмена данными

Данная программа открывает сеанс обмена по протоколу DDE с Microsoft Excel, помещает значения в ячейки первой строки новой электронной таблицы и создает диаграмму по этим значениям. Вначале инструкция DDEInitiate открывает канал связи DDE. Затем в инструкции DDEExecute в Microsoft Excel передается команда открытия новой электронной таблицы. В инструкции DDERequest в Microsoft Excel запрашивается имя созданной электронной таблицы. После этого открывается новый канал и инструкция DDEPoke передает в Microsoft Excel данные, по которым строится диаграмма. И наконец, инструкция DDETerminate закрывает канал связи DDE с Microsoft Excel, а инструкция DDETerminateAll закрывает все активные каналы DDE.

```
Sub ExcelDDE()
Dim intI As Integer, intChan1 As Integer
Dim strTopics As String, strResp As String, strSheetName As String
On Error Resume Next                                    ' Обработка ошибок.
intChan1 = DDEInitiate("Excel", "System")               ' Открытие сеанса обмена.
If Err Then                                             ' При возникновении ошибки
        Err = 0                 ' электронная таблица не запускается. Сброс ошибки
        Shell "C:\Excel\Excel.exe", 1                           ' и запуск.
        If Err Then Exit Sub                            ' Выход при повторной ошибке.
```

```
' Открытие сеанса обмена.
        intChan1 = DDEInitiate("Excel", "System")
End If
' Создание новой электронной таблицы.
DDEExecute intChan1, "[New(1)]"
' Считываем список имен документов, имя таблицы.
strTopics = DDERequest(intChan1, "Selection")
strSheetName = Left(strTopics, InStr(1, strTopics, "!") - 1)
' Закрываем канал DDE.
DDETerminate intChan1
' Открываем канал обмена с новой электронной таблицей.
intChan1 = DDEInitiate("Excel", strSheetName)
For intI = 1 To 10                                  ' Заносим значения.
        DDEPoke intChan1, "R1C" & intI, intI        ' в первую строку.
Next intI
' Строим диаграмму.
DDEExecute intChan1, "[Select(""R1C1:R1C10"")][New(2,2)]"
' Закрываем все каналы обмена.
DDETerminateAll
End Sub
```

Использование Microsoft Access в роли сервера DDE

Microsoft Access поддерживает протокол динамического обмена данными (DDE, dynamic data exchange), выполняя роль как приложения, принимающего данные (клиента), так и источника данных (сервера). Например, база данных Microsoft Access может быть сервером для приложения Microsoft Word, выступающего в роли клиента, принимающего данные по каналу связи DDE.

Для того чтобы работать с объектами другого приложения из Microsoft Access, следует использовать механизм программирования объектов.

Сеанс связи DDE между клиентом и сервером открывается для конкретного документа. Документ может быть либо файлом данных в формате, поддерживаемом приложением-сервером, либо являться системным документом, содержащим информацию о самом приложении-сервере. Во время сеанса связи, открытого для конкретного документа, возможна передача данных только из раздела данных, связанного с этим документом.

Предположим, например, что выполняется приложение Microsoft Word для Windows, в документ которого требуется вставить данные из конкретной базы данных Microsoft Access. Сеанс связи DDE с Microsoft Access начинается с открытия канала связи DDE с помощью функции DDEInitiate и указания в качестве документа имени файла базы данных. После этого становится возможной передача данных в Microsoft Word по этому каналу.

В качестве сервера DDE Microsoft Access поддерживает следующие документы:

- документ System (системный);
- имя файла базы данных (документ *базаДанных*);
- имя таблицы (документ *имяТаблицы*);
- имя запроса (документ *имяЗапроса*);
- инструкция SQL Microsoft Access (документ *инструкцияSQL*).

После открытия сеанса обмена данными по протоколу DDE для передачи команд из приложения-клиента в приложение сервер используют инструкцию DDEExecute. В роли сервера DDE Microsoft Access принимает как допустимую любую из перечисленных ниже команд.

- Имя макроса в текущей базе данных.
- Любая макрокоманда, которая может быть выполнена в программе Visual Basic с помощью одного из методов объекта **DoCmd**.
- Макрокоманды **OpenDatabase** и **CloseDatabase**, которые используются только в инструкциях DDE. (Примеры использования этих макрокоманд приведены ниже в данном разделе).

Примечание. При указании имени макрокоманды в инструкции DDEExecute сама макрокоманда и все ее аргументы должны в соответствии с синтаксисом объекта **DoCmd** заключаться в прямые скобки ([ ]). Приложения, поддерживающие протокол DDE, не допускают использования внутренних констант в операциях DDE. Строковые аргументы должны заключаться в прямые кавычки (" "), если в строке содержится запятая. В остальных случаях кавычки не являются обязательными.

Вызванная в приложении-клиенте функция DDERequest передает требование на пересылку текстовых данных из приложения-сервера по открытому каналу DDE. Для пересылки данных из приложения-клиента в приложение-сервер следует вызвать функцию DDEPoke. После того как передача данных закончится следует вызвать в приложении-клиенте функцию DDETerminate для закрытия текущего канала обмена данными DDE или функцию DDETerminateAll для закрытия всех открытых каналов.

В следующем примере демонстрируется создание макроса на языке Visual Basic приложения Word для Windows, использующего Microsoft Access в качестве сервера DDE. (Для выполнения данного примера необходимо предварительно запустить Microsoft Access).

```
Sub AccessDDE()
Dim intChan1 As Integer, intChan2 As Integer
Dim strQueryData As String
' Используем документ System и открываем базу данных Base.mdb.
' Необходимо открыть базу данных перед использованием других документов
' DDE.
intChan1 = DDEInitiate("MSAccess", "System")
' Данный путь следует заменить на реальный путь к базе данных Base.mdb.
DDEExecute   intChan1, "[OpenDatabase C:\Access\Samples\Base.mdb]"
' Считываем все данные из запроса «Десять самых дорогих товаров».
intChan2 = DDEInitiate("MSAccess", "Base.mdb;" _
                        & "QUERY Десять самых дорогих товаров")
strQueryData = DDERequest(intChan2, "All")
DDETerminate   intChan2
' Закрываем базу данных.
DDEExecute   intChan1, "[CloseDatabase]"
DDETerminate   intChan1
' Печатаем полученные данные в окне отладки.
Debug.Print   strQueryData
End  Sub
```

Ниже приводится описание допустимых имен документов DDE, поддерживаемых Microsoft Access.

Документ System

System является стандартным именем документа для всех приложений, работающих в среде Microsoft Windows. Данное имя позволяет получать сведения о других именах документов, поддерживаемых приложением. Для того чтобы получить доступ к этой информации необходимо сначала вызвать функцию DDEInitiate, указав "**System**" в аргументе *документ*, после чего выполнить инструкции DDERequest с одним из следующих значений аргумента *раздел*.

***SysItems*:** список имен разделов, входящих в документ System в Microsoft Access.

***Formats*:** список форматов, в которых Microsoft Access может копировать данные в буфер обмена.

Status : состояние сеанса обмена: "Busy" (занято) или "Ready" (свободно).

Topics: список всех открытых баз данных.

В следующем примере демонстрируется использование функций DDEInitiate и DDERequest с документом System.

```
' В программе Visual Basic открывается сеанс обмена DDE с Microsoft Access.
Dim intChan1 As Integer, strResults As String
intChan1 = DDEInitiate("MSAccess", "System")
' Запрашивается список документов, поддерживаемых документом System.
strResults = DDERequest(intChan1, "SysItems")
' Вызывается макрокоманда OpenDatabase, открывающая базу данных Base.mdb.
' Данный путь следует заменить на реальный путь к базе данных Base.mdb.
DDEExecute intChan1, "[OpenDatabase C:\Access\Samples\Base.mdb]"
```

Документ базаДанных

Документ *базаДанных* представляет имя файла существующей базы данных. Допускается непосредственное указание имени базы данных (например, Base) или указание имени с путем и расширением .mdb (C:\Msoffice\Access\Samples\Base.mdb). После открытия сеанса обмена DDE с базой данных можно затребовать список всех объектов этой базы данных.

Примечание. Не допускается использование протокола DDE для запросов к файлу рабочей группы Microsoft Access.

Документ *базаДанных* поддерживает следующие имена разделов.

TableList - Список таблиц.

QueryList - Список запросов.

FormList - Список форм.

ReportList - Список отчетов.

MacroList - Список макросов.

ModuleList - Список модулей.

В следующем примере демонстрируется открытие в программе Visual Basic формы «Сотрудники» из базы данных «Base» (Base.mdb).

```
' в программе Visual Basic открывается сеанс обмена DDE с базой данных Base.
' База данных должна быть уже открыта.
intChan2 = DDEInitiate("MSAccess", "Base")
' Запрашиваем список форм из базы данных Base.mdb.
strResponse = DDERequest(intChan2, "FormList")
' Для открытия формы «Сотрудники» вызываем макрокоманду ОткрытьФорму (OpenForm) с необходимыми аргументами.
DDEExecute intChan2, "[OpenForm Сотрудники,0,,,1,0]"
```

Документы TABLE имяТаблицы, QUERY имяЗапроса и SQL инструкцияSQL

Для данных документов требуется следующий синтаксис:

имяБазыДанных; **TABLE** и*мяТаблицы*

имяБазыДанных; **QUERY** *имяЗапроса*

имяБазыДанных; **SQL** [*инструкцияSQL*]

Аргументы инструкций означают следующее.

ИмяБазыДанных - имя базы данных, которой принадлежат таблица или запрос, или к которой относится инструкция SQL. После имени базы данных необходимо помещать точку с запятой (;). Допускается непосредственное указание имени базы данных (например, Base) или указание имени с путем и расширением .mdb (C:\Msoffice\Access\Samples\Base.mdb).

ИмяТаблицы - имя существующей таблицы.

ИмяЗапроса - имя существующего запроса.

ИнструкцияSQL - допустимая инструкция SQL длиной до 256 символов, заканчивающаяся точкой с запятой. Для создания инструкции SQL с длиной, превышающей 256 символов, следует опустить данный аргумент и взамен использовать несколько инструкций **DDEPoke**.

Например, в следующей программе Visual Basic инструкция SQL строится с помощью инструкций DDEPoke, после чего запрашиваются результаты данного запроса.

```
intChan1 = DDEInitiate("MSAccess", "Base;SQL")
DDEPoke intChan1, "SQLText", "SELECT *"
DDEPoke intChan1, "SQLText", " FROM Заказы"
DDEPoke intChan1,"SQLText","WHERE[Стоимость доставки] > 100;"
strResponse = DDERequest (intChan1,"NextRow")
DDETerminate intChan1
```

Ниже перечислены допустимые имена разделов для документов **TABLE** *имяТаблицы*, **QUERY** *имяЗапроса* и **SQL** *инструкцияSQL*.

All - все данные из таблицы, включая имена полей.

Data - содержимое строк данных без имен полей.

FieldNames - Одна строка с именами полей.

FieldNames;T - список из двух строк, содержащий имена полей (первая строка) и их типы данных (вторая строка). Например, возвращаемыми значениями и соответствующими им типами данных могут быть:

0 - неверный тип данных.

1 - **True/False** (не пустое **Null** значение)

2 - однобайтовое целое без знака.

3 - двухбайтовое целое со знаком (Integer).

4 - четырехбайтовое целое со знаком (Long).

5 - восьмибайтовое число со знаком (Currency) и т.д.

NextRow - данные из следующей записи таблицы или запроса. При открытии канала NextRow возвращает содержимое первой записи. Если текущая запись является последней, вызов NextRow приводит к ошибке.

PrevRow - данные из предыдущей записи таблицы или запроса. Если PrevRow является первым вызовом на новом канале, возвращается содержимое последней записи таблицы или запроса. Если текущей является первая запись, вызов PrevRow приводит к ошибке.

FirstRow - данные из первой записи таблицы или запроса.

LastRow - данные из последней записи таблицы или запроса.

FieldCount - число полей в таблице или запросе.

SQLText - инструкция SQL, представляющая таблицу или запрос. Для таблиц данный аргумент возвращает инструкцию SQL вида "SELECT * FROM *имяТаблицы*;".

SQLText;n - инструкция SQL, представляющая таблицу или запрос, разбитая на части по n символов, где n является целым числом, не превышающим 256. Предположим, например, что запрос представляется следующей инструкцией SQL:

"SELECT * FROM Заказы;"

Если указать раздел в виде "SQLText;7", то возвращаются следующие части инструкции, разделяемые символами табуляции: "SELECT " "* FROM " "Заказы;"

В следующем примере демонстрируется использование механизма DDE в программе Visual Basic для получения данных из таблицы, содержащихся в данных «Base», и вставки этих данных в текстовый файл.

```
Sub NorthwindDDE
Dim intChan1 As Integer, intChan2 As Integer, intChan3 As Integer
```

```
Dim strResp1 As Variant, strResp2 As Variant, strResp3 As Variant
' В модуле Visual Basic запрашиваем данные из таблицы «Типы»,
' запроса «Каталог» и таблицы «Заказы» в базе данных Base.mdb.
' База данных должна быть уже открыта.
intChan1 = DDEInitiate("MSAccess", "Base;TABLE Типы")
intChan2 = DDEInitiate("MSAccess", "Base;QUERY Каталог")
intChan3 = DDEInitiate("MSAccess", "Base;SQL SELECT * " _
        & "FROM Заказы "&"WHERE КодЗаказа > 10050;")
strResp1 = DDERequest(intChan1, "All")
strResp2 = DDERequest(intChan2, "FieldNames;T")
strResp3 = DDERequest(intChan3, "FieldNames;T")
DDETerminate intChan1
DDETerminate intChan2
DDETerminate intChan3
' Вставляем данные в текстовый файл.
Open "C:\Данные.TXT" For Append As #1
Print #1, strResp1
Print #1, strResp2
Print #1, strResp3
Close #1
End Sub
```

Примечание. При работе с большим количеством данных часто бывает удобно записывать данные в файл или считывать из файла. Чтобы получить возможность выполнить любую операцию ввода/вывода, файл необходимо открыть. Инструкция **Open** резервирует буфер ввода/вывода для файла и определяет режим использования этого буфера.

Инструкция **Open** позволяет напрямую создать файл и получить к нему доступ, используя **три типа** доступа к файлам.

· **Последовательный доступ** (режимы **Input**, **Output** и **Append** режимы), обычно используемый для записи текстовых файлов, например протоколов ошибок или отчетов.

· **Произвольный доступ** (режим **Random**), используемый при необходимости считать и записать данные в файл без его закрытия. Файлы произвольного доступа содержат данные в виде записей, которые упрощают и ускоряют поиск нужных сведений.

· **Двоичный доступ** (режим **Binary**), используется, когда требуется считать или записать байт в любую позицию в файле, например при сохранении или отображении точечных изображений.

Инструкцию **Open** не следует использовать для доступа к собственным типам файлов приложений. Например, не следует использовать **Open** для доступа к документу Word, к электронной таблице Microsoft Excel или к базе данных Microsoft Access, поскольку это вызовет потерю целостности и порчу файла.

Ниже приведены инструкции, обычно используемые для записи данных в файлы и для чтения данных из файлов.

Тип доступа	Запись данных	Чтение данных
Последовательный	Print #, Write #	Input #
Произвольный	Put	Get
Двоичный	Put	Get

Дополнительные сведения по операциям ввода/вывода.

А. Инструкция **Open**

разрешает выполнение с файлом операций ввода/вывода.

Синтаксис:

Open *путь* **For** *режим* [**Access** *доступ*] [*блокировка*] **As** [#]номерФайла [**Len**=длина]

Синтаксис инструкции **Open** содержит следующие элементы:

путь - обязательный. Строковое выражение, указывающее имя файла, может содержать имя каталога или папки и имя диска.

режим - обязательный. Ключевое слово, указывающее режим файла: **Append**, **Binary**, **Input**, **Output** или **Random**.

По умолчанию, файл открывается для доступа в режиме **Random**.

доступ - необязательный. Ключевое слово, указывающее операции, разрешенные с открытым файлом: **Read**, **Write** или **Read Write**.

блокировка - необязательный. Ключевое слово, указывающее операции, разрешенные с открытым файлом другим процессам: **Shared**, **Lock Read**, **Lock Write** и **Lock Read Write**.

номерФайла - обязательный. Допустимый номер файла в интервале от 1 до 511 включительно. Для определения следующего свободного номера файла следует использовать функцию **FreeFile**.

длина - необязательный. Число, меньшее либо равное 32 767 (байт). Для файлов, открытых в режиме **Random**, это значение является длиной записи. Для файлов с последовательным доступом это значение является числом буферизуемых символов.

Если аргумент путь описывает несуществующий файл, такой файл будет создан при открытии в режиме **Append**, **Binary**, **Output** или **Random**.

Если файл уже открыт другим процессом и указанный режим доступа не разрешен, инструкция **Open** приведет к ошибке.

Если аргумент режим имеет значение **Binary**, предложение **Len** игнорируется.

Внимание! В режимах **Binary**, **Input** и **Random** можно еще раз открыть уже открытый файл под другим номером, не закрывая его. В режиме **Append** и **Output** необходимо закрыть файл, чтобы получить возможность открыть его еще раз под другим номером.

Примеры использования инструкции Open.

Ниже показаны различные способы использования инструкции **Open**, чтобы разрешить выполнение операций ввода/вывода с файлом.

Следующие инструкции открывают файл TESTFILE для последовательного чтения.

```
Open "TESTFILE" For Input As #1
' Закрывает файл перед повторным открытием в другом режиме.
Close #1
```

Следующие инструкции открывают файл в режиме Binary только для записи.

```
Open "TESTFILE" For Binary Access Write As #1
' Закрывает файл перед повторным открытием в другом режиме.
Close #1
```

Следующие инструкции открывают файл в режиме **Random**. Файл содержит записи определенного пользователем типа **Record**.

```
Type Record   ' Тип, определенный пользователем.
    ID As Integer
    Name As String * 20
End Type
Dim MyRecord As Record    ' Объявляет переменную.
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
' Закрывает файл перед повторным открытием в другом режиме.
Close #1
```

Следующая инструкция открывает файл для последовательного вывода; любой процесс может читать из этого файла или записывать в него.

```
Open "TESTFILE" For Output Shared As #1
' Закрывает файл перед повторным открытием в другом режиме.
Close #1
```

Следующая инструкция открывает файл в режиме **Binary** для чтения; другие процессы не могут читать этот файл.

```
Open "TESTFILE" For Binary Access Read Lock Read As #1
```

В. Инструкция **Print #**

Записывает отформатированные данные в файл с последовательным доступом.

Синтаксис:

Print #*номерФайла*, [*списокВывода*]

Синтаксис инструкции **Print #** содержит следующие элементы:

номерФайла - обязательный. Любой допустимый номер файла.

списокВывода - необязательный. Выражение или список выражений, которые следует напечатать.

Ниже приведены допустимые значения аргумента *списокВывода*:

[{**Spc(***n***)** | **Tab**[**(***n***)**]}] [*выражение]* [*позиция*]

Spc(*n***)** - используется для вставки пробелов в файл; здесь *n* число пробелов, которые следует вставить.

Tab(*n***)** - устанавливает курсор в столбец с указанным номером; здесь *n* номер столбца. **Tab** без аргумента устанавливает курсор в начало следующей зоны печати.

выражение - числовые выражения или строковые выражения, которые следует напечатать.

позиция - указывает позицию, в которой следует печатать следующий символ. Для установки курсора сразу после последнего напечатанного символа используйте точку с запятой. Для установки курсора в столбец с указанным номером используйте **Tab(***n***)**. Для установки курсора в начало следующей зоны печати используйте **Tab** без аргумента. Если аргумент *позиция* опущен, следующий символ печатается на следующей строке.

Данные, записанные с помощью инструкции **Print #**, обычно считываются из файла с помощью инструкций **Line Input #** или **Input**. Если аргумент *списокВывода* опущен, после аргумента *номерФайла* идет только разделитель списка, в файл печатается

пустая строка. Для разделения выражений можно использовать пробелы или точки с запятой, которые в данной ситуации полностью эквивалентны.

При выводе логических данных (тип **Boolean**) в файл записываются слова **True** или **False**.

При выводе данных типа **Date** используется текущий краткий системный формат даты. Если компонент, описывающий дату или время, отсутствует или равен нулю, в файл записывается только имеющийся в наличии компонент.

Если *списокВывода* имеет значение **Empty**, в файл ничего не записывается. Однако если *списокВывода* имеет значение **Null**, в файл записывается ключевое слово **Null**.

Данные типа **Error** записываются в файл как **Error** *кодОшибки*.

Инструкция **Print #** записывает в файл данные, отформатированные с учетом национальной настройки. Это означает, в частности, что используется соответствующий разделитель целой и дробной части числа.

Поскольку инструкция **Print #** записывает в файл отформатированные данные, необходимо использовать разделители данных, обеспечивающие правильную печать. Если **Tab** без аргумента используется для перемещения позиции вывода в начало следующей зоны печати, **Print #** также записывает пробелы между полями печати в файле.

Примечание. Для записи в файл данных, который в будущем планируется читать с помощью инструкции **Input #**, следует вместо инструкции **Print #** использовать инструкцию **Write #**. Использование инструкции **Write #** гарантирует, что записанные данные будут корректно разделены, что позволит прочитать их с помощью инструкции **Input #**.

C. Инструкция Line Input #

Читает строку из открытого последовательного файла и присваивает ее переменной типа **String**.

Синтаксис:

Line Input #*номерФайла*, *имяПеременной*

Синтаксис инструкции **Line Input #** содержит следующие элементы:

номерФайла - обязательный. Любой допустимый номер файла.

имяПеременной - обязательный. Допустимое имя переменной типа **Variant** или **String**.

Данные, считываемые с помощью инструкции **Line Input #**, обычно записываются в файл с помощью инструкции **Print #**.

Инструкция **Line Input #** последовательно считывает из файла по одному символу до тех пор, пока не встретит символ возврата каретки (**Chr**(13)) или комбинацию символов возврата каретки и перевода строки (**Chr**(13) + **Chr**(10)). Когда считанная строка присваивается переменной, символы возврата каретки и конца строки отбрасываются.

Пример использования инструкции Line Input #

В данном примере инструкция **Line Input #** используется для чтения строки из последовательного файла и присвоения ее переменной. Предположим, что текстовый файл TESTFILE существует и содержит несколько строк текста.

```
Dim TextLine
Open "TESTFILE" For Input As #1  ' Открывает файл.
Do While Not EOF(1)                    ' Цикл до конца файла.
        Line Input #1, TextLine        ' Читает строку в переменную.
        Debug.Print TextLine           ' Выводит в окно "Отладка".
Loop
Close #1                               ' Закрывает файл.
```