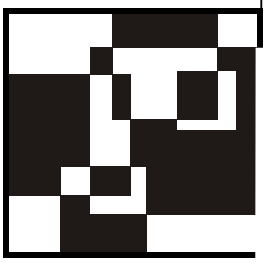
ГЛАВА 12



Программирование в Access

12.1. Отличительные особенности создания приложений в Access

Программирование в Access сильно отличается от программирования в Word, Excel и других приложениях Office. Главное принципиальное отличие заключается в том, что Word, Excel, PowerPoint, Project и т. п. предназначены, прежде всего, для непосредственной работы пользователя, без какой-либо их доработки со стороны разработчиков предприятия. Access иногда используется пользователями как конечное приложение, но чаще он все-таки применяется как платформа для создания своих приложений разработчиками.

Второе отличие заключается в том, что в Access встроено свое собственное ядро для работы с данными. Фактически Access — это полноценная система управления базами данных, поэтому для полного использования его возможностей необходимы знания о принципах работы с базами данных: что такое таблицы и отношения между ними (система ключей), что такое нормализация данных, типы данных, ограничения целостности и т. п. Очень часто пользователи на предприятиях такими знаниями не обладают.

Кроме того, существуют разные варианты использования Access с точки зрения архитектуры приложения. Иногда Access (файл MDB) используется просто как ядро, которое управляет данными, находящимися в таблицах. Пользователи работают с этими данными через внешние приложения, созданные разработчиками, например, на Visual Basic, Delphi или C++. В других ситуациях Access, наоборот, используется только для предоставления пользовательского интерфейса для работы с данными, которые физически расположены на серверах баз данных, например, SQL Server, Oracle, IBM D2 и т. п.

В Access предусмотрен встроенный язык запросов JET SQL, который активно используется разработчиками для работы с данными в базах Access.

Подводя итоги, можно сказать, что программирование средствами VBA в Access, которое будет рассмотрено в этой главе, — это лишь малая часть возможностей этого приложения. Очень многие возможности работы с Access (например, его язык запросов или проектирование и создание таблиц) останутся за пределами этой книги. Для этого существует отдельная литература. Заметим, что во многих книгах по Access за рамками остается как раз язык VBA и объектная модель самого Access, так что эта глава может послужить хорошим дополнением к ним.

Какие задачи на предприятии чаще всего решаются средствами автоматизации в Access?

Сразу же скажем, что поскольку Access — это система управления базами данных, то он очень часто используется как контейнер для хранения данных. При этом данные могут быть самыми разными, например, обычные данные о заключенных договорах или клиентах вашего предприятия, цифровые фотографии, шаблоны Word и Excel, которые используются для генерации отчетов из баз данных. В Access все эти данные вместе с графическим интерфейсом можно "упаковать" в один MDB-файл, что позволяет сделать приложение очень компактным и удобным для переноса с одного компьютера на другой.

Еще одно важное назначение Access — обеспечение клиентского интерфейса для работы с данными, которые хранятся на мощных клиент-серверных системах, таких как SQL Server, Oracle, IBM DB2. Согласно Microsoft, рекомендуется использовать настольные системы (такие как Access, FoxPro, Paradox и т. п.), если к данным одновременно будут обращаться не более 10 пользователей. Если пользователей больше или самих данных очень много (несколько гигабайт), то рекомендуется использовать более сложные, но и более функциональные клиент-серверные системы. А уже в рамках обеспечения доступа к данным (на клиент-серверных системах или прямо в базах данных Access) решаются более узкоспециализированные задачи приложений:

- *создание обычных форм*, т. е. формирование программных интерфейсов для занесения, изменения или просмотра данных в базе и Web-форм (они называются страницами доступа к данным);
- *создание отчетов к базам данных*, в том числе параметризованных;
- *создание программной логики приложения* обычным способом — на VBA (модули) и для начинающих пользователей (макросы, которые всегда можно преобразовать в модули);
- *вспомогательные действия* — печать, экспорт и преобразование данных (хотя для преобразования данных обычно удобнее использовать объектную модель DTS), загрузка данных, репликация и т. п.

12.2. Основные этапы создания приложений в Access

Если вы будете выступать в роли профессионального разработчика и создавать приложение, работающее с информацией на источнике данных (в базе данных Access или на клиент-серверной системе), то имеет смысл последовательно выполнить все рекомендованные далее этапы проектирования и создания таких приложений. Этим вы, возможно, убережете себя от ошибок, которые будет впоследствии трудно исправить.

- Первый этап — *сбор информации о потребностях предприятия*, *его подразделений и пользователей*: с какими унаследованными системами они работают, с чем нужно обеспечивать совместимость, как организованы информационные потоки, какова изменчивость системы и т. п. Часто при этом используются специальные программные средства, такие как Microsoft Visio и ERWin.
- Второй этап — *выбор архитектуры приложения*, *выбор подходящей системы управления базами данных и проектирование СУБД*. На этом этапе определяется, будет ли информация храниться с использованием ядра Jet, на котором работает Access, или клиент-серверной системы, проектируется система таблиц для хранения информации и отношений между ними, проектируются некоторые другие объекты базы данных. Кроме того, определяется архитектура приложения — сколько в ней будет уровней, будут ли использоваться терминальные или Web-технологии, будет ли применяться репликация и т. п.
- Третий этап — *реализация СУБД и бизнес-логики приложения*. На этом этапе проектируются, создаются, настраиваются и заполняются исходными данными объекты базы данных: таблицы, представления, хранимые процедуры, формы, отчеты, макросы, программные модули и т. п. При создании приложений в Access бóльшая часть этих операций выполняется при помощи графического интерфейса разработчика. Код VBA используется для проверки вводимых пользователем значений, для работы с элементами управления на форме, для переключения между формами, отчетами, другими элементами управления, для обращения к внешним объектным моделям и т. п. На этом этапе опять-таки могут помочь Visio и ERWin.
- Четвертый этап — *оптимизация производительности базы данных*, что часто упускается разработчиками. Задача эта комплексная, но включает в себя в том числе и оптимизацию кода VBA.
- Пятый этап — *тестирование и отладка приложения (см. гл. 6)*.
- Шестой этап — *развертывание приложения*.

Как мы видим, эти этапы достаточно комплексные, и рассмотреть их все в этой книге не представляется возможным. Поэтому в данной главе будет рассказано только про те моменты, в которых нам может помочь VBA.

12.3. Объект *Application*, его свойства и методы

Объектная модель Access по своей архитектуре сильно отличается от объектных моделей Word и Excel. Возможно, это объясняется тем, что Access — не "родной" для Microsoft, как Word и Excel, а приобретенный продукт третьей фирмы.

Один из немногих моментов, в котором программирование в Access похоже на программирование в Word и Excel — это наличие объекта `Application`, который находится на вершине иерархии объектной модели Access. Он точно так же может использоваться для программного запуска Access из других приложений, и его свойства и методы доступны из любой части кода. Запуск Access из другого приложения может выглядеть так:

```
Dim appAccess As Object
Set appAccess = CreateObject("Access.Application")
appAccess.Visible = True
MsgBox appAccess.Name
```

Если запустить такой код из процедуры Word или Excel, то по умолчанию по завершении работы этой процедуры объект будет удален из оперативной памяти (поэтому здесь и поставлено окно сообщения, чтобы этот процесс задержать).

Можно программно запускать Access еще множеством разных способов — через объектную модель Windows Explorer, через командный интерпретатор операционной системы, через текстовый ярлык mad, через API и т. п.

На практике программным образом запускать Access приходится достаточно редко, поскольку обычно удобнее оболочку приложения, запускающего Word, Excel и т. п., делать именно в Access. Открывать программным способом Access для доступа к данным базы в MDB-файле не рекомендуется, для этого лучше использовать объекты ADO, более простые и менее ресурсоемкие.

Теперь поговорим о свойствах и методах объекта `Application`. Как можно убедиться, их набор в Access мало похож на соответствующий набор в Word и Excel.

- `AutomationSecurity` — это свойство позволяет определить уровень безопасности при открытии базы данных. По умолчанию используется значе-

ние msoAutomationSecurityByUI — использовать то, что настроено на графическом экране через меню **Макрос | Безопасность**. Можно вообще запретить открытие файлов баз данных с макросами (msoAutomationSecurityForceDisable), но чаще используется значение msoAutomationSecurityLow, которое позволяет открыть файл данных без лишних вопросов.

- BrokenReference — позволяет проверить, есть ли разорванные ссылки (когда ваше приложение не может найти модуль dll или другую базу данных). Наличие конкретной ссылки можно проверить при помощи объекта Reference.
- CodeContextObject — очень полезное свойство, которое позволяет определить, из какого объекта базы данных (формы, отчета и т. п.) был запущен модуль или макрос. Это свойство можно, например, использовать для обнаружения источника ошибок.
- CodeData — еще одно важнейшее свойство. Оно позволяет получить доступ к коллекциям AllDatabaseDiagrams, AllFunctions, AllQueries, AllStoredProcedures, AllTables и AllViews. Правда, в этих коллекциях находятся одни и те же объекты AccessObject. На первый взгляд возможностей у них не так много, но на самом деле при помощи этого объекта мы можем настраивать десятки свойств для таблиц, запросов, диаграмм и других объектов базы данных Excel. Пример использования свойства CodeData для получения информации о всех таблицах в базе данных может выглядеть так:

```
For Each oTable In CodeData.AllTables
    Debug.Print oTable.Name
Next
```

- CodeProject — используется для тех же целей, что и CodeData, но предоставляет доступ к коллекциям программных модулей AllForms, AllReports, AllMacros, AllModules и AllDataAccessPages.
- CommandBars — возвращает коллекцию объектов CommandBar, т. е. панелей инструментов Access. Эту коллекцию можно использовать для настройки пользовательского интерфейса приложения, построенного на основе Access.
- CurrentData — действует аналогично CodeData и CodeProject. Это свойство позволяет получить доступ к тем же коллекциям, включая полученные с внешнего источника данных (SQL Server). Аналогично работает свойство CurrentProject.
- CurrentObjectName и CurrentObjectType — позволяют определить, какой объект на момент запуска процедуры находился в фокусе (из какого объ-

екта был вызван данный код). Эти свойства, конечно, удобно использовать для проверок, когда один и тот же код может быть вызван разными способами. При этом свойство `CurrentObjectType` ведет себя несколько неожиданно. В обычных ситуациях оно возвращает значение из перечисления, но если вызвавшего объекта уже нет (объектная ссылка установлена в `Nothing`) или информацию о нем получить не удалось, это свойство почему-то возвращает `True`.

- `DataAccessPages` — позволяет получить ссылку на одноименную коллекцию, в которой находятся объекты всех Web-форм базы данных (они называются страницами доступа к данным) — объектов `DataAccessPage`.
- `DBEngine` — позволяет получить ссылку на объект `DBEngine`, при помощи которого можно просмотреть или настроить свойства ядра Jet, на котором работает Access. Например, при помощи этого свойства можно сжать базу данных, настроить для нее пароль, определить используемую кодировку и т. п.
- `DoCmd` — позволяет получить доступ к еще одному очень важному объекту — `DoCmd`, при помощи которого можно выполнить множество важных операций. Фактически этот объект — основная "рабочая лошадка" Access с точки зрения VBA. Он будет рассмотрен в *разд. 12.4*. Сам объект `DoCmd` создавать нет необходимости — он всегда доступен через свойство объекта `Application`, например:

  ```
  Application.DoCmd.RunSQL "Delete from table1"
  ```

- `Forms` — позволяет вернуть ссылку на коллекцию объектов `Form`. От уже упомянутой коллекции `AllForms` эта коллекция отличается двумя особенностями:
  - в ней находятся только открытые в настоящий момент формы;
  - в ней находятся не объекты `AccessObject`, как в `AllForms`, а объекты `Form` с гораздо более богатым набором свойств и методов.
- `MenuBar` — позволяет не совсем стандартным образом настроить пользовательское меню (уровня всего приложения), вернуться к показу встроенного меню или вообще отключить показ меню. Для работы с контекстным меню предусмотрено свойство `ShortcutMenuBar`.
- `Modules` — действует аналогично свойству `Forms`. Оно позволяет получить доступ к одноименной коллекции с объектами `Module`, представляющими стандартные модули и модули классов. У объектов `Module` намного больше свойств и методов, чем у объекта `AccessObject`.
- `Printers` и `Printer` — возвращают соответственно коллекцию всех установленных в системе принтеров и принтер по умолчанию. При помощи

объекта Printer можно настроить множество свойств принтера, которые будут использоваться при печати.

- References — позволяет получить ссылку на коллекцию объектов Reference (ссылки на библиотеки типов). Можно использовать для проверки работоспособности ссылок или для добавления ссылок программным способом.
- Reports — работает аналогично свойствам Forms и Modules, позволяя получить доступ к коллекции объектов Report.
- Screen — это свойство возвращает специальный объект Screen, при помощи которого можно программным образом обратиться к активным элементам Access: формам, отчетам, элементам управления и т. п. (в Word и Excel свойства с префиксом Active... встроены непосредственно в объект Application). При помощи этого объекта можно также изменить вид курсора мыши и вернуться к предыдущему элементу управления.
- UserControl — позволяет определить, как именно был запущен Access: вручную пользователем или программным способом из другого приложения (в зависимости от этого, к примеру, можно решить, закрывать Access автоматически или положиться на пользователя).
- Version — позволяет вернуть версию Access (например, для дополнительных проверок работоспособности приложений). Для Access 2003 эта версия — 11.0.
- Visible — позволяет сделать Access видимым для пользователя или, наоборот, спрятать. По умолчанию Accesss, запущенный программным способом, невидим для пользователя, и это свойство необходимо установить в True.

Методов у объекта Application также очень много (кроме того, часть методов искусственно перенесена в объект DoCmd). Далее представлены самые важные из них.

- AccessError() — очень важный метод для обработки ошибок. Он позволяет получить описание ошибок библиотек Access и DAO, для которых стандартное Err.Description возвращает "Application-defined or object-defined error".
- BuildCriteria() — позволяет очень быстро и удобно сконструировать критерий отбора записей, который может применяться в SQL-запросах, фильтрах для формы и отчетов, и т. п. Возвращает правильно сконструированное строковое значение.
- CloseCurrentDatabase() — закрывает текущую базу данных без закрытия Access. Обычно применяется для того, чтобы затем открыть другую базу данных без запуска нового экземпляра Access.

- `CodeDb()` — возвращает объект `DAO.Database`, представляющий базу данных, в которой в настоящее время выполняется код (обычно используется, когда у вас есть специальная библиотечная база данных с программными модулями, выполняющими различные операции с другими базами данных). Ссылку на такой же объект для текущей базы данных можно получить при помощи метода `CurrentDb()`.
- `CompactRepair()` — позволяет сжать или починить базу данных Access и вернуть код ошибки (можно также записать протокол). Сжимаемая или ремонтируемая база данных должна быть в это время закрыта.
- `ConvertAccessProject()` — позволяет выполнить еще одну служебную операцию, на этот раз по преобразованию версии базы данных Access. Возможны варианты от `acFileFormatAccess2` до `acFileFormatAccess2002`.
- `CreateAccessProject()` — позволяет программным образом создать проект Access (так называется программный интерфейс для доступа к SQL Server). Для того чтобы его сразу создать и открыть, можно использовать метод `NewAccessProject()`, а чтобы просто открыть существующий — `OpenAccessProject()`.
- `CreateAdditionalData()` — позволяет создать объект `AdditionalData`, который можно использовать вместе с методом `ExportXML()` при экспорте родительской таблицы в XML-файл. Применение этого объекта позволяет экспортировать набор таблиц.
- `CreateControl()` — позволяет программным образом создать элемент управления на форме. Принимает множество параметров, которые определяют данный элемент управления. Для создания элемента управления в отчете используется метод `CreateReportControl()`. Удалить элемент управления можно при помощи соответственно `DeleteControl()` и `DeleteReportControl()`.
- `CreateForm()` — позволяет также программным образом создать форму Access и получить ссылку на объект созданной формы. Затем можно настроить свойства этой формы, добавить для нее элементы управления и т. п. Создание таким же образом отчета можно произвести при помощи метода `CreateReport()`.
- `CreateGroupLevel()` — позволяет программным образом создать группировку в отчете или отсортировать записи. Принимает имя отчета, столбец, по которому производится сортировка, формулы для создания верхнего и нижнего колонтитула групп.
- `CreateNewWorkgroupFile()` — позволяет программным способом создать файл рабочей группы с разрешениями для пользователей. На графическом

экране эту операцию можно выполнить при помощи пункта меню **Сервис | Защита | Администратор рабочих групп**.

- `CurrentUser()` — этот метод позволяет получить в виде строкового значения имя текущего пользователя базы данных. По умолчанию работа производится от имени пользователя Admin.
- Методы с префиксом `D...` — очень удобны для выполнения различных операций, не прибегая к коду SQL, напрямую из Access:
  - `DAvg()`, `DSum()`, `DCount()`, `DMax()`, `DMin()` и т. п. — позволяют применить агрегатные функции к столбцу (или набору записей) в таблице или представлении;
  - `DLookup()` — исключительно удобный метод, который позволяет найти и вернуть нужное вам значение из таблицы или представления (включая двоичные объекты, например шаблоны Word). Принимает в качестве параметров имя таблицы или представления, имя столбца и фильтр. Если условию фильтра удовлетворяет несколько значений, то возвращается первое из них;
  - `DFirst()` и `DLast()` — несмотря на свои названия, эти методы работают одинаково, возвращая случайное значение из столбца таблицы или представления.
- `Echo()` — позволяет перерисовать экран Access, а также вывести текст в строку состояния.
- `Eval()` — этот метод очень удобен во многих ситуациях. Он позволяет произвести над текстовой строкой операции, как будто это строка кода VBA. Этот метод возвращает значение типа `Variant`, чтобы подходили любые возвращаемые значения. Например:

  ```
  Eval("1 + 1")
  ```

  вернет 2, а вызов:

  ```
  Eval("Моя_Функция()")
  ```

  вернет то, что возвращает эта функция. `Eval()` очень удобно использовать, чтобы избежать громоздких проверок и преобразований типов, например, когда мы принимаем разные значения, вводимые пользователем.
- `ExportXML()` и `ImportXML()` — позволяют экспортировать и импортировать наборы таблиц с данными (включая информацию о ключах, индексах, кодировках и т. п.) в XML-совместимый текстовый файл. При этом экспорт и импорт из Access при помощи этих методов можно производить не только для баз данных Access, но и баз данных SQL Server, начиная с версии 6.5.

- `GetOptions()` и `SetOptions()` — позволяют получить информацию и установить те настройки, которые доступны через меню **Сервис | Параметры**. Например, чтобы при нажатии клавиши <Enter> в таблице производился переход не вправо (по умолчанию), а вниз, можно использовать код:

```
Application.SetOption "Move After Enter", 2
```

- `hWndAccessApp()` — очень нужный метод для тех, кто работает с Windows API. Позволяет вернуть указатель на окно Access.
- `NewCurrentDatabase()` — позволяет создать и сразу открыть новую базу данных Access. Для открытия существующей базы данных можно использовать метод `OpenCurrentDatabase()`.
- `Nz()` — исключительно удобный метод для практической работы. Позволяет возвращать пустую строку или другое значение, если значение в данном столбце таблицы не определено (`Null`). Опытные разработчики очень часто используют эту функцию, чтобы избежать ошибок при обращении к пустым значениям (любым, включая `Memo`), например, при поиске по таблице.
- `Quit()` — закрывает Access. Может ничего не сохранять, сохранять все или спрашивать у пользователя.
- `RefreshDatabaseWindow()` — позволяет обновить окно базы данных. Обычно применяется после программного создания форм, отчетов и т. п.
- `Run()` — позволяет вызвать процедуру или функцию VBA из кода и передать ей до 30 параметров. Может использоваться для вызова пользовательских или встроенных функций, но поскольку их можно вызвать и стандартными способами, то чаще всего этот метод используется при вызове процедуры Access из внешней откомпилированной программы, например, DLL или EXE.
- `RunCommand()` — позволяет выполнить одну из десятков встроенных команд Access (практически все, что есть на панелях управления и во встроенных меню). Например, чтобы максимизировать окно Access, можно воспользоваться командой:

```
RunCommand acCmdAppMaximize
```

- `SysCmd()` — позволяет выполнить множество служебных операций: получить информацию о домашнем каталоге, о версии Access, о состоянии указанного вами объекта базы данных, запустить графический индикатор выполнения в строке состояния и т. п.

12.4. Макрокоманды и объект *DoCmd*

Объект `DoCmd` — это "рабочая лошадка" программирования в Access. Этот объект позволяет программным образом выполнять макрокоманды Access — те действия *(actions)*, которые можно просмотреть (на русском языке) в окне конструктора макрокоманд. Действия — это самые распространенные операции, которые обычно приходится выполнять в Access программным способом.

У объекта `DoCmd` нет свойств, только методы. Для целей унификации в последних версиях Access методы `DoCmd` "переезжают" в объект `Application`, но для совместимости со старыми приложениями они оставлены и в `DoCmd`. Microsoft рекомендует по возможности пользоваться одноименными методами объекта `Application`.

Приводить здесь методы `DoCmd` с комментариями нет никакого смысла — эти методы в точности соответствуют набору действий в конструкторе макрокоманд (макрокоманд чуть больше за счет того, что для некоторых из них — окно сообщений, запуск внешнего приложения, передача нажатий клавиш и т. п. — предусмотрены отдельные средства VBA).

Подробно описывать макрокоманды мы не будем: для каждой из них предусмотрено описание на русском языке и подробная справка на английском по нажатию клавиши <F1>. Просмотреть такое описание можно из окна создания макросов (для этого в окне базы данных нужно перейти на вкладку **Макросы** и нажать кнопку **Создать**. Писать код вручную для них также нет никакого смысла — всегда есть возможность преобразовать созданный макрос в модуль и просмотреть полученный код. Далее перечислены лишь основные возможности макрокоманд (методов объекта `DoCmd`).

- `OutputTo()` (соответствует макрокоманде `ВывестиВФормате`), `TransferText()` (`ПреобразоватьТекст`), `TransferDatabase()` (`ПреобразоватьБазуДанных`), `TransferSpreadsheet()` (`ПреобразоватьЭлектроннуюТаблицу`) — макрокоманды, которые обеспечивают экспорт и импорт данных (в формат Excel, RTF, SNP, TXT, DBF, с источниками данных ODBC и т. п.). У каждого из форматов есть свои особенности и недостатки. Можно использовать и рассмотренные нами ранее средства Word и Excel.
- `RunSQL()` (`ЗапускЗапросаSQL`), `RunMacro()` (`ЗапускМакроса`) — позволяют выполнить запрос на языке SQL или макрос соответственно.
- Методы с префиксом `Open...()` (макрокоманды с префиксом `Открыть...` — `Таблицу`, `Запрос`, `Представление`, `Форму` и т. п.) — их действия также понятны из названий. Можно выбрать режим открытия (конструктор, просмотр и т. п.) и многие другие параметры. После открытия объекта можно воспользоваться его кодом и его элементами управления.

Есть возможность также копировать базу данных и отдельные файлы, искать записи, активизировать элементы управления и выполнять множество других операций.

12.5. Работа с формами Access из VBA (объект *Form*)

Один из важнейших элементов Access, который широко используется в приложениях, — это формы. Формы Access предназначены для того же, для чего и обычные формы VBA — это прежде всего контейнеры для графических элементов управления. Но устройство форм Access, их функциональные возможности, приемы работы с ними и даже наборы элементов управления, которые на них можно размещать, сильно отличаются от привычных нам форм VBA, которые можно использовать в Word и Excel.

Формы Access используются:

1. *Для редактирования записей в таблицах базы данных Access и внешних источников данных*. Для того чтобы создать такие формы, вообще не нужно никакого программирования — достаточно создать форму в режиме конструктора или воспользоваться мастером создания форм. Подключиться к внешнему источнику данных (например, к базе данных SQL Server или Oracle) можно, воспользовавшись в Access меню **Файл | Внешние данные | Связь с таблицами**.
2. *Как панели управления вашего приложения.* Очень часто в приложении на основе Access создается начальная форма, которая открывается при запуске этого приложения. На этой формы предусмотрены кнопки и другие элементы управления для вызова других форм, отчетов, макросов, выхода из приложения и выполнения прочих операций. После закрытия других форм управление опять передается начальной форме.
3. *Просто для предоставления пользователю возможностей для выполнения любых действий.* Например, форму можно использовать для выбора пользователем параметров отчета, выгрузки данных во внешнее приложение (например, Excel) и т. п.

Как работать с формами Access из VBA?

Первое, что необходимо сказать — для работы с формами во многих ситуациях нам придется использовать общий объект `AccessObject`, который представляет в Access не только формы, но и таблицы, макросы, модули, отчеты и множество других элементов. Поскольку этот объект универсальный, то, конечно, большой помощи от подсказки в редакторе VBA у нас не будет. Обратиться к объекту формы можно через коллекцию `AllForms`, которая доступна

через объекты CodeProject и CurrentProject. Например, получить информацию о всех формах в базе данных Access можно так:

```
Dim oA As AccessObject
For Each oA In CurrentProject.AllForms
    Debug.Print oA.Name
Next
```

Если вы будете обращаться к формам в коллекции AllForms по индексу, обратите внимание, что нумерация форм в этой коллекции начинается с 0. Обращаться к элементам в этой коллекции можно и по имени:

```
Debug.Print CurrentProject.AllForms("Форма1").IsLoaded
```

Специальное свойство IsLoaded определяет, открыта ли эта форма (т. е. загружена ли она в оперативную память).

Программно формы можно найти и другим способом. Все открытые формы Access автоматически помещаются в коллекцию Application.Forms и представляются в виде объекта Form. Это уже нормальный объект, свойства которого соответствуют свойствам формы, доступным через графический интерфейс. Например, если форма Форма1 открыта, получить информацию о ее ширине можно так:

```
Debug.Print Application.Forms("Форма1").Width
```

Это свойство можно использовать и для изменения ширины формы, но для этой цели рекомендуется использовать метод DoCmd.MoveSize(), который изменяет размеры активного объекта (например, нашей формы, если она активна):

```
DoCmd.MoveSize Width:=10000
```

Еще одна возможность: если вы работаете с кодом самой формы или ее элементов управления (например, внутри события Click кнопки, которая расположена на форме), то обратиться к объекту самой этой формы можно просто, используя ключевое слово Form.

Как открыть форму?

Если в Word или Excel нам обязательно нужно открывать форму программным способом, то в Access это делать совсем необязательно. Можно открыть форму и вручную из окна базы данных (рис. 12.1). Из этого же окна обычно производится создание новых форм или изменение уже существующих.

Еще один часто используемый способ — это просто запустить форму при открытии базы данных Access. Для этого в меню **Сервис** нужно выбрать пункт **Параметры запуска** и выбрать нужную форму в списке **Вывод формы/страницы**. Если при этом вы уберете все остальные флажки, то приложение при открытии может выглядеть так, как показано на рис. 12.2.

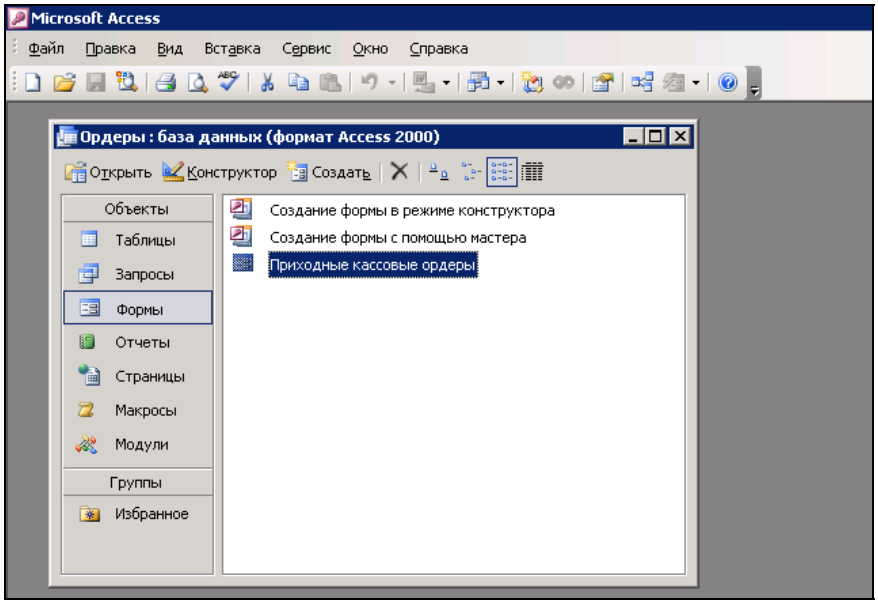


Рис. 12.1. Окно базы данных

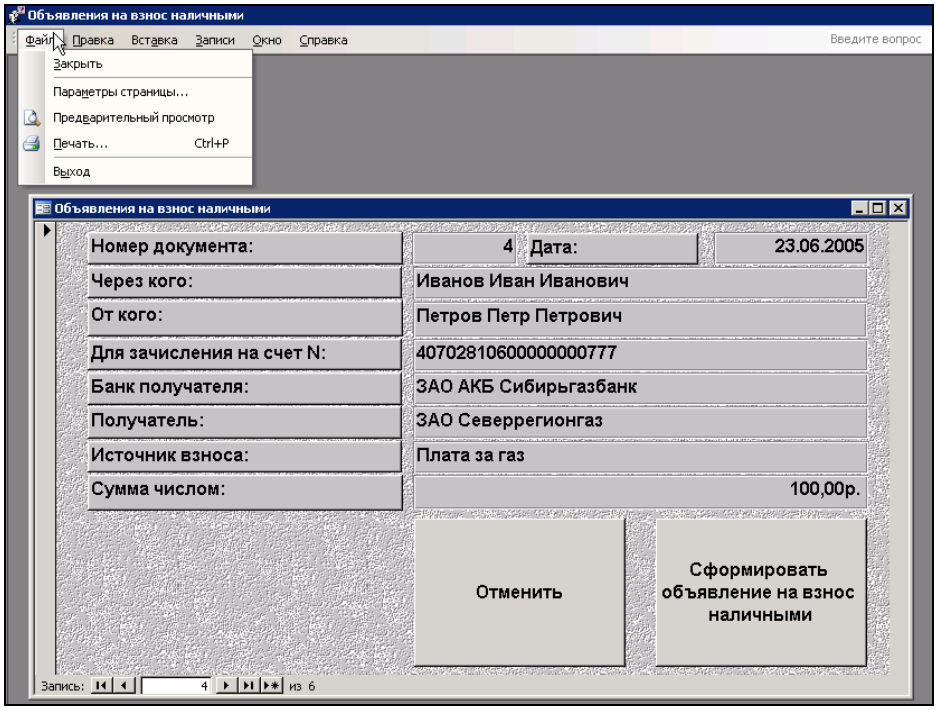


Рис. 12.2. При открытии Access пользователь видит только одну форму

Окно базы данных будет скрыто от пользователя, а это значит, что пользователь не сможет получить информацию ни о таблицах нашей базы данных, ни о других формах, ни о модулях — все служебные элементы базы данных будут от него спрятаны. В принципе, пользователь может обойти такую защиту, держа нажатой при запуске Access клавишу <Shift>, но программным способом можно закрыть и такую возможность.

Если все-таки нужно открыть форму программно (например, из другой формы), то для этой цели можно использовать метод `DoCmd.OpenForm()`. В самом простом варианте этот метод просто принимает параметр с именем формы:

```
DoCmd.OpenForm "Форма1"
```

Если форма уже открыта, то этот метод не открывает ее заново, а просто активизирует. Метод `DoCmd.OpenForm()` принимает также несколько необязательных параметров, при помощи которых вы можете настроить фильтр на отображение записей в форме, режим открытия формы (например, модальность) и т. п. Закрытие формы производится при помощи метода `DoCmd.Close()`. Если же вам нужно просто спрятать форму, чтобы сохранить введенные на ней пользователем значения и отобразить их при следующем показе, можно просто сделать форму невидимой, назначив ее свойству `Visible` значение `False`.

Форма нам обычно нужна как контейнер для расположенных на ней элементов управления. Обычно элементы управления программным способом создавать не нужно, намного проще и удобнее поместить их на форму в режиме конструктора. В наборе элементов управления для формы предусмотрены как знакомые нам элементы управления (текстовые поля, надписи, кнопки, флажки и переключатели), так и новые (свободная и присоединенная рамки объектов, разрывы страниц, подчиненные формы/отчеты и т. п.). В верхнем правом углу **Панели инструментов** в конструкторе формы Microsoft Access находится специальная кнопка **Мастера**. Если она нажата, то добавление на форму привычных элементов управления (например, кнопки) приведет к появлению окна мастера, который попытается помочь вам автоматически сгенерировать нужный код VBA для этого элемента управления (рис. 12.3).

Можно использовать генерируемый мастером код как заменитель макрорекордера (которого в Access нет), чтобы понять, как можно выполнить те или иные действия.

В Access добавлены нестандартные (по отношению к обычным формам VBA) элементы управления.

- **Свободная рамка объекта** — предоставляет возможность разместить на форме OLE-объект (например, документ Word, лист Excel, презентацию PowerPoint, рисунок, звукозапись или видеоклип), который может быть

встроен в базу данных Access (но не помещен в таблицу) или находиться во внешнем по отношению к базе данных Access файлу.

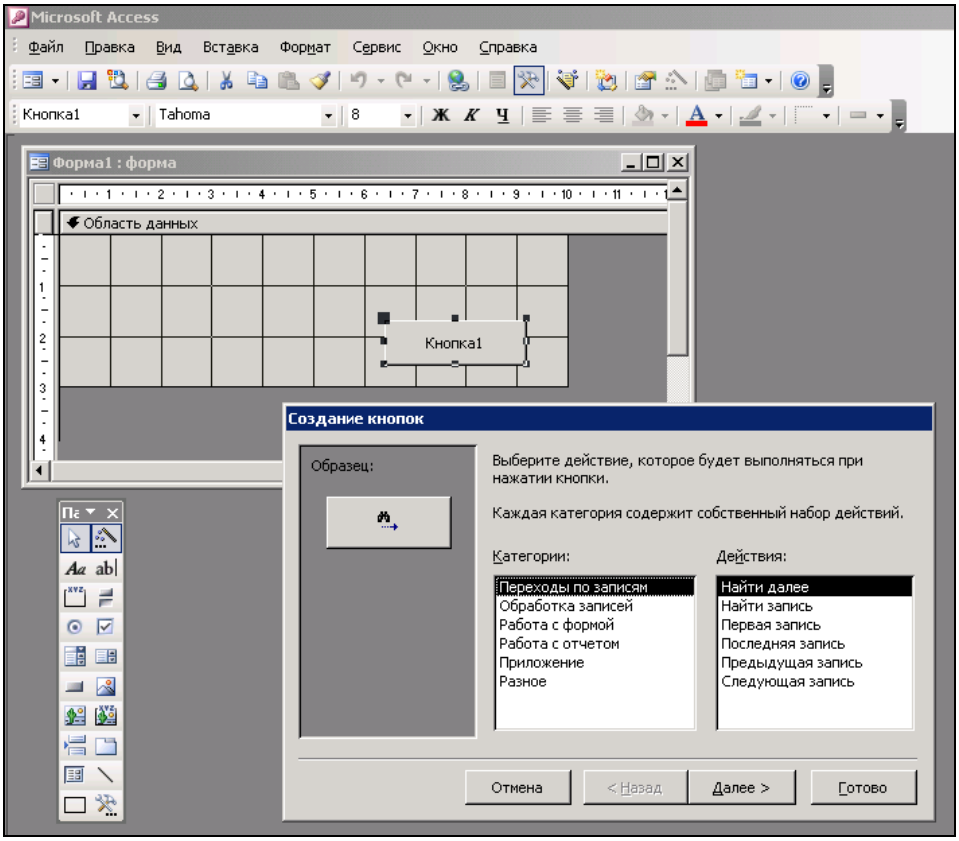


Рис. 12.3. Мастер создания кнопок

- **Присоединенная рамка объекта** — то же самое, за исключением того, что этот элемент применяется для работы с объектами OLE, которые хранятся в таблицах внутри баз данных Access или внешнего источника данных. Это самый удобный способ генерации отчетов в Word.

  Предположим, что в нашей базе данных Access находится таблица с тремя столбцами, как показано на рис. 12.4.

  В столбце **File** у нас хранятся шаблоны Word, которые используются для генерации отчетов. Мы помещаем на форму элемент управления **Присоединенная рамка объекта** с именем `WordTemplate`. После этого все, что нужно для создания файла Word на основе шаблона из базы данных, у нас уже есть.

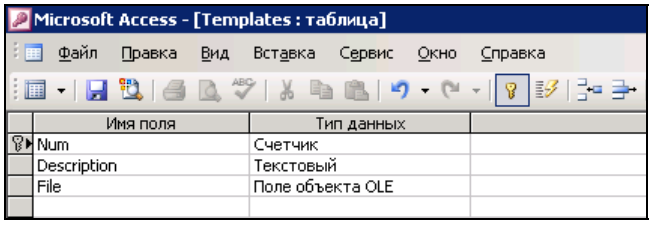


Рис. 12.4. Таблица для хранения шаблонов Word

Для кнопки, по нажатию на которой будет формироваться отчет, можно использовать следующий код:

```
'Получаем ссылку oFrame на объект присоединенной рамки на форме
Dim oFrame As BoundObjectFrame
Set oFrame = oForm.Controls("WordTemplate")

'При помощи метода DLookup() скачиваем в него значение столбца File
'из таблицы Templates, где номер строки (значение столбца Num) = 1
oFrame = Application.DLookup("[File]", "Templates", "[Num] = 1")

'Открываем объект в отдельном окне приложения, т. е. создаем
'документ Word на основе шаблона, загруженного в рамку объекта на форме
oFrame.Verb = acOLEVerbOpen

'Активизируем объект приложения
oFrame.Action = acOLEActivate

'Получаем ссылку на Word в переменную oWord
Dim oWord As Word.Application
Set oWord = GetObject(, "Word.Application")

'Получаем ссылку на созданный нами документ
Dim oDoc As Word.Document
Set oDoc = oWord.ActiveDocument

'Дальше работаем средствами Word, например, вставляем нужный текст
'в места, отмеченные закладками
```

Конечно же, правильнее будет при этом сделать эту присоединенную рамку объекта на форме изначально невидимой, чтобы пользователь не мог активизировать этот объект по собственной инициативе.

- **Разрыв страницы** — этот элемент управления определяет начало нового экрана формы.
- **Подчиненная форма/отчет** — используется для размещения на форме подчиненных форм, таблиц или отчетов.

Как уже говорилось, программным способом элементы управления в форме Access приходится создавать редко. Если на форме вам нужен переменный набор элементов управления, то правильнее будет с самого начала добавить на форму все нужные элементы управления и по необходимости делать их то видимыми, то невидимыми. Тем не менее создать программным способом элементы управления на форме тоже можно. Эта операция выполняется при помощи метода `Application.CreateControl()`, который принимает множество параметров: имя формы, на которой создается элемент управления, тип элемента управления, его месторасположение на форме и т. п.

Обращение к значениям элементов управления на форме производится через коллекцию `Controls`, которая умеет работать с именами элементов управления:

```
cSumInNumber = oForm.Controls("SumNumber").Value
```

12.6. Свойства, методы и события форм

Далее для справки приводится информация о самых важных свойствах, методах и событиях, которые разработчики предусмотрели для объекта `Form` в Access.

Этот объект имеет следующие свойства.

- `ActiveControl` — позволяет определить, в каком элементе управления формы в данный момент находится фокус. Обычно используется для проверок. Это свойство, помимо формы, есть также у отчета и специального объекта `Screen`. При помощи свойства `Screen.ActiveControl` можно определить не только имя активного в настоящий момент элемента управления, но и к какой форме или отчету он относится, например:

  ```
  MsgBox Screen.ActiveControl.Parent.Name
  ```

- Свойства с префиксом `After...` — позволяют заменить обычные событийные процедуры, назначив имя процедуры какому-либо событию (`AfterInstall` — вставка новой записи, `AfterUpdate` — изменение существующей записи и т. п.).

- `AllowAdditions` — определяет, разрешается или нет пользователю добавлять новые записи через данную форму. Для того чтобы разрешить или запретить удаление записей, используется свойство `AllowDeletions`, для разрешения или запрета редактирования существующих записей — свойство `AllowEdits`. Остальные свойства с префиксом `Allow...` относятся к разрешению или запрету для пользователя использовать различные режимы отображения информации.

- `AutoCenter` — определяет, будет ли форма выводиться точно по центру окна приложения.
- `AutoResize` — определяет, будет ли форма автоматически изменять свои размеры, подстраиваясь под размеры записей на ней. Это свойство нужно использовать очень осторожно, поскольку при таком автоматическом изменении размеров пользователь может увидеть на экране совсем не то, что вы задумывали.
- Свойства с префиксом `Before...` — эти многочисленные свойства, так же как и `After...`, заменяют событийные процедуры форм.
- `Bookmark` — очень полезное свойство. Работает точно так же, как и одноименное свойство объекта `Recordset` в ADO *(см. гл. 9)*. Оно позволяет сохранить информацию о текущей записи (закладку на нее) в строковой переменной. Если мы присвоим этому свойству значение, сохраненное в строковой переменной, то форма вернется на ту запись, для которой была сохранена закладка.
- `BorderStyle` — позволяет определить рамку на форме (напрямую, без вложенных объектов).
- `Caption` — определяет заголовок формы.
- `ChartSpace` — представляет специальный контейнерный объект, который может содержать в себе до 64 диаграмм. Используется для программного создания диаграмм на формах и для программного изменения существующих диаграмм.
- `CloseButton` — позволяет не возиться со свойствами кнопки **Close**, а делать ее то доступной, то недоступной для пользователя прямо из свойств формы. Используется, конечно, чтобы не дать пользователю прервать выполнение каких-то действий (например, ввода данных) на полпути.
- `ControlBox` — позволяет добавить или убрать меню **Control** (значок в левой части заголовка формы с командами **Переместить**, **Свернуть**, **Развернуть** и т. п.). Если убрать это меню, то исчезнут и специальные иконки в правом углу заголовка формы — **Свернуть**, **Развернуть** и **Закрыть**. Обычно это свойство используется для специальных модальных форм, например, выполняющих роль окон предупреждений. К сожалению, настраивать это свойство можно только тогда, когда форма открыта в режиме конструктора. Схожим образом работает свойство `MinMaxButtons`, которое позволяет определить, будут ли на форме видны кнопки **Свернуть** и **Развернуть** (тоже только в режиме конструктора).
- `Controls` — одно из важнейших свойств. Возвращает коллекцию `Controls` со всеми элементами управления на данной форме.

- `Count` — возвращает количество элементов управления на форме. Можно сказать, что относится к коллекции `Controls`, а не к самой форме.
- `CurrentRecord` — возвращает номер текущей записи из числа всех записей, которые открыты на форме. Это свойство доступно только для чтения.
- `CurrentView` — очень важное и полезное свойство. Оно позволяет определить, в каком режиме открыта данная форма: в режиме конструктора (**Design View**), в режиме формы (**Form View**) или в виде таблицы данных (**Datasheet View**). Свойство доступно только для чтения. Для переключения между режимами нужно использовать методы `DoCmd.Close()` и `DoCmd.OpenForm()` с соответствующими параметрами.
- `Cycle` — это свойство позволяет определить, что будет, когда пользователь нажмет на клавишу <Tab>, находясь на последней записи по порядку перехода: перейдет ли он на первый элемент управления для следующей записи (по умолчанию), вернется ли на первый элемент управления для текущей записи или будет ходить по кругу в рамках текущей страницы (экрана) формы.
- `DataEntry` — позволяет перевести форму в хитрый режим ввода новых данных, когда для пользователя на форме будет доступна только одна новая запись — пустая. Никакие уже существующие записи пользователю видны не будут.
- Свойства с префиксом `DataSheet...` — эти многочисленные свойства позволяют определить внешний вид формы, когда она открыта в режиме **Datasheet View**, т. е. в виде таблицы.
- `DefaultControl` — предоставляет интересную возможность, которая может сильно сократить количество набираемого кода, если вы создаете форму программным способом. При этом вы можете определить параметры для виртуального элемента управления `DefaultControl` любого типа. После того, как вы создадите новый элемент управления такого же типа, к нему будут автоматически применены те настройки, которые вы определили для `DefaultControl`. Такая возможность может быть очень удобной, например, если на форме вам программным способом придется создать десяток текстовых полей с одинаковым шрифтом, цветом фона и т. п. Например:

```
'Создаем текстовое поле по умолчанию
Set oDefaultTextBox = oForm.DefaultControl(acTextBox)
oDefaultTextBox.FontSize = 12
'Создаем новое текстовое поле на форме — уже с размером шрифта 12
Set oTextBox1 = CreateControl(oForm.Name, acTextBox, , , , 500, 500)
```

- `DefaultView` — определяет, в каком режиме форма будет открываться по умолчанию. Пользователь сможет переключиться в другой режим, если это будет разрешено ему при помощи свойства `ViewsAllowed`.
- `Dirty` — принимает значение `True`, если текущая запись была изменена, но еще не сохранена в базе данных. Как только текущая запись будет сохранена, значение этого свойства опять изменится на `False`. Обычно используется для проверок во избежание потери введенных пользователем данных.
- `DividingLines` — определяет, будут ли на форме разделительные линии, отделяющие друг от друга области формы или записи, если их на форме очень много.
- `FetchDefaults` — определяет, будут ли на форме выводиться значения по умолчанию для столбцов при заполнении новой записи.
- `Filter` — исключительно важное свойство. Оно позволяет отфильтровать записи на форме (например, по диапазону дат, выбранных пользователем, по значению какого-либо столбца, которое пользователь может, например, выбрать при помощи раскрывающегося списка и т. п.). В качестве значения этому свойству передается то выражение, которое должно идти после ключевого слова `Where` в запросах на языке SQL. Вместе с этим свойством обычно для включения фильтра используется свойство `FilterOn`. Например, если форма привязана к таблице со столбцом `City` и нужно в форме отобразить только те записи, для которых в этом столбце значится `"Санкт-Петербург"`, можно использовать следующий код:

  ```
  oForm.Filter = "City = 'Санкт-Петербург'"
  oForm.FilterOn = True
  ```

  Учтите, что это свойство используется для фильтрации записей только в самой форме. Если вы обращаетесь при помощи формы к данным в базе данных SQL Server или Oracle, то чаще всего бывает выгоднее применить фильтр в запросе, передаваемом на сервер (чтобы не скачивать в приложение лишние данные). Для этой цели используется свойство `ServerFilter` *(см. далее)*.
- `Form` — это свойство позволяет получить ссылку на форму или отчет, которые помещены на форму при помощи элемента управления **Подчиненная форма/отчет**.
- `FrozenColumns` — позволяет определить количество закрепленных столбцов, т. е. тех столбцов, которые не будут уходить за экран при прокрутке формы вправо. Это свойство доступно только для чтения. Для настройки параметров в форме предлагается использовать на графическом экране команду **Закрепить столбцы** в меню **Формат**.

- `HasModule` — это свойство определяет, есть ли у данной формы или отчета свой собственный модуль класса (по умолчанию нет). При помощи пользовательского модуля класса можно добавить для формы свои собственные свойства и методы или переопределить существующие. Обычно такая возможность, как и работа с пользовательскими классами в VBA, используется только в очень больших и сложных проектах со специфическими требованиями.
- `HelpFile` и `HelpContextId` — позволяют определить файл справки и закладку в нем для данной формы.
- `Hwnd` — позволяет вернуть дескриптор окна Windows для данной формы. Используется при работе с Windows API.
- `InputParameters` — позволяет вернуть (в виде строкового значения) информацию о параметрах, которые были переданы на источник данных при выполнении запроса, результаты которого были загружены в форму.
- `KeyPreview` — это свойство определяет, кому будут передаваться нажатия клавиш: только активному элементу управления на форме (по умолчанию) или вначале форме, а затем уже активному элементу управления. При помощи этого свойства можно обрабатывать событийными процедурами для формы клавиатурные комбинации, в каком бы месте на форме не находился фокус, например, так:

```
Private Sub Form_Load()
    Me.KeyPreview = True
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    Select Case KeyCode
        Case vbKeyF5
            MsgBox "Нажата клавиша F5!"
        Case vbKeyF6
            MsgBox "Нажата клавиша F6!"
    End Select
End Sub
```

- `MenuBar` — позволяет указать имя пользовательского меню, которое будет автоматически появляться при открытии данной формы или отчета (см. также свойства `ShortcutMenuBar` и `Toolbar`).
- `Modal` — очень важное свойство. Если для него будет установлено значение `True`, то форма станет модальной. Это значит, что ее придется закрыть, прежде чем фокус можно будет передать другой форме. Чтобы отключить еще и панели инструментов и меню на время работы формы, нужно установить также значение свойства `Popup` в `True`.

- `Module` — это свойство позволяет получить ссылку на объект модуля класса для данной формы (см. далее свойство `HasModule`). С этим свойством нужно быть осторожным: если форма находится в режиме конструктора, то обращение к нему автоматически приведет к созданию нового модуля класса для формы (если для этой формы модуля класса еще не существовало).
- `Moveable` — позволяет определить, сможет ли пользователь перемещать форму на экране. На программное изменение местонахождения формы не влияет: программным способом перемещать форму можно всегда.
- `Name` — это, конечно, имя формы, под которым она будет видна в окне базы данных.
- `NavigationButtons` — это свойство позволяет определить, будут ли в распоряжение пользователя предоставлены кнопки для перехода по записям (на первую, последнюю, новую, следующую и т. п.) в нижней части формы.
- `NewRecord` — позволяет определить, является ли текущая запись для формы новой записью. Это свойство используется для проверок.
- `ObjectPalette` и `PaintPalette` — позволяют определить для вновь создаваемых элементов на форме и для самой формы соответственно цветовую схему Windows. В отличие от системы Windows, Access позволяет использовать неограниченное количество цветовых схем одновременно. Цветовая схема будет сохранена вместе со схемой или отчетом. Свойство `PaletteSource` позволяет указать цветовую схему для формы или отчета в виде файлов на диске.
- Свойства с префиксом `On...` (`OnClick`, `OnClose`, `OnConnect` и т. п.) — эти многочисленные свойства, как `After...` и `Before...`, заменяют событийные процедуры для формы, позволяя назначить событию имя макроса Access или процедуры VBA.
- `OnTimer` — это свойство позволяет определить реакцию на события, которые автоматически генерируются Access через определенный интервал времени, определяемый при помощи свойства `TimeInterval`.
- `OpenArgs` — позволяет получить доступ к параметру, передаваемому форме при ее открытии методом `DoCmd.OpenForm()`.
- `OrderBy` — свойство, которое позволяет настроить сортировку записей в форме. Принимает строковое выражение, состоящее из имени столбца, по которому производится сортировка, и направления сортировки, например:

```
oForm.OrderBy = "City DESC"
oForm.OrderByOn = True
```

Непосредственно включение сортировки производится при помощи свойства `OrderByOn`.

- `Page` — возвращает информацию о номере текущей страницы формы или отчета при печати. Обычно используется в текстовых полях для генерации номеров страниц. Доступно только в режиме предпросмотра перед печатью или при печати. Совместно с ним часто используется свойство `Pages`, которое возвращает общее количество страниц в форме или отчете.
- `Painting` — позволяет запретить перерисовку данной формы или отчета. Обычно устанавливается в `False` на время выполнения длинной ресурсоемкой операции для сбережения ресурсов. После завершения такой операции это свойство нужно опять установить в `True`. Отключить перерисовку для всех окон Access можно при помощи макрокоманды `ВыводНаЭкран` (метода `Echo()` объекта `DoCmd`).
- Свойства с префиксом `Picture...` — позволяют поместить на форму изображение (обычно в виде фонового режима) и настроить его параметры.
- `PivotTable` — позволяет получить доступ к объекту `PivotTable`, если он размещен на форме. Объект `PivotTable` в Access — это Web-версия (компонент ActiveX) сводной таблицы, которая была рассмотрена в *разд. 11.8*.
- `PopUp` — это свойство, установленное в `True`, определяет, что форма будет открыта как всплывающее *(popup)* окно поверх всех других окон Access. При этом стандартные меню и панели инструментов Access станут недоступны. Обычно используется вместе со свойством `Modal`.
- `Printer` — позволяет получить информацию или (чаще всего) настроить принтер по умолчанию — для формы, отчета или всего приложения (это свойство есть и у объекта `Application`). Проверить, будет ли использоваться принтер по умолчанию, можно при помощи свойства `UseDefaultPrinter`.
- `Properties` — возвращает коллекцию объектов `Property`, представляющих все встроенные свойства формы или отчета. У этих объектов всего два своих свойства: `Name` (имя свойства) и `Value` (его значение). Обычно используется для того, чтобы пройти циклом по всем свойствам формы или отчета.
- Свойства с префиксом `Prt...` — предназначены для настройки принтера перед печатью формы или отчета. Принимают в качестве параметров достаточно сложные байтовые структуры. Рекомендуется работать с этими свойствами, только обладая полной информацией о возможностях драйвера принтера.
- `RecordLocks` — позволяет определить, как будет происходить блокировка записей в форме при одновременном редактировании данных через эту форму сразу несколькими пользователями: 0 — записи блокироваться не

будут (вы рискуете тем, что пользователь при попытке сохранения изменений в своей записи столкнется с сообщением об ошибке); 1 — все записи в таблице, на которую ссылается форма, будут заблокированы, остальные пользователи смогут их лишь просматривать; 2 — будет заблокирована страница с изменяемой записью (примерно 4 Кбайт в базе данных), остальным пользователям данные этой страницы будут доступны только для чтения.

- `RecordSelectors` — определяет, будет ли доступна в режиме формы (**Form View**) область выделения записи — прямоугольник в левой части формы, при помощи которого можно выделить всю запись.
- `Recordset` — позволяет вернуть или настроить объект `Recordset` из библиотек ADO *(см. гл. 9)* или DAO (унаследованная библиотека, которая в этой книге не рассматривается), который используется в качестве источника информации для формы. Применение этого свойства может быть очень удобным при обращении к внешним источникам данных. Кроме того, в вашем распоряжении появляются очень удобные и хорошо известные многим разработчикам свойства и методы объекта `Recordset`.
- `RecordsetType` — позволяет определить тип объекта `Recordset`, на котором основана форма, с точки зрения возможности редактирования его информации.
- `RecordSource` — позволяет определить источник данных для формы или отчета. Принимает текстовое значение, которое может быть именем таблицы или представления или запросом на языке SQL. Если вы изменили свойство `Recordset`, то значение этого свойства меняется автоматически.
- `RecordSourceQualifier` — используется только тогда, когда источник данных для формы Access — это база данных SQL Server. Возвращает имя владельца таблицы или представления, на которое ссылается форма.
- `ResyncCommand` — позволяет самостоятельно определить команду, которая пойдет на источник данных для записи изменений, внесенных пользователем через форму. Это свойство используется только тогда, когда команда на изменение данных, генерируемая автоматически, вас по какой-то причине не устраивает.
- `ScrollBars` — определяет, отображать ли на форме горизонтальную и вертикальную полосы прокрутки. По умолчанию обе полосы прокрутки на форме отображаются.
- `Section` — это свойство очень похоже на метод. Оно принимает параметр, обозначающий область (секцию) формы или отчета, и используется для настроек свойств этой области, например:

```
oForm.Section(acPageHeader).Visible = False
```

- `ServerFilter` — это свойство позволяет вставить фильтр в запрос, который при открытии формы отправляется источнику данных (например, базе данных SQL Server или Oracle). При этом синтаксическую правильность строкового выражения, которое вы передаете в качестве фильтра, Access не проверяет. Чтобы включить работу с серверным фильтром, необходимо установить значение свойства `ServerFilterByForm` в `True`:

  ```
  oForm.ServerFilter = "City = 'Санкт-Петербург'"
  oForm.ServerFilterByForm = True
  ```

  Удобнее всего помещать этот код в событийную процедуру для события `OnOpen` формы.

- `ShortcutMenu` — если установить значение этого свойства в `False`, то будет запрещен показ контекстных меню как для самой формы, так и для расположенных на ней элементов управления.

- `ShortcutMenuBar` — позволяет определить свое собственное пользовательское контекстное меню, которое будет показываться по щелчку правой кнопкой мыши на форме или отчете. Это свойство предусмотрено, кроме формы и отчета, практически для всех элементов управления.

- `Tag` — простой тег для формы. Можно считать его пользовательским атрибутом, который доступен только из программы и ни на что не влияет. Обычно используется для хранения служебной информации (например, меток). Принимает строковое значение с максимальной длиной 2048 символов.

- `TimerInterval` — позволяет установить значение (в миллисекундах), через которое будет срабатывать таймер формы (по умолчанию он отключен). Обычно настраивается в событийной процедуре `Load` для формы. Чаще всего используется с событийной процедурой для события `Timer`. Применяется для организации ожидания (например, мы ждем, пока завершит работу какое-то внешнее приложение, проверяя его состояние каждые полсекунды), для анимации формы и т. п.

- `Toolbar` — позволяет определить панель инструментов, которая будет появляться каждый раз при открытой форме или отчете. Конечно, чаще всего используется своя собственная пользовательская панель инструментов.

- `UniqueTable` — используется тогда, когда таблица привязана к форме или хранимой процедуре, которые ссылаются сразу на несколько таблиц, и нам нужно определить, как в этой ситуации будут вноситься изменения при конфликте.

- `ViewsAllowed` — определяет, в каком режиме пользователь может работать с формой: только в режиме **Form View**, только в режиме **DatasheetView**

или в обоих режимах (по умолчанию). Режим конструктора доступен для пользователя всегда. Запретить его можно только при помощи разрешений (меню **Сервис | Защита | Разрешения**).

- `Visible` — это свойство позволяет скрыть форму или, наоборот, сделать ее видимой.
- `WindowHeight`, `WindowLeft`, `WindowTop`, `WindowWidth` — определяют размеры окна формы.

По сравнению со свойствами методов у объекта `Form` совсем немного. Для большинства из них назначение понятно из названия.

- `GoTo()` — позволяет перейти на указанную страницу многоэкранной формы.
- `Move()` — перемещает форму на экране.
- `Recalc()` — позволяет пересчитать значения в вычисляемых элементах управления формы. Если форма находится в фокусе, то на графическом экране можно вместо этого просто нажать клавишу <F9>.
- `Refresh()` — позволяет отобразить изменения, которые внесены в текущий набор данных в форме. Если нужно еще раз скачать данные из базы (например, они могли быть изменены другими пользователями), то нужно воспользоваться методом `Requery()`.
- `Repaint()` — перерисовывает форму. Обычно используется в тех ситуациях, когда перед этим автоматическая перерисовка была отключена при помощи свойства `Painting`.
- `SetFocus()` — позволяет установить фокус на форме.
- `Undo()` — очищает информацию, которую пользователь ввел для текущей записи (если эта информация ошибочна). Переход на новую запись обычно приводит к сохранению текущей, поэтому после такого перехода метод `Undo()` уже не поможет.

Для формы предусмотрено также несколько десятков событий, которые на практике используются очень активно. Эти события включают в себя как стандартные события форм VBA (например, `Load`), так и специфические, такие как `Query` — запрос к источнику данных, вставка, изменение или удаление записи через форму и т. п. С ними можно работать двумя способами: через специальные свойства формы (которые начинаются на `Before...`, `After...`, `On...` и т. п.) и обычным способом через событийные процедуры. Для этого достаточно в **Project Explorer** в списке объектов выбрать нужную форму, нажать клавишу <F7>, а затем выбрать нужную событийную процедуру (рис. 12.5).

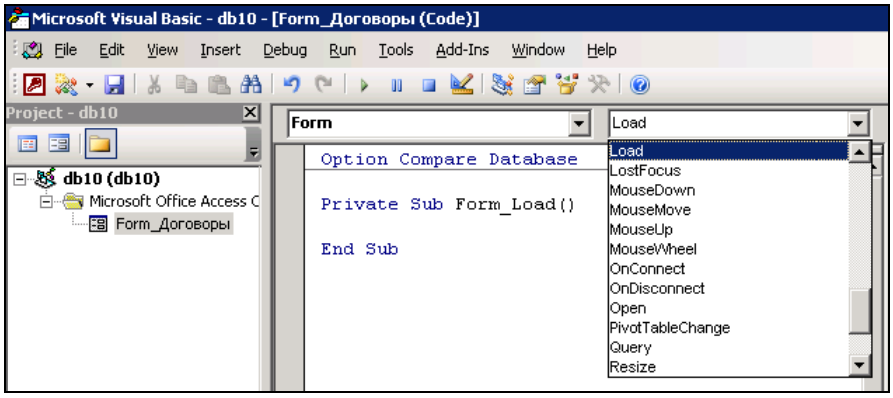


Рис. 12.5. Список событий формы

12.7. Работа с отчетами (объект *Report*)

Еще один часто используемый в программах объект Access — это отчет, представленный объектом `Report`. Отчеты Access — это, возможно, самый простой способ генерации отчетов к базам данных (по сравнению с другими способами генерации отчетов, например, такими, как применение Crystal Reports, Microsoft Reporting Services или уже рассмотренных нами связок VBA/ADO/Word и VBA/ADO/Excel). При помощи отчетов Access можно, конечно, генерировать отчеты не только для самих баз данных Access, но и для внешних источников данных, например, баз данных SQL Server или Oracle. При этом в отчетах дополнительные функциональные возможности (например, условное форматирование) реализуются именно средствами VBA.

С программной точки зрения работа с отчетами очень похожа на работу с формами. Точно так же доступ к объектам всех отчетов можно получить при помощи коллекции `Application.CurrentProject.AllReports` (в которой находятся объекты `AccessObject`), а доступ ко всем открытым отчетам — при помощи коллекции `Reports` с более традиционными объектами `Report`. Точно так же очень редко приходится создавать отчеты программными средствами — обычно эта операция производится из вкладки **Отчеты** окна базы данных. Однако отчеты программным образом приходится создавать все-таки чаще, чем формы. Программно создать отчет можно при помощи метода `Application.CreateReport()`:

```
Dim oReport As Report
Set oReport = Application.CreateReport()
```

В этом случае отчет будет создан только в оперативной памяти, откуда он бесследно исчезнет после завершения работы создавшей его процедуры. Со-

хранить отчет с именем по умолчанию (в русской версии Access это `Отчет1`, `Отчет2` и т. п.) можно, добавив в код строку:

```
DoCmd.Close , , acSaveYes
```

Но, конечно, чаще нам нужно сохранить созданный отчет с указанным нами именем. Для этой цели перед вызовом метода `DoCmd.Close()` нужно поставить вызов другого метода — `DoCmd.Save()`:

```
DoCmd.Save , "Отчет_по_продажам"
DoCmd.Close
```

Метод `CreateReport()` создаст пустой отчет, если только мы не передадим ему в качестве параметра шаблон отчета с уже имеющимися элементами управления. В отчете можно использовать те же элементы управления, что и в форме. Но прежде, чем помещать в отчет элементы управления, необходимо разобраться, где они будут находиться.

Отчет в Access может состоять из девяти областей, из которых по умолчанию видны только три:

- *Верхний колонтитул (Page Header)* — подразумевается верхний колонтитул для страницы. Он будет повторяться столько раз, сколько страниц в отчете. Обычно в него помещается номер страницы, информация о самом отчете, его авторе, в ленточных отчетах (когда в области данных данные идут в виде столбцов) возможно еще и заголовки столбцов;
- *Область данных (Details)*, иногда также называется "подробности" — основная область отчета, в которую обычно выводятся записи из базы данных. Эта область будет повторяться столько раз, сколько записей у нас в базе данных;
- *Нижний колонтитул (Page Footer)* — нижний колонтитул для страницы. Используется обычно для того же, что и верхний колонтитул.

Если щелкнуть правой кнопкой мыши по пустому месту в отчете и в контекстном меню выбрать **Заголовок/примечание отчета**, то будут показаны еще две области отчета (рис. 12.6):

- *Заголовок отчета (Report Header)* — специальная область, которая идет перед всем отчетом и не повторяется. Обычно в нее помещается сводная информация об отчете: автор, количество страниц, время вывода, итоги, диаграмма, представляющая данные в отчете, и т. п.;
- *Примечание отчета (Report Footer)* — область в конец отчета, которая также не повторяется. Используется для тех же целей, что и заголовок отчета.

Если вы используете в отчете группировку, то у вас появятся и другие области — верхние и нижние колонтитулы для ваших групп. Максимальное число

уровней группировки в отчетах Access — два и для каждого из них можно использовать свои колонтитулы. Обычно в них помещаются заголовки групп и итоги по группам.

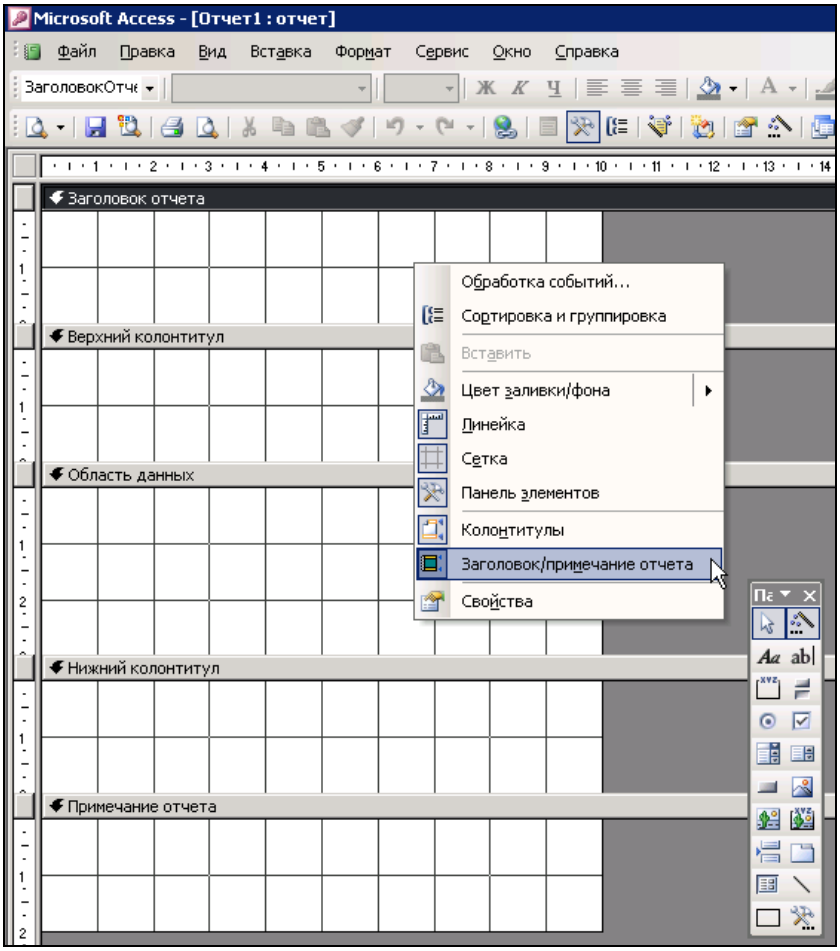


Рис. 12.6. Дополнительные области отчета в окне конструктора

После того, как мы разобрались, куда именно мы хотим поместить элемент управления, для его непосредственного размещения в отчете можно использовать метод `Application.CreateReportControl()`. В качестве параметров этот метод принимает имя отчета (он обязательно должен быть открыт), тип элемента управления, область отчета, в которую необходимо поместить этот элемент управления, а также имя столбца, к которому может быть привязан элемент управления и координаты элемента управления в отчете. Например,

создать в области данных ни к чему не привязанное текстовое поле с именем можно так:

```
Set txt1 = CreateReportControl(oReport.Name, acTextBox, acDetail)
txt1.Name = "txtCustomerID"
```

Если же вы пошли обычным путем и создали отчет не программно, а на графическом экране при помощи мастера или конструктора, то, возможно, вам потребуется его открыть программным образом. Делается это при помощи метода `DoCmd.OpenReport()`. Однако здесь есть одна тонкость: если вы выполните самый простой вариант кода, например:

```
DoCmd.OpenReport "Отчет_по_продажам"
```

то отчет, вместо того, чтобы открыться в окне просмотра, отправится на печать. Чтобы все-таки просмотреть его, нужно использовать код вида:

```
DoCmd.OpenReport "Отчет_по_продажам", acViewPreview
```

Обращаться к элементам управления в отчете можно точно так же, как и к элементам управления формы — при помощи коллекции `Controls`. Например, сделать наш отчет невидимым можно так:

```
oReport.Controls("Отчет_по_продажам").Visible = False
```

Свойства, методы и события объекта `Report` практически полностью совпадают со свойствами, методами и событиями объекта `Form`, которые были рассмотрены в *разд. 12.5*. Единственное принципиальное исключение — в объекте `Report` предусмотрен специальный набор методов для рисования, таких как `Circle()` (нарисовать круг или эллипс), `Line()` (нарисовать линию), `PSet()` (установить цвет для отдельного пиксела) и т. п., а также набор свойств и методов для программного вывода текстовых надписей. Но обычно нет смысла увлекаться довольно трудоемким рисованием или выводом текста, намного проще выполнить необходимые действия в режиме конструктора на графическом экране. Если вместо рисования кругов и линий на экране вы воспользуетесь элементов управления `Image` (рисунок), то возможностей у вас будет намного больше.

12.8. Другие объекты Access

В объектной модели Access предусмотрены и другие объекты, которые в программировании средствами VBA используются реже, чем уже рассмотренные нами `Application`, `DoCmd`, `Form` и `Report`. Далее приведена краткая информация об этих объектах.

В Access предусмотрена очень удобная возможность, которая, по моему опыту, мало известна как программистам, так и пользователям. Эта возможность

называется *Страницы доступа к данным* (*Data Access Pages*). Применение страниц доступа к данным — это самый простой способ создать Web-форму для занесения информации на источник данных (в качестве источников для страниц доступа к данным в настоящее время можно использовать только базы данных Access и Microsoft SQL Server). Физически вы создаете Web-страницу с элементом управления ActiveX, который и обеспечивает необходимую функциональность для подключения к источнику данных, выполнения на нем различных операций и т. п. На форме вы можете использовать привычный набор элементов управления (кнопки, текстовые поля и т. п.), код для которых можно писать на языке VBScript (ближайший родственник VBA). Одним из главных преимуществ страниц доступа к данным является то, что этот код будет выполняться не в среде выполнения браузера, а в среде выполнения этого элемента ActiveX, поэтому в вашем распоряжении останутся все возможности работы с объектными моделями Windows. Например, из кода обработки события `Click` вы можете создать объект `ADO.Recordset`, или запустить Word, Excel или Access на компьютере пользователя, или подключить сетевой диск и т. п.

Страницы доступа к данным можно создавать как в виде объектов базы данных Access (для этой цели в окне базы данных предусмотрена вкладка **Страницы**), так и отдельно от баз данных — просто в виде HTML-файлов. Для этого в Access в меню **Файл** нужно выбрать пункт **Создать**, а потом — **Пустая страница доступа к данным**.

В объектной модели Access страницы доступа к данным представлены объектом `DataAccessPage`. Поскольку такие страницы — это фактически Web-формы, то набор их свойств и методов представляет из себя урезанный набор свойств и методов обычных форм. Web-параметры для страниц доступа данных определяются при помощи специального объекта `WebOptions`.

В формах или отчетах часто приходится использовать условное форматирование, когда формат какого-либо текстового поля или комбинированного списка зависит от различных условий (например, от значения в столбце текущей записи). Например, в зависимости от значения соседнего столбца текстовое поле может становиться то видимым, то невидимым. Для применения условного форматирования используются объекты `FormatCondition`, которые сведены в коллекцию `FormatConditions`.

Объект `GroupLevel` используется при работе с группировкой в отчетах и формах.

Объект `Module` представляет программные модули в базе данных Access — стандартные или модули классов. Эти объекты сведены в коллекцию `Modules`, доступную через одноименное свойство объекта `Application`. Обычно объект `Module` используется для автоматического добавления программного кода в

проекты. Для этой цели в этом объекте предусмотрены специальные методы, такие как `AddFromFile()`, `AddFromString()`, `CreateEventProc()` и т. п.

Для автоматического добавления (или замены) ссылок на объектные библиотеки в Access предусмотрена специальная коллекция `References` с набором объектов `Reference` (ссылок). Свойства и методы коллекции `References` в официальной документации почему-то не рассматриваются, но найти нужные методы можно при помощи подсказок в окне редактора кода. Добавление в проект новой ссылки на объектную библиотеку обычно производится при помощи метода `References.AddFromFile()`.

Объект `SmartTag` представляет смарт-тег — специальное слово или словосочетание, которое автоматически распознается Microsoft Office. В реальных приложениях смарт-теги используются редко, но эта новая возможность приложений Office активно продвигается Microsoft.

Задание для самостоятельной работы 12: Создание приложения VBA в Access

Ситуация:

Приложение, которое было создано для автоматизации формирования договоров в Word *(см. самостоятельную работу к гл. 10)*, решено было изменить, поскольку в нем обнаружились следующие недостатки:

- программа не обеспечивала сохранение данных, введенных пользователем через форму, поэтому эти данные нельзя было использовать повторно. Намного удобнее было бы сохранять вводимые данные в базе данных, например, Access;
- для работы программы необходимы два компонента: программный код (включая форму VBA), который физически находится в файле Normal.dot и шаблон DogovorTemplate.dot, который должен находиться в корневом каталоге диска C:. Такое построение приложения затрудняет его перенос между компьютерами.

ЗАДАНИЕ:

Измените созданное вами в работе 10 приложение, которое обеспечивает автоматическое создание договоров в формате документов Word, таким образом, чтобы оно обеспечивало сохранение данных о договорах в базе данных Microsoft Access. Шаблон документа Word, на основе которого должен формироваться договор, должен находиться в той же базе данных Access. Обеспечьте минимальную защиту приложения от ошибочных действий со стороны пользователя: при запуске приложения пользователь не должен видеть

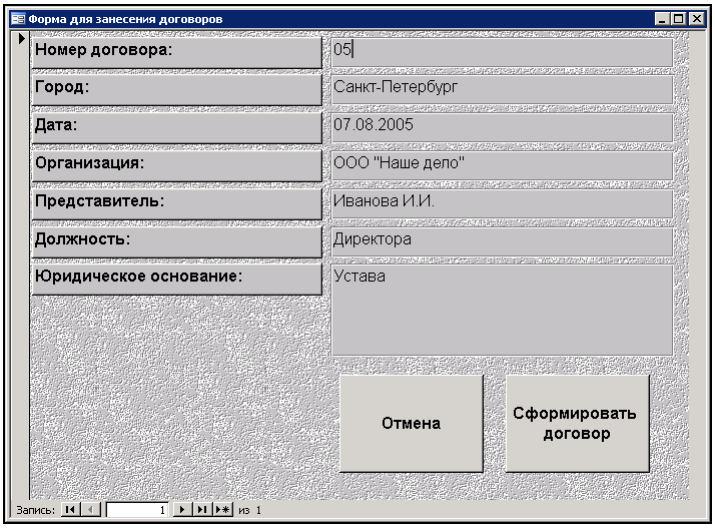


Рис. 12.7. Форма для занесения данных договоров

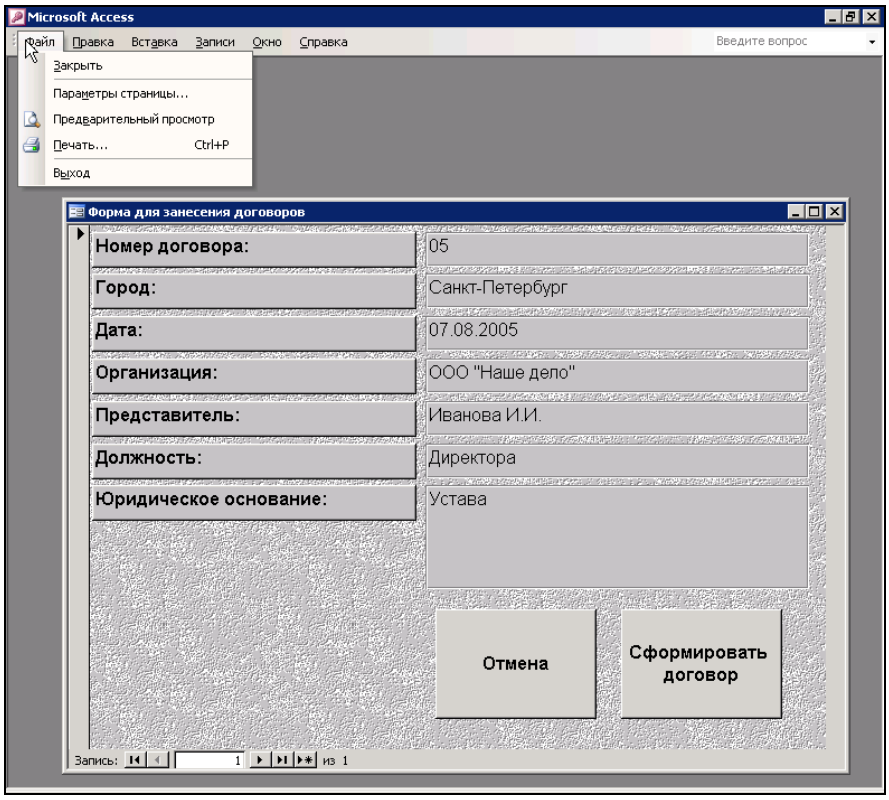


Рис. 12.8. Окно готового приложения

никакие другие объекты, кроме формы для заполнения данных договора. Для этого:

1. Создайте новую базу данных Access, а в ней — необходимые таблицы для хранения данных договоров и шаблонов документов Word.
2. Создайте в этой базе данных форму, аналогичную представленной на рис. 12.7, для занесения информации о договорах в таблицу базы данных и для формирования договора в формате документа Word. Создайте для этой формы необходимый программный код.
3. Обеспечьте минимальную защиту от неверных действий пользователя: при запуске приложения пользователь должен видеть только созданную вами форму (рис. 12.8). Все другие объекты должны быть спрятаны.

Ответ к заданию 12

К пункту 1 задания (создание базы данных и необходимых таблиц):

1. Запустите Microsoft Access и выберите пункт меню **Файл | Создать**. В окне **Создание файла** выберите **Новая база данных**. Сохраните созданную базу данных в корневом каталоге диска C: и назовите ее Dogovors.mdb.
2. В открывшемся окне базы данных перейдите на вкладку **Таблицы** и щелкните два раза левой кнопкой мыши по строке **Создание таблицы в режиме конструктора**. В созданной таблице определите три столбца (рис. 12.9).

   Сохраните эту таблицу как **Шаблоны** и закройте окно конструктора.

Рис. 12.9. Структура таблицы **Шаблоны**

3. В окне базы данных на вкладке **Таблицы** еще раз щелкните по строке **Создание таблицы в режиме конструктора**. Набор столбцов для новой таблицы должен выглядеть так, как приведено в табл. 12.1.

Таблица 12.1. *Структура таблицы* ***Договоры***

Имя столбца	Тип данных	Размер
НомерДоговора	Текстовый (первичный ключ)	10
Город	Текстовый	30

Таблица 12.1 *(окончание)*

Имя столбца	Тип данных	Размер
Дата	Дата/время	50
Организация	Текстовый	50
Представитель	Текстовый	50
Должность	Текстовый	50
ЮрОснование	Текстовый	100

Сохраните эту таблицу с именем **Договоры** и закройте окно конструктора таблицы.

4. На вкладке **Таблицы** окна базы данных щелкните два раза левой кнопкой мыши по созданной таблице **Шаблоны**, чтобы открыть ее в режиме ввода данных. В первую строку этой таблицы в столбец **Описание** введите "Шаблон договора", а затем выделите ячейку в столбце **Шаблон** и в меню **Вставка** выберите **Объект**. В открывшемся окне переставьте переключатель в положение **Создать из файла**, затем нажмите на кнопку **Обзор** и выберите шаблон C:\DogovorTemplate.dot, который вы создали в работе 10. Затем нажмите кнопку **OK**, чтобы поместить шаблон внутрь базы данных.
5. Щелкните по ячейке для помещенного вами шаблона, чтобы убедиться, что он действительно помещен в базу данных. Убедитесь, что в столбце **НомерШаблона** для первой строки автоматически сгенерировано значение 1, закройте эту таблицу.

К пункту 2 задания (создание формы Access и программного кода для формирования файла договора):

1. В окне базы данных перейдите на вкладку **Формы** и щелкните два раза левой кнопкой мыши по строке **Создание формы с помощью мастера**. Откроется окно мастера создания форм.
2. На первом экране мастера в списке **Таблицы и Запросы** выберите **Таблица: Договоры**, затем поместите в список **Выбранные поля** все поля из этой таблицы и нажмите кнопку **Далее**.
3. На следующем экране выберите внешний вид формы в один столбец и нажмите кнопку **Далее**.
4. На следующем экране выберите наиболее вам понравившийся стиль и нажмите кнопку **Далее**.
5. На последнем экране в окне определения имени формы введите имя формы **Форма для занесения договоров**, установите переключатель в поло-

жение **Изменить макет формы** и нажмите кнопку **Готово**. Форма будет открыта в режиме конструктора.

6. Произведите расстановку и поправьте оформление созданных элементов на форме средствами конструктора по вашему вкусу.
7. Добавьте при помощи **Панели инструментов** на свободную часть формы элемент управления **Присоединенная рамка объекта**. Удалите автоматически сгенерированную вместе с ним надпись, а затем откройте свойства этого объекта. Для свойства **Имя** установите значение `OLEObject1`, а для свойства **Вывод на экран** — значение `Нет`.
8. Добавьте на форму две кнопки: **Отмена** и **Сформировать договор**. Элемент управления для первой кнопки должен называться `cmdCancel`, а для второй — `cmdDog`. В открывающемся окне мастера при создании кнопки нажимайте на кнопку **Отмена**.
9. Убедитесь, что для элементов управления текстовых полей оставлены имена по умолчанию (**НомерДоговора**, **Город**, **Дата** и т. п.). В итоге форма в окне конструктора должна выглядеть, например, так, как представлено на рис. 12.10.

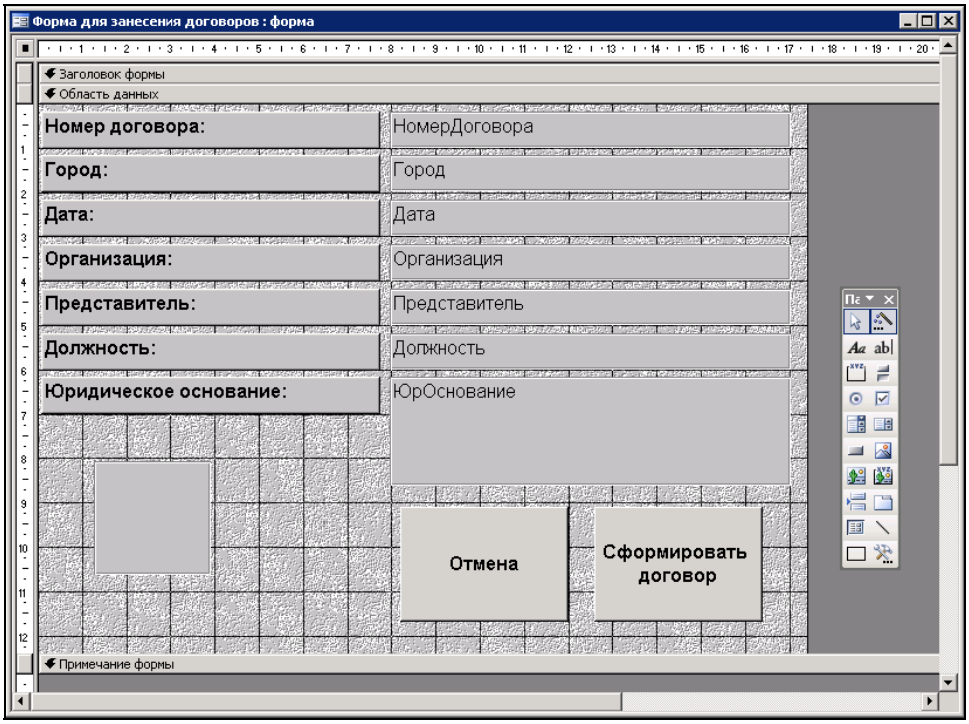


Рис. 12.10. Форма в окне конструктора

10. При помощи меню **Tools | References** в окне редактора кода добавьте ссылку на объектную библиотеку Microsoft Word 11.0 Object Library.
11. Щелкните правой кнопкой мыши по кнопке **Отмена** и в контекстном меню выберите **Обработка событий**. В открывшемся окне **Построитель** выберите **Программы** и нажмите **OK**. Введите следующий код для события `Click` этой кнопки:

```
Private Sub cmdCancel_Click()
    Form.Undo
End Sub
```

12. Так же откройте код для события `Click` кнопки **Сформировать договор** и введите следующий код:

```
Private Sub cmdDog_Click()
    Dim dDate As Date
    Dim НомерДоговора, Город, Организация, Представитель, Должность, _
        ЮрОснование As String

    'Присваиваем значения переменным при помощи элементов управления
    If Form.Controls("Дата").Value <> "" Then _
        dDate = Form.Controls("Дата").Value
    If Form.Controls("НомерДоговора").Value <> "" Then _
        НомерДоговора = Form.Controls("НомерДоговора").Value
    If Form.Controls("Город").Value <> "" Then _
        Город = Form.Controls("Город").Value
    If Form.Controls("Организация").Value <> "" Then _
        Организация = Form.Controls("Организация").Value
    If Form.Controls("Представитель").Value <> "" Then _
        Представитель = Form.Controls("Представитель").Value
    If Form.Controls("Должность").Value <> "" Then _
        Должность = Form.Controls("Должность").Value
    If Form.Controls("ЮрОснование").Value <> "" Then _
        ЮрОснование = Form.Controls("ЮрОснование").Value

    'Получаем шаблон — теперь из базы данных
    Dim oBOF As BoundObjectFrame
    Set oBOF = Form.Controls("OLEObject1")
    oBOF = DLookup("[Шаблон]", "Шаблоны", "[НомерШаблона] = 1")
    oBOF.Verb = acOLEVerbOpen

    oBOF.Action = acOLEActivate

    'Получаем ссылки на запущенный нами Word и открытый в нем документ
    Dim oWord As Word.Application
```

```
    Set oWord = GetObject(, "Word.Application")
    Dim oDoc As Word.Document
    Set oDoc = oWord.ActiveDocument
    oWord.Visible = True
    oWord.ActiveWindow.WindowState = wdWindowStateMaximize
    oDoc.Activate

    'Вставляем данные в закладки
    oDoc.Bookmarks("bNumber").Range.Text = НомерДоговора
    oDoc.Bookmarks("bCity").Range.Text = Город
    oDoc.Bookmarks("bDate").Range.Text = dDate
    oDoc.Bookmarks("bOrg").Range.Text = Организация
    oDoc.Bookmarks("bTitle").Range.Text = Должность
    oDoc.Bookmarks("bPerson").Range.Text = Представитель
    oDoc.Bookmarks("bLaw").Range.Text = ЮрОснование

End Sub
```

13. Запустите созданный вами код на выполнение и убедитесь в его работоспособности.

К пункту 3 задания (обеспечение минимальной защиты от действий пользователя):

1. В окне базы данных Access в меню **Сервис** выберите **Параметры запуска**.
2. В открывшемся окне **Параметры запуска** снимите все флажки, а в списке **Вывод формы/страницы** выберите **Форма для занесения договоров** (рис. 12.11).

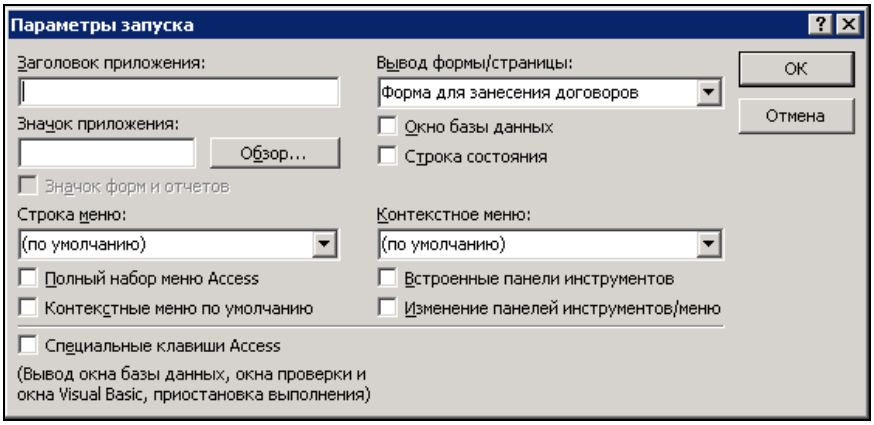


Рис. 12.11. Настройка параметров приложения в окне **Параметры запуска**

3. Нажмите кнопку **OK**. Затем закройте и вновь откройте созданную вами базу данных. Убедитесь, что все объекты базы данных, кроме формы для занесения договоров, спрятаны от пользователя.

Примечание

База данных откроется в обычном режиме, если при ее открытии удерживать нажатой клавишу <Shift>.