Томский государственный университет

Факультет информатики

А.В. Скворцов, Т.Н. Поддубная

Автоматизация ActiveX

(методические указания к

лабораторной работе № 9)

Томск – 1999

Указания РАССМОТРЕНЫ и УТВЕРЖДЕНЫ методической комиссией факультета информатики.

Протокол №____ от «____»_____1999 г.

Председатель методической комиссии факультета информатики,

профессор _____ Поддубный В.В.

Утверждаю. Зав. кафедрой прикладной информатики, доцент

_____ Сущенко С.П. «____» _____ 1999 г.

Методические указания посвящены программной среде Borland Delphi компании Inprise Corporation – одному из мощнейших современных средств разработки приложений для Windows.

Указания разработаны для студентов межфакультетской специализации факультета информатики Томского государственного университета. Содержат подробное описание лабораторной работы: цель, задачи, описание работы и используемых компонентов Delphi. В конце работы приведены тексты программ.

Резенцент – канд. техн. наук, доцент **Ю.Л. Костюк**.

В настоящее время в технологиях создания программного обеспечения видна явная тенденция к переходу на объектно-ориентированные принципы разработки. Эти принципы позволяют в большинстве случаев существенно упростить написание и отладку программ, хотя этап проектирования может несколько усложниться. Основные принципы объектно-ориентированной разработки предписывают рассматривать всю анализируемую предметную область как совокупность объектов, которые могут взаимодействовать между собой. Каждый объект имеет некоторый набор «методов» (процедур и функций) и свойств, только через которые и можно взаимодействовать с этим объектом.

Современные версии операционных систем от Microsoft по сути частично уже являются объектно-ориентированными. Для взаимодействия между объектами фирма Microsoft разработала целый ряд стандартов, объединенных между собой под единой маркой ActiveX.

Один из таких стандартов, называемый автоматизация, позволяет одним программам (называемым серверами автоматизации) определять список своих объектов, их свойств и методов, которые становятся доступными для внешнего использования, а другим (клиентам автоматизации) обращаться к этим объектам.

В данной работе на примере программы Microsoft Excel мы рассмотрим, как можно из программы в Delphi запустить Excel, создать в нем новую книгу и заполнить ее данными. Поэтому для выполнения данной работы на компьютере должен быть установлен Microsoft Excel 97/2000.

В создаваемом приложении должна быть обеспечена возможность выбора таблицы базы данных, её просмотр в виде таблицы, а также команда создания отчета по данной таблице с использованием ActiveX-автоматизации.

Цель работы

Ознакомление с основами программирования в Windows с помощью средств ActiveX-автоматизации.

Задачи работы

1. Ознакомление с основами программирования с применением автоматизации.
2. Создание программы для просмотра произвольных таблиц и создания отчета по заданной таблице в виде книги Microsoft Excel.

Описание работы

Наше приложение будет состоять всего из одной формы, на которой необходимо разместить компоненты для выбора базы данных и таблицы, компоненты для просмотра и навигации по таблице, а также кнопки для выбора таблицы базы данных, выхода из программы и создания отчета. Внешний вид формы приведен на рис. 35.

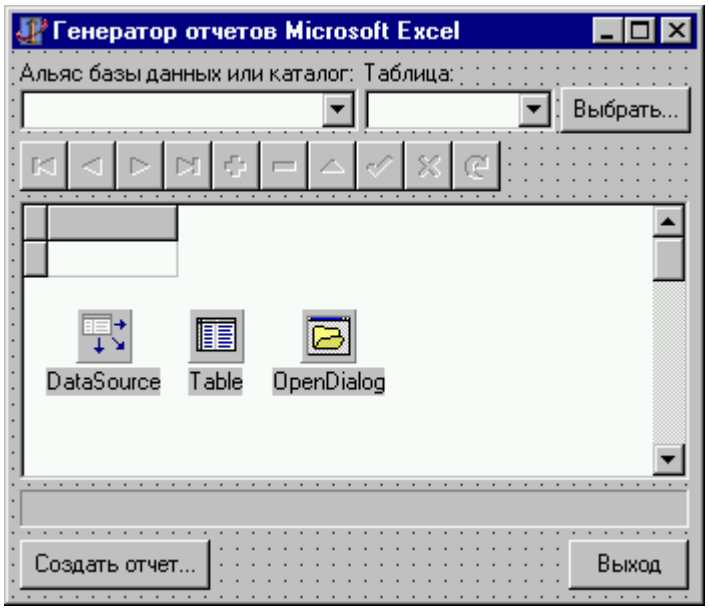


Рис. 35. Внешний вид главной формы

Для обращения к таблицам баз данных нам необходимо поместить на форму компонент `Table` типа `TTable`. Значения её свойств `DatabaseName`, `TableName` и `Active` должны изменяться в программе в зависимости от выбора пользователя. На форму необходимо поместить компонент `DataSource` типа `TDataSource` и связать его с компонентом `Table`. Далее на форму необходимо поместить также компоненты типов `TDBGrid` и `TDBNavigator`. Значение их свойств `DataSource` необходимо установить равным ранее размещенному компоненту `DataSource`.

Для того чтобы пользователь в программе мог выбирать имена баз данных и таблиц, соответственно необходимо разместить два комбосписка. Список элементов первого комбосписка для выбора имени базы данных необходимо заполнить в обработчике события формы `OnCreate`. Для этого необходимо вызвать метод `Session.GetDatabaseNames`. Список элементов второго комбосписка для выбора имени таблицы необходимо формировать в ответ на изменение имени базы данных с помощью вызова метода `Session.GetTableNames`. При изменении имени базы данных или таблицы необходимо заново попытаться открыть таблицу или выдать сообщение об ошибке открытия.

Для упрощения выбора таблицы, размещенной локально в некотором каталоге и имеющей формат DBase, FoxPro или Paradox, можно разместить на форме кнопку и диалог выбора файла. При нажатии кнопки нужно выдать диалоговое окно для выбора файла с таблицей и по его завершении занести в комбосписки с именами базы данных и таблицы соответственно значения пути к файлу и его имя.

После выполнения указанных шагов пользователь сможет выбирать произвольные таблицы и просматривать их содержимое. Теперь перейдем к шагу создания отчета.

Для программирования с использованием автоматизации вначале необходимо установить связь с программой-сервером автоматизации и связаться с каким-то из его объектов, например с книгой Microsoft Excel или с документом Microsoft Word. В Delphi это делается с помощью функции `GetActiveOleObject` для подключения к объекту уже запущенного сервера либо с помощью `CreateOleObject` для запуска приложения-сервера автоматизации и подключения к его объектам. Эти функции определены в модуле `ComObj`, поэтому перед их использованием этот модуль необходимо указать в секции **`uses`**. Аргументом вызова этих функций является имя создаваемого объекта сервера автоматизации, например «Excel.Application» для получения доступа к Microsoft Excel в целом либо «Word.Document» для создания в памяти временного текстового документа Microsoft Word.

Если при подключении к серверу произошла ошибка, например приложение-сервер не запущено в случае вызова функции `GetActiveOleObject` либо приложение-сервер вообще не установлено на компьютере, то возникает исключение, которое необходимо в программе соответствующим образом обработать.

Результатом вызова указанных функций соединения с сервером является специальный т.н. «диспетчеризуемый» интерфейс запрошенного объекта. Такие интерфейсы-диспетчеры позволяют вызвать процедуры, функции, обращаться к массивам и свойствам объектов сервера. В Delphi для упрощения доступа к объектам автоматизации такие интерфейсы лучше всего запоминать в переменных типа `Variant`, который позволяет хранить в себе данные разнообразных типов, в т.ч. и диспетчеризуемые интерфейсы объектов автоматизации.

В нашей программе результат вызова функции обращения к объекту «Excel.Application» необходимо запомнить, например, в переменной `Excel`, объявленной как имеющей тип `Variant`. Детальное описание всех имеющихся методов и свойств серверов автоматизации обычно имеется в соответствующих файлах справки.

Отключение от программы-сервера автоматизации производится в Delphi автоматически в тот момент, когда в переменных вариантного типа не останется интерфейсов к объектам сервера. Например, если все вариантные переменные объявлены внутри процедуры, то при выходе из процедуры эти переменные автоматически обнулятся, и соединение с сервером автоматически разорвется. Другой вариант заключается в принудительном присвоении вариантной переменной любого другого значения, например специального значения `UnAssigned`.

После подключения к Microsoft Excel в нашей программе необходимо создать новую пустую книгу с помощью вызова `Excel.WorkBooks.Add`. В данном случае выражение `Excel.WorkBooks` позволяет обратиться к списку всех открытых книг Excel, а его функция `Add` создает новую пустую книгу и возвращает её диспетчеризуемый интерфейс. Результат этого вызова запомним в переменной `WorkBook` также типа `Variant`. Рабочие книги Excel состоят из листов; новая книга обычно содержит 3 листа. В нашем случае мы будем передавать содержимое таблицы базы данных на первый лист. Для этого необходимо обратиться к массиву листов созданной книги и запомнить первый лист в переменной `Sheet` с помощью вызова `WorkBook.Sheets[1]`.

Обратите внимание, что при компиляции программы Delphi не может обеспечить проверку правильности обращения к объектам автоматизации. Это возможно сделать только на этапе непосредственно обращения к этим объектам. Поэтому обращение к объектам автоматизации через переменные типа `Variant` выполняется по сути в режиме интерпретации, т.е. работает относительно медленно.

Для обращения к ячейкам листа Excel необходимо использовать двумерный массив-свойство объекта-листа с помощью выражения `Sheet.Cells[Row, Col]`, где `Row` и `Col` задают соответственно номера столбца и строки внутри листа (нумерация ведется начиная с 1).

Далее необходимо все поля всех записей ранее выбранной таблицы базы данных скопировать в лист Excel.

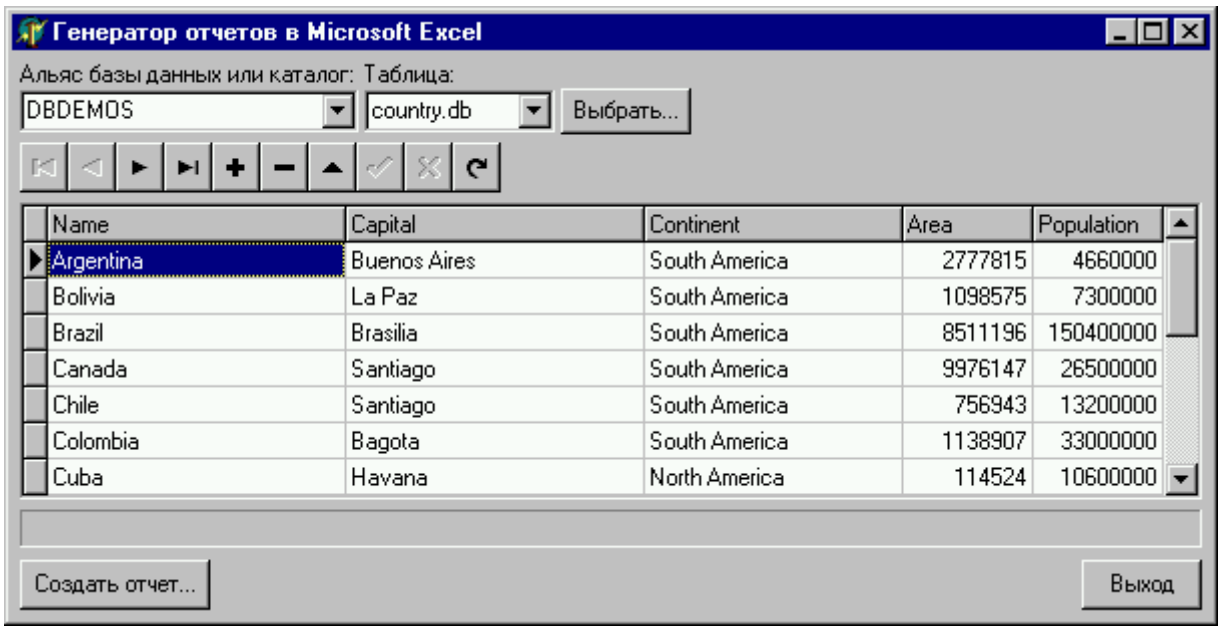
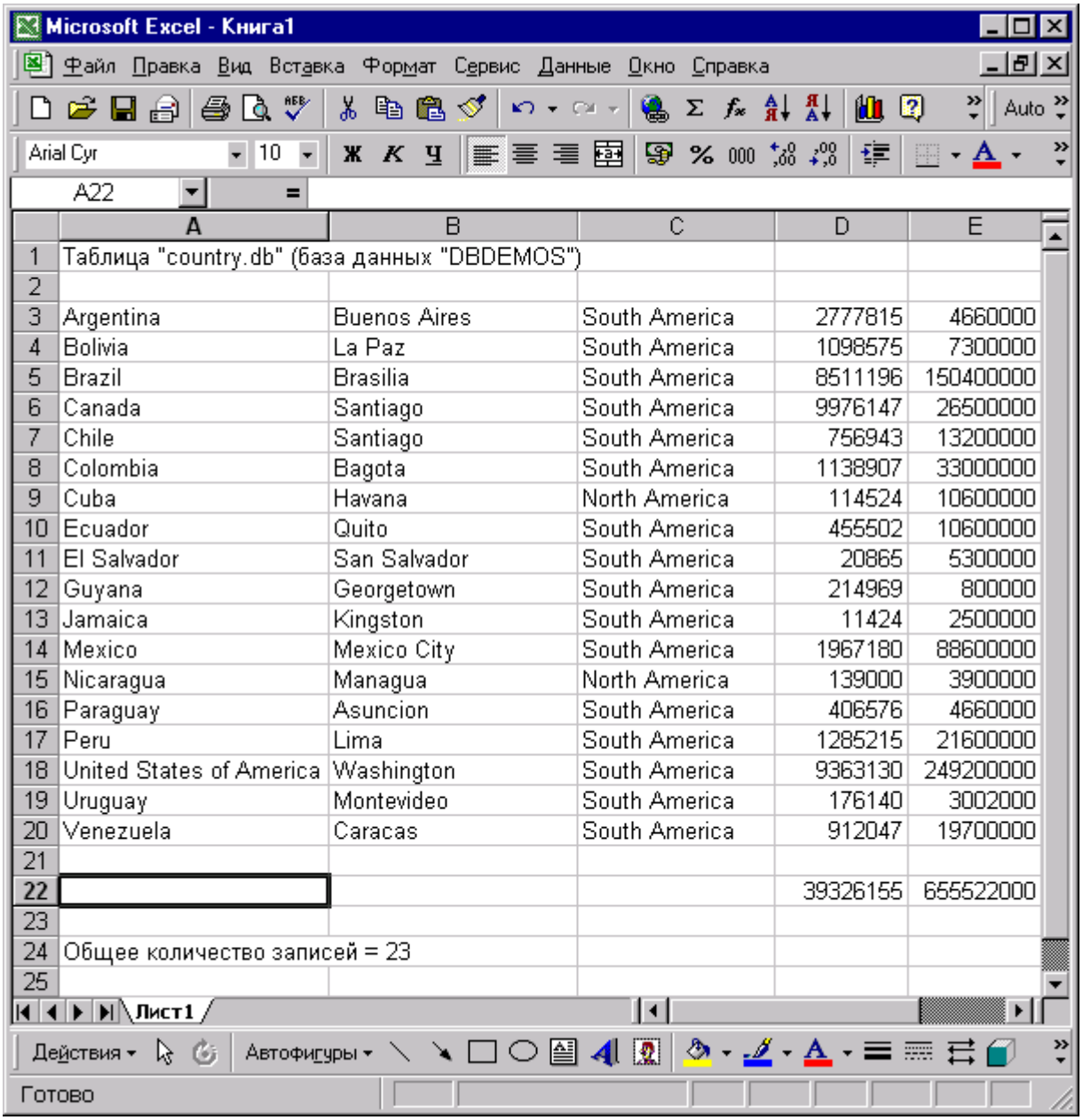


Рис. 36. Внешний вид запущенного приложения

По окончании формирования содержимого листа необходимо сделать текущей рабочую книгу в Excel, а сам Microsoft Excel вывести на передний план.

В заключение обратим внимание, что возможно обращение к программе-серверу автоматизации, размещенной на другом компьютере в сети, при этом в созданной программе практически ничего менять не надо, кроме процедуры подключения к серверу, где дополнительно надо указать имя компьютера.

На рис. 36 приведен внешний вид запущенного приложения после того, как окно было несколько растянуто. На рис. 37 приведен внешний вид сгенерированного с помощью Microsoft Excel отчета по таблице базы данных. В листинге 20 приведен текст файла модуля.



	A	B	C	D	E
1	Таблица "country.db" (база данных "DBDEMOS")				
2					
3	Argentina	Buenos Aires	South America	2777815	4660000
4	Bolivia	La Paz	South America	1098575	7300000
5	Brazil	Brasilia	South America	8511196	150400000
6	Canada	Santiago	South America	9976147	26500000
7	Chile	Santiago	South America	756943	13200000
8	Colombia	Bagota	South America	1138907	33000000
9	Cuba	Havana	North America	114524	10600000
10	Ecuador	Quito	South America	455502	10600000
11	El Salvador	San Salvador	South America	20865	5300000
12	Guyana	Georgetown	South America	214969	800000
13	Jamaica	Kingston	South America	11424	2500000
14	Mexico	Mexico City	South America	1967180	88600000
15	Nicaragua	Managua	North America	139000	3900000
16	Paraguay	Asuncion	South America	406576	4660000
17	Peru	Lima	South America	1285215	21600000
18	United States of America	Washington	South America	9363130	249200000
19	Uruguay	Montevideo	South America	176140	3002000
20	Venezuela	Caracas	South America	912047	19700000
21					
22				39326155	655522000
23					
24	Общее количество записей = 23				
25					

Рис. 37. Сгенерированный с помощью Microsoft Excel отчет по таблице

Листинг 20. Текст главного модуля MainUnit.pas

```
unit MainUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  OleCtnrs, StdCtrls, Grids, DBGrids, Db, DBTables, ExtCtrls, DBCtrls;

type
  TMainForm = class(TForm)
    ComboBoxDatabase: TComboBox;
    LabelBoxDatabase: TLabel;
    ComboBoxTable: TComboBox;
    LabelBoxTable: TLabel;
    DBGrid: TDBGrid;
    DataSource: TDataSource;
    Table: TTable;
    ButtonGenerate: TButton;
    ButtonExit: TButton;
    PanelStatus: TPanel;
    DBNavigator: TDBNavigator;
    ButtonFolder: TButton;
    OpenDialog: TOpenDialog;
    procedure ComboBoxDatabaseChange(Sender: TObject);
    procedure ComboBoxTableChange(Sender: TObject);
    procedure ButtonFolderClick(Sender: TObject);
    procedure ButtonGenerateClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ButtonExitClick(Sender: TObject);
  private
    procedure OpenTable; // Открыть выбранную таблицу
  end;

var
  MainForm: TMainForm;

implementation

{$R *.DFM}

uses
  ComObj;

procedure TMainForm.FormCreate(Sender: TObject);
begin // Получаем список всех зарегистрированных баз данных
  Session.GetDatabaseNames(ComboBoxDatabase.Items);
end;

procedure TMainForm.ButtonExitClick(Sender: TObject);
begin // Выход из программы
  Close;
end;

procedure TMainForm.OpenTable;
begin // Открыть выбранную таблицу
  try
    PanelStatus.Caption:=''; // Очистить строку статуса
    Table.Close; // Закрыть ранее открытую таблицу
    Table.DatabaseName:=ComboBoxDatabase.Text; // Указать имя базы данных
    Table.TableName:=ComboBoxTable.Text; // Указать имя таблицы
```

```
    Table.Open; // Открыть таблицу
  except
    on E: Exception do // Вывести сообщение об ошибке в строку статуса
      PanelStatus.Caption:=E.Message;
  end;
end;

procedure TMainForm.ComboBoxDatabaseChange(Sender: TObject);
begin // Выбрана новая база данных из списка
  try // Для заданной базы данных извлекаем все имена таблиц
    Session.GetTableNames(ComboBoxDatabase.Text, '', True, True, ComboBoxTable.Items);
  except
    ComboBoxTable.Items.Clear;
  end;
  OpenTable;
end;

procedure TMainForm.ComboBoxTableChange(Sender: TObject);
begin // Выбрана новая таблица, открыть ее для просмотра
  OpenTable;
end;

procedure TMainForm.ButtonFolderClick(Sender: TObject);
begin // Нажать кнопка "Выбрать..."
  if OpenDialog.Execute then
  begin
    ComboBoxDatabase.Text:=ExtractFilePath(OpenDialog.FileName);
    ComboBoxTable.Text:=ExtractFileName(OpenDialog.FileName);
    OpenTable;
  end;
end;

procedure TMainForm.ButtonGenerateClick(Sender: TObject);
  function ColToStr(Col: integer): string;
  begin // Преобразовать номер колонки в её имя в терминологии Microsoft Excel
    Result:=Chr(ord('A')+(Col-1) mod 26);
    if Col>26 then Result:=Chr(ord('A')+(Col-1) div 26)+Result;
  end;
var Excel, WorkBook, Sheet: Variant; OldBookmark, S: string; Row, Col: integer;
begin // Нажать кнопка "Создать отчет..."
  if not Table.Active then
    ShowMessage('Таблица не указана')
  else
  try
    Screen.Cursor:=crHourGlass; // Выводим курсор в виде песочных часов
    try
      PanelStatus.Caption:='';
      try // Подключение к уже запущенному Microsoft Excel
        Excel:=GetActiveOleObject('Excel.Application');
      except
        try // Запуск Microsoft Excel и подключение к нему
          Excel:=CreateOleObject('Excel.Application');
        except
          raise Exception.Create('Ошибка доступа к Microsoft Excel (он не установлен?)');
        end;
      end;
      WorkBook:=Excel.WorkBooks.Add;  // Создание книги Microsoft Excel
      Sheet:=WorkBook.Sheets[1];      // Отчёт будем создавать на первой странице
      Sheet.EnableCalculation:=False; // Запретить внутренние вычисления Excel
      try
        Sheet.Cells[1,1]:=Format('Таблица "%s" (база данных "%s")',
```

```
        [Table.TableName, Table.DatabaseName]);
      Table.DisableControls; // Запретить работу визуальных компонент, связанных с таблицей
      try
        OldBookmark:=Table.Bookmark; // Запомнить предыдущее положение в таблице
        try
          // Передаем в Excel все записи
          Row:=3;
          Table.First; // Переместиться в таблице на начало
          while not Table.EOF do // Пока не конец таблицы делаем цикл
          begin
            for Col:=1 to Table.FieldCount do // В цикле по всем полям таблицы
              Sheet.Cells[Row, Col]:=Table.Fields[Col-1].DisplayText;
            Table.Next; // Переместиться в таблице на следующую запись
            inc(Row);
          end;
          if Row>3 then
          begin
            // Создаем формулы для суммирования числовых полей всех записей
            inc(Row);
            for Col:=1 to Table.FieldCount do
            begin
              S:=ColToStr(Col);
              if Table.Fields[Col-1] is TNumericField then
              begin // Подведем сумму для числовых полей
                Sheet.Cells[Row, Col].Formula:=Format('=Sum(%s3:%s%d)',[S,S,Row-1]);
                Sheet.Columns[Col].HorizontalAlignment:=xlHAlignRight;
              end
              else
                Sheet.Columns[Col].HorizontalAlignment:=xlHAlignLeft;
              // Указать ширину колонки
              Sheet.Columns[Col].ColumnWidth:=Table.Fields[Col-1].DisplayWidth;
            end;
            inc(Row);
          end;
          Sheet.Cells[Row+1, 1]:='Общее количество записей = '+IntToStr(Row);
        finally
          Table.Bookmark:=OldBookmark; // Восстановить предыдущее положение в таблице
        end;
      finally
        Table.EnableControls; // Разрешить работу связанных визуальных компонентов
      end;
    finally
      Sheet.EnableCalculation:=True; // Разрешить внутренние вычисления Excel
    end;
    WorkBook.Activate;   // Сделать созданную книгу текущей в Microsoft Excel
    Excel.Visible:=True; // Активизировать Microsoft Excel
  finally
    Screen.Cursor:=crDefault; // Восстановить курсор
  end;
except // Вывести сообщение об ошибке в строке статуса
  on E: Exception do PanelStatus.Caption:=E.Message;
end;
end;

end.
```

Вопросы и задания для самостоятельной работы

1. Для чего предназначены средства ActiveX-автоматизации операционной системы Windows?
2. Каким образом в Delphi производится подключение к программе-серверу автоматизации? Как производится отключение?
3. Почему при компиляции Delphi не выдает сообщения об ошибках при неправильном обращении к методам объектов автоматизации?
4. Откройте файл справки по программированию в Microsoft Excel, попробуйте вызвать какие-нибудь методы объектов и поизменять из программы на Delphi их свойства.

Учебное издание

Алексей Владимирович Скворцов
Поддубная Тамара Николаевна

Автоматизация ActiveX

(методические указания к
лабораторной работе № 9)

Томский государственный университет. Томск, 1999. – 12 с.