Томский государственный университет

Факультет информатики

А.В. Скворцов, Т.Н. Поддубная

Знакомство с Delphi

(методические указания к

лабораторной работе № 1)

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**.

Среда программирования Borland Delphi предназначена для быстрой разработки разнообразных программ для операционных систем Windows 95/98/NT. Этот продукт относится к классу так называемых визуальных средств разработки. Delphi позволяет создавать довольно сложные пользовательские интерфейсы, при этом написание программного кода выполняется во многих случаях автоматически, что значительно ускоряет и упрощает процесс разработки.

Основой визуальной разработки программ в Delphi является понятие компонента. Они могут быть визуальными, т.е. видимыми в окне на экране: строка ввода, текстовая метка, кнопка, список, компонента для отображения картинок, таблица и др. Существуют также невизуальные компоненты, т.е. выполняющие какие-то функции, но не имеющие прямого графического представления на экране. Это могут быть компоненты для создания меню, связи с базами данных, таймер и др.

В середине 1999 г. фирмой Inprise Corporation выпущена 5-я версия. Delphi является очень популярной средой визуальной разработки во всем мире (в настоящее время имеется около 3 миллионов зарегистрированных пользователей). Для этой системы существует множество дополнительных библиотек процедур, визуальных компонентов и их число постоянно увеличивается.

Цель работы

Начальное освоение визуальной среды программирования Delphi.

Задачи работы

1. Ознакомление с основными визуальными элементами Delphi – *формой* и ее *свойствами*, а также некоторыми часто используемыми компонентами: *метками, кнопками, строками ввода*.
2. Создание простейшей программы-калькулятора для вычисления суммы и произведения двух чисел.

Описание работы

При работе в среде Delphi создается достаточно большое количество различных файлов. Поэтому перед началом разработки программ рекомендуется создать на диске отдельную папку для каждой лабораторной работы, куда будут сохранятся все файлы проектов. Например, удобно на рабочем диске создать папку с именем Delphi, а внутри нее папки Lab1, Lab2 и т.д. для каждой лабораторной работы соответственно (рис. 1).

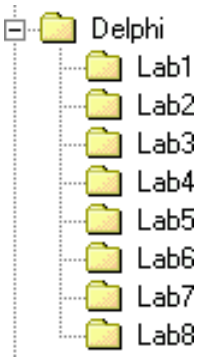

Рис. 1

Итак, вы впервые запустили Delphi. На экране появляется среда разработки, по умолчании состоящая из 4 окон (рис. 2). Самое верхнее окно содержит строку меню, панели инструментов с самыми основными командами и палитру компонентов. Левое окно – Object Inspector (инспектор объектов) – предназначено для изменения различных свойств редактируемых компонентов проекта Delphi. Находящееся в середине пустое окно с регулярной сеточкой точек является первой автоматически созданной формой проекта, на которой можно размещать различные компоненты, определяющие интерфейс разрабатываемого приложения. Находящееся под формой окно текстового редактора позволяет писать код на языке Object Pascal, определяющий функциональность программы.

После запуска Delphi для дальнейшей работы необходимо загрузить какой-либо старый проект через команду меню File|Open Project… для продолжения работы с ним либо воспользоваться созданным по умолчании новым (кроме того, его всегда можно создать с помощью команды меню File|New Application). Созданный новый проект необходимо сохранить на диске (командой File|Save), например, в нашем случае в каталог …\Delphi\Lab1. При сохранении Delphi сначала предложит сохранить мо-

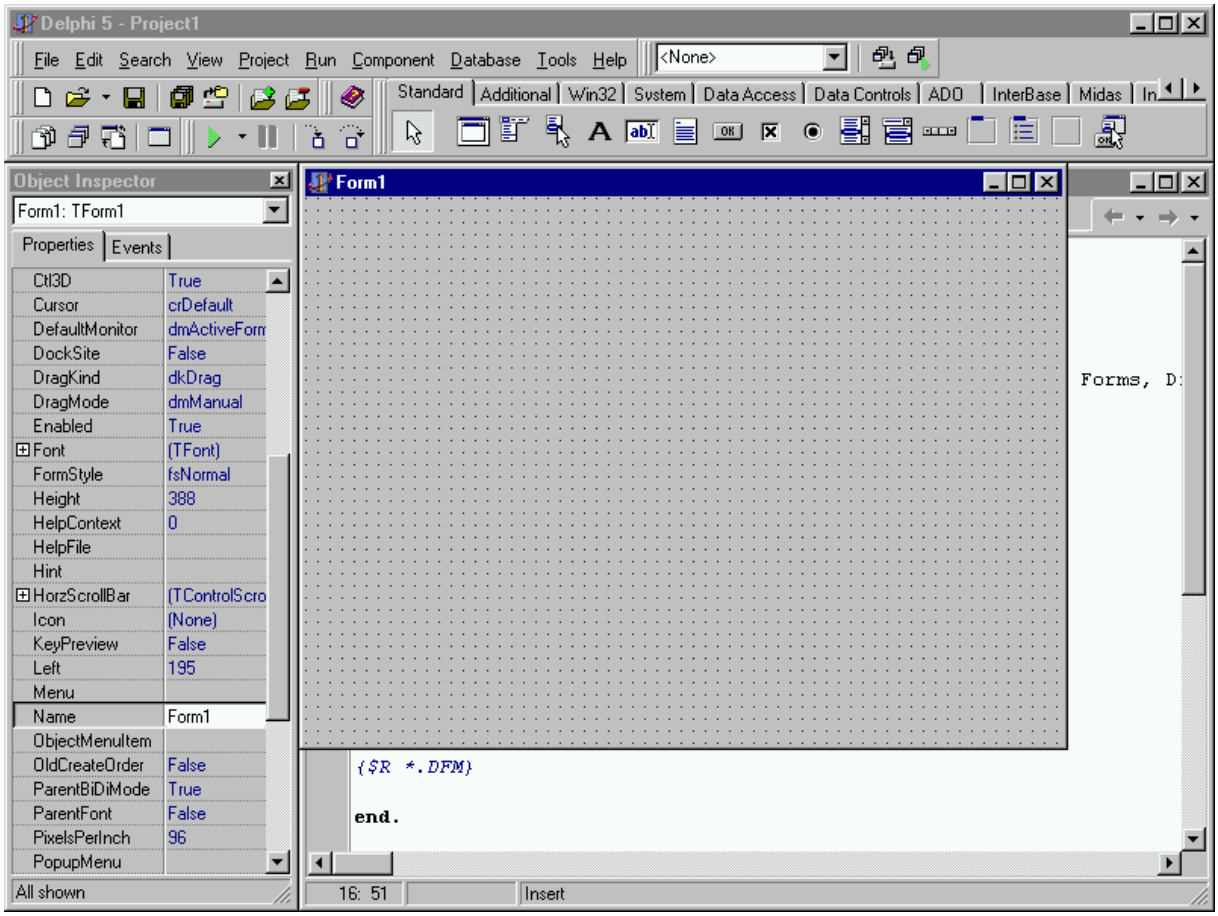

Рис. 2. Внешний вид среды визуальной разработки Delphi

дуль с формой под именем Unit1.pas, а проект – как Project1.dpr. Можно оставить эти имена, но лучше указать свои осмысленные имена, например MainUnit.pas и Lab1.dpr. Кстати, задаваемые имена файлов являются идентификаторами языка Object Pascal, поэтому при задании имён можно пользоваться только цифрами, латинскими буквами и символом подчерка «_», при этом первым символом не может быть цифра.

Замечание

Во время разработки программ регулярно пользуйтесь командой сохранения проекта File|Save All. *Это позволит избежать неприятных потерь файлов в результате случайного выключения питания или зависания компьютера.*

После того как вы открыли проект или создали новый, можно начинать размещать на форме необходимые компоненты, изменять их параметры и писать в редакторе кода программы.

Для начала попробуйте поменять различные свойства формы в инспекторе объектов на закладке Properties (другая закладка Events будет использована нами позднее для создания обработчиков событий). В первую очередь необходимо указать свойство `Name` формы. Оно будет использоваться в качестве имени переменной, ссылающейся на объект, представляющий форму в коде программы. Например, установите его значение в `MainForm`. Теперь измените свойство `Caption`. Оно устанавливает текст заголовка окна. Так как мы собираемся в данной работе создать калькулятор, то установите значение `Caption`, например, равным слову «Калькулятор».

Замечание

Не путайте свойства `Name` *и* `Caption`. *Свойство* `Name` *определяет имя переменной для ссылки в программе на объект, а* `Caption` *– текст заголовка визуального компонента.*

Каждый компонент имеет множество свойств. Некоторые свойства не требуют особых пояснений. Ясно, например, что свойство `Color` определяет цвет, `Width` – ширину, `Height` – высоту, `Name` – имя и т.д. Назначение многих других свойств не так очевидно. Их детальное описание представлено в электронной документации Delphi.

Замечание

Любая компонента имеет множество свойств. Хотя большинство из них не понадобится для выполнения данной работы, ознакомьтесь с ними. При необходимости пользуйтесь электронной подсказкой Delphi, вызываемой по F1.

Многие свойства являются составными, при этом в инспекторе объектов слева от названия свойства отображается знак «+». Увидеть все под-

множество можно, если сделать двойной щелчок на имени этого свойства, что приводит к раскрытию списка подсвойств. Таково, например, свойство `Font`, определяющее характеристики шрифта и имеющееся у большинства компонентов, работающих с текстом. Дополнительно у свойства `Font` имеется справа от значения свойства небольшая кнопка с многоточием. Если на ней щелкнуть мышкой, то это приведет к появлению диалогового окна, в котором можно задать такие характеристики шрифта, как его тип, начертание, размер, цвет.

При разработке приложений в среде Windows очень важно создавать такие экранные формы, которые были бы похожи на остальные окна Windows. Это позволит пользователю, привыкшему работать в Windows, легко научиться работать с вашей программой. Поэтому при создании форм желательно не изменять установленные по умолчании значения характеристик стандартных визуальных компонентов, например цвета (например, свойство `Color` формы), параметры шрифта (свойство `Font`), не менять без надобности стандартные размеры кнопок, строк ввода и т.д.

Замечание

Не забывайте, что ваши приложения должны быть похожи на остальные программы Windows. Поэтому не меняйте без особой нужды стандартные значения визуальных свойств (цвета, шрифты, размеры).

При соблюдении всех этих рекомендаций экранные формы созданного вами приложения сами будут менять цвета, шрифты, размеры своих компонентов в зависимости от системных установок, сделанных пользователем централизованно для всех приложений Windows (для этого можно воспользоваться пунктом Дисплей в Панели управления Windows).

Итак, приступим к размещению на форме компонентов, необходимых для работы нашего первого приложения. На рис. 3 показан внешний вид формы во время разработки. Мы видим, что необходимо разместить на форме 5 текстовых меток с надписями «Введите два числа:», «Сумма», «Произведение» и двумя «0», две строки ввода и две кнопки «Вычислить» и «Выход». Для этого надо выбрать на палитре компонентов необходимые элементы, щелкнуть по ним мышкой и затем щелкнуть мышкой в место их расположения на форме. При необходимости компоненты можно будет переместить на новое место и изменить их размер.

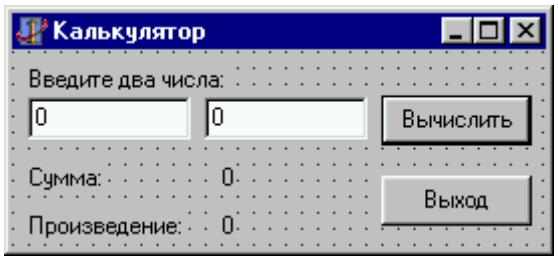

Рис. 3. Калькулятор во время разработки

Таблица 1. Значения основных устанавливаемых свойств.

Свойство	Значение	Комментарий
MainForm: TMainForm		
Name	MainForm	Имя переменной для ссылки на форму
Caption	Калькулятор	Заголовок формы
BorderStyle	bsDialog	Запрещается растяжение формы
Edit1, Edit2: TEdit		
Text	0	Вводимые значения по умолчанию
ButtonCalc: TButton		
Caption	Вычислить	Текст на кнопке
Default	True	Указывается, что нажатие клавиши Enter в любом месте формы, кроме кнопок, трактовалось как нажатие на эту кнопку
ButtonExit: TButton		
Caption	Выход	Текст на кнопке
Cancel	True	Указывается, что нажатие клавиши Esc в любом месте формы трактуется как нажатие на эту кнопку
Label1: TLabel		
Caption	Введите два числа:	Текст приглашения для ввода чисел
Label2: TLabel		
Caption	Сумма:	Комментарий к выводимому значению
Label3: TLabel		
Caption	Произведение:	Комментарий к выводимому значению
LabelSum: TLabel		
Caption	0	Значение суммы по умолчанию
LabelProduct: TLabel		
Caption	0	Значение произведения по умолчанию

После размещения компонентов им следует задать необходимые свойства. Для этого надо мышкой выделить на форме нужный компонент, после чего указать его свойства в инспекторе объектов. Для изменения

свойств формы нужно указать мышкой в форму мимо всех других компонент, либо нажать клавишу Esc. В табл. 1 кратко приведен список свойств различных компонентов формы, требующих установок.

Следующим шагом при разработке нашей программы будет написание обработчиков событий различных компонентов. В Delphi большинство компонентов могут порождать различные события в ответ на действия пользователя или в ответ на системные сообщения Windows. Примерами событий могут быть, например, перемещение мышки, нажатие кнопок мышки или клавиш на клавиатуре, открытие и закрытие окна и т.д.

Для создания обработчика какого-либо события необходимо выбрать мышкой нужный компонент, выбрать в инспекторе объектов закладку Events, найти требуемое событие и дважды щелкнуть на нем мышкой.

При создании нового обработчика Delphi создает в программном модуле, соответствующем данной форме, пустую заготовку процедуры обработки события примерно в таком виде:

> ***Замечание***
>
> *Не пишите пустые заготовки сами – они не будут вызываться при выполнении программы!*

```
procedure T<Имя_формы>.<Имя_компонента><Имя_события>(Sender: TObject);
begin

end;
```

Пока эта процедура пустая, но в дальнейшем она должна быть заполнена программистом необходимым кодом, который будет вызываться в ответ на указанное событие. При этом в форме в качестве специального свойства компонента будет установлено соответствие между выбранным событием и именем созданной процедуры. Поэтому, если самостоятельно написать в тексте кода соответствующие процедуры-обработчики без инспектора объектов, они никогда не будут вызваны. Однако, если у вас уже написаны нужные процедуры, то вы можете в инспекторе объектов указать это имя для нужного события с помощью выпадающего списка, появляющегося при нажатии кнопки справа от имени события (рис. 4).

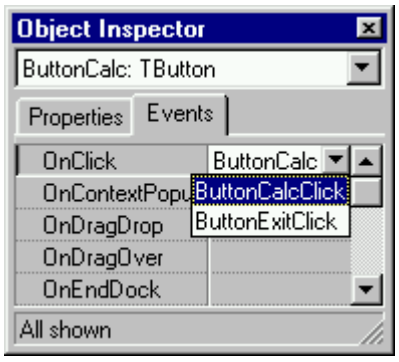

Рис. 4. Выбор обработчика события

В нашем приложении необходимо создать два обработчика события в ответ на нажатия кнопок `ButtonCalc` («Вычислить») и `ButtonExit` («Выход»). Такое событие для компонента типа `TButton` называется `OnClick`. В ответ на двойной щелчок в инспекторе объектов на месте значения обработчика этого события Delphi сформирует следующие заготовки:

```
procedure TMainForm.ButtonCalcClick(Sender: TObject);
begin

end;

procedure TMainForm.ButtonExitClick(Sender: TObject);
begin

end;
```

Формальный параметр Sender является указателем на породивший событие компонент. Он может использоваться в случае, если один обработчик событий назначен для нескольких разных компонентов. В данной работе он нам не понадобится, но будет использоваться в последующих.

Итак, теперь осталось заполнить полученные заготовки обработчиков необходимым кодом. В листинге 1 приведен полный текст программного модуля вместе с обработчиками событий.

Замечание

Не создавайте все пустые заготовки сразу. При сохранении формы Delphi найдет все пустые обработчики событий, удалит их, и вам придется их создавать заново.

В первом обработчике – процедуре ButtonCalcClick – выполняется извлечение чисел из строк ввода Edit1 и Edit2, которые затем складываются и перемножаются и выводятся в метках LabelSum и LabelProduct. Обратите внимание, что строки ввода типа TEdit предназначены для ввода текстовых значений, поэтому эти значения необходимо преобразовать в числа. Это выполняется с помощью функции StrToFloat. Так как не любая строка может быть преобразована в число, то при вызове данной функции возможно возникновение ошибок, называемых в Delphi исключениями (exception). Примерами исключений могут служить арифметические ошибки (переполнение, деление на ноль, извлечение квадратного корня из отрицательного числа), файловые ошибки (попытка открытия несуществующего файла, попытка записи в защищенный от записи файл), системные ошибки (обращение в памяти по несуществующему адресу, переполнение системного стека, нехватка памяти) и т.д. В нашем случае функция StrToFloat может вызвать ошибку преобразования строки в число, если в строке присутствуют недопустимые символы (например, недопустимы следующие строки: «0,12,1», «1-2», «ABC», «»).

При возникновении любого исключения дальнейшее выполнение кода программы прерывается и управление передается ближайшему установленному обработчику исключений. Если он не установлен, то выполнение всех процедур останавливается и на экран выводится сообщение об ошибке.

Для отлавливания исключений в языке Object Pascal служит конструкция **try...except...end**. Она гарантирует, что если в блоке кода между **try** и **except** возникнет исключение, то управление будет передано обработчику исключений между **except** и **end**. Если же исключений не возникнет, то обработчик не будет вызван.

Другой вид конструкции для работы с исключениями – конструкция **try...finally...end**. Она гарантирует, что код между **finally** и end будет выполнен вне зависимости от того, возникали ли какие-либо ошибки в участке кода между **try** и **finally**.

В нашем случае в качестве обработчика ошибок преобразования строки в число имеет смысл указать действие о необходимости переключиться на строку ввода, содержащую ошибочное число (команда `Edit1.SetFocus`), выдать сообщение об ошибке (команда `ShowMessage('Ошибка в первом числе');`) и прекратить дальнейшее выполнение обработчика события (команда `exit;`).

Если оба преобразования строк в числа пройдут успешно, то далее можно выполнить сложение и умножение полученных чисел, а затем вывести их в текстовые метки `LabelSum` и `LabelProduct`. Эти метки имеют свойство `Caption` строкового типа, выводимое на экран. Поэтому после выполнения арифметических операций необходимо преобразовать полученные числа в строковое представление с помощью функции `FloatToStr`.

Второй обработчик – процедура `ButtonExitClick` – предназначен для закрытия формы и завершения работы. Для этого достаточно вызвать метод `Close` формы.

После написания обработчиков событий программа готова к выполнению. Это можно сделать с помощью команды меню Run|Run или с помощью клавиши F9. На рис. 5 приведен внешний вид готового приложения. Как и в любом приложении Windows, для перехода по элементам можно использовать клавишу Tab, для вычисления суммы можно воспользоваться клавишей Enter, а для выхода из калькулятора – клавишей Esc.

В заключение кратко рассмотрим текст всего сформированного программного модуля в целом.

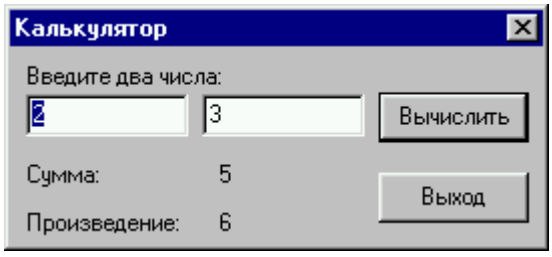

Рис. 5. Внешний вид готового приложения

Любой модуль (**unit**) на языке Object Pascal состоит из двух секций: интерфейсной и реализации. Их начала соответственно определяются ключевыми словами **interface** и **implementation**. Ключевое слово **uses** определяет список других используемых модулей.

Для удобства программиста Delphi включает в код программы небольшой список основных наиболее используемых при разработке модулей, таких как `Windows` (процедуры для работы с Windows на нижнем уровне), `SysUtils` (базовые процедуры Delphi), `Classes` (определения основных классов), `Graphics` (работа с графикой), `Controls` (базовые определения визуальных компонент), `Forms` (работа с формами), `Dialogs` (процедуры для вывода простейших диалоговых форм).

Любая форма с точки зрения Delphi является классом языка Object Pascal, поэтому после ключевого слова **`type`** идёт описание класса, соответствующего форме, а после слова **`var`** следует объявление переменной, ссылающейся на форму. Так как форма является классом, то нам ничто не мешает создавать по мере необходимости любое количество реальных экземпляров форм и выводить на экран одновременно множество одинаковых окон.

В начале секции **`implementation`** присутствует директива *`{$R *.DFM}`*, которая говорит компилятору о необходимости связать в конечном выполняемом файле Windows (с расширением .exe) модуль кода (файл с расширением .pas) и файл описания формы (расширение .dfm).

Листинг 1. Текст программного модуля MainUnit.pas

```
unit MainUnit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TMainForm = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    ButtonCalc: TButton;
    Label2: TLabel;
    Label3: TLabel;
    LabelSum: TLabel;
    LabelProduct: TLabel;
    ButtonExit: TButton;
    procedure ButtonCalcClick(Sender: TObject);
    procedure ButtonExitClick(Sender: TObject);
  end;

var
  MainForm: TMainForm;

implementation

{$R *.DFM}

procedure TMainForm.ButtonCalcClick(Sender: TObject);
var A, B: double;
begin // Нажата кнопка "Вычислить"
  try
    A:=StrToFloat(Edit1.Text); // Преобразовать текст в число
  except
    Edit1.SetFocus;
    ShowMessage('Ошибка в первом числе');
    exit;
  end;
  try
    B:=StrToFloat(Edit2.Text); // Преобразовать текст в число
  except
    Edit2.SetFocus;
    ShowMessage('Ошибка во втором числе');
    exit;
  end;
  LabelSum.Caption:=FloatToStr(A+B);
  LabelProduct.Caption:=FloatToStr(A*B);
end;

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

end.
```

Вопросы и задания для самостоятельной работы

1. Какие окна присутствуют по умолчании на экране в момент начала работы над новым проектом в Delphi и каковы их функции?
2. Что такое Properties и Events в окне инспектора объектов?
3. В чем разница между свойствами `Caption` и `Name`?
4. Что означают значок «+» перед названием свойства в окне инспектора объектов и кнопка с многоточием в строке свойства?
5. Какие файлы создает Delphi при работе с проектом? Каково их назначение? Где они сохраняются?
6. Какой алгоритмический язык используется для программирования в Delphi?
7. Каким образом в Delphi создается стандартная заготовка для обработчиков событий?
8. Какие функции используются для преобразования строковых значений в численные и наоборот.
9. Что такое исключения и как они используются?
10. Что будет, если в программе ввести в строку ввода не число?
11. Модифицируйте код программы для следующего случая: после ввода двух чисел программа определяет, какое из введенных чисел больше, и если больше первое число, то вычисляется разность введенных чисел, в противном случае вычисляется их сумма. Программа также должна отреагировать на ситуацию, когда числа равны друг другу. Добавьте на форму необходимые компоненты для отображения информации о введенных числах.

Литература

1. Архангельский А.Я. Программирование в Delphi 5. – М.: ЗАО «БИНОМ», 2000. – 1072 с.
2. Архангельский А.Я. 100 компонентов общего назначения библиотеки Delphi 5. – М.: ЗАО «БИНОМ», 1999. – 272 с.
3. Баас Р., Фервой М., Гюнтер Х. Delphi 5: для пользователя / Пер. с нем. – Киев: BHV, 2000. – 496 с.
4. Бобровский С. Delphi 5: начальный курс. – М.: Изд-во «ДЕСС», 1999. – 270 с.
5. Епанешников А.М., Епанешников В.А. Delphi 5. Язык Object Pascal. – М.: ДИАЛОГ-МИФИ, 2000. – 368 с.

Учебное издание

Алексей Владимирович Скворцов
Поддубная Тамара Николаевна

Знакомство с Delphi

(методические указания к
лабораторной работе № 1)

Томский государственный университет. Томск, 1999. – 16 с.