ИНФОРМАТИКА

Е. В. Санников

«СОЛОН»

Курс практического программирования в Delphi

Объектно-ориентированное программирование

– курс-практикум для начинающих программистов
– от простого к сложному
– 15 практических работ
– формирование навыков при разработке приложений
– книга для тех, кто хочет научиться объектно-ориентированному программированию!

ISBN 5-91359-122-7
9785913591227

ЭЛЕКТИВНЫЙ КУРС ✸ ПРОФИЛЬНОЕ ОБУЧЕНИЕ

УДК 681.3
ББК 32.973-018
С 185

Санников Е. В.
Курс практического программирования в Delphi. Объектно – ориентированное программирование. — М.: СОЛОН-ПРЕСС, 2016. — 188 с.: ил.

Данный курс-практикум рассчитан преимущественно на студентов информационных и технических специальностей, а также на начинающих программистов, желающих научиться разработке приложений в среде Delphi и понять суть объектного программирования. Также практикум может использоваться в качестве элективного курса у школьников старших классов.

Весь процесс создания приложений расписан очень подробно и иллюстрирован рисунками, а приведенный программный код в большинстве случаев снабжен комментариями.

Особенное внимание уделено именно практической части и формированию умений при разработке приложения.

Практикум состоит из 15 практических работ, основанных на принципе «от простого к сложному».

Сайт журнала «Ремонт & Сервис»: www.remserv.ru
Сайт издательства «СОЛОН-ПРЕСС»: www.solon-press.ru

КНИГА — ПОЧТОЙ

Книги издательства «СОЛОН-ПРЕСС» можно заказать наложенным платежом (оплата при получении) по фиксированной цене. Заказ оформляется одним из трех способов:

1. Послать открытку или письмо по адресу: 123001, Москва, а/я 82.
2. Оформить заказ можно на сайте **www.solon-press.ru** в разделе «Книга — почтой».
3. Заказать по тел.: (499) 254-44-10, (499) 795-73-26.

Каталог издательства высылается по почте бесплатно.

При оформлении заказа следует правильно и полностью указать адрес, по которому должны быть высланы книги, а также фамилию, имя и отчество получателя. Желательно указать дополнительно свой телефон и адрес электронной почты.

Через Интернет вы можете в любое время получить свежий каталог издательства «СОЛОН-ПРЕСС», считав его с адреса **www.solon-press.ru/kat.doc**.

Интернет-магазин размещен на сайте **www.solon-press.ru**.

По вопросам приобретения обращаться:
ООО «АЛЬЯНС-БУКС»
Тел: (499) 725-54-09, 725-50-27,
www.alians-kniga.ru

ISBN 978-5-91359-122-7

© Обложка, «СОЛОН-ПРЕСС», 2016
© Санников Е. В., 2016

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ 5

ПРАКТИЧЕСКАЯ РАБОТА №1 6

1.1. Основы работы в среде объектно-ориентированного программирования Delphi7 6

ПРАКТИЧЕСКАЯ РАБОТА №2 20

2.1. Работа с буфером обмена, рисование мышью, контекстное меню, компонент Memo1 и Image1 20
2.2. Рисование мышкой на форме. Контекстное меню 22
2.3. Компонент Memo 25

ПРАКТИЧЕСКАЯ РАБОТА № 3 28

3.1. Перемещение мышью панели Panel1 на форме 28
3.2. Работа с таймером на примере программы «Бильярд» 29
3.3. Изменим надпись на кнопке «Пуск» 30

ПРАКТИЧЕСКАЯ РАБОТА №4 32

4.1. Построение графиков в среде Delphi 32
4.2. Моделирование бегущих волн 35

ПРАКТИЧЕСКАЯ РАБОТА №5 41

5.1. Работа с файлами (на примере текстового). Создание текстового файла. Запись информации в текстовый файл 41
5.2. Чтение из файла. Открытие файла в режиме чтения 44

ПРАКТИЧЕСКАЯ РАБОТА № 6 48

6.1. Краткий обзор некоторых стандартных компонентов Delphi 48
6.2. Исследуем компоненты PageControl1, ListBox1, ComboBox1, а также еще ряд полезных компонентов 51

ПРАКТИЧЕСКАЯ РАБОТА № 7 57

7.1. Процедуры и функции 57
7.2. Перегрузка функций 65

ПРАКТИЧЕСКАЯ РАБОТА № 8 69

8.1. Создание приложения «МикроПроводник». Создание заставки 69
8.2. Запуск Программ 73
8.3. Окно заставки 74

ПРАКТИЧЕСКАЯ РАБОТА № 9 76

9.1. Виртуальные и ASC II коды 76

ПРАКТИЧЕСКАЯ РАБОТА № 10 82

10.0. Сообщения и их обработка 82
10.1. Исследуем стандартный виртуальный метод WndProc 84
10.2. Класс TApplication 88
10.3. Собственные обработчики событий. Функции SendMessage и PostMessage 91
10.4. Создание своего собственного сообщения 94

ПРАКТИЧЕСКАЯ РАБОТА № 11 96

11.0. Базы Данных 96
11.1. Работа с мастером Баз Данных 96
11.2. Создание Базы Данных с использованием стандартного приложения Database Desktop 101
11.3. Работа с Базой Данных непосредственно в среде Delphi 104
11.4. Самостоятельные задания по работе с базами данных в Delphi 114

ПРАКТИЧЕСКАЯ РАБОТА № 12 115

12.0. Библиотеки 115
12.1. Создание библиотеки с использованием Dll Wizard 115
12.2. Библиотека хранящая объекты 119
12.3. Динамически подключаемые библиотеки 122

ПРАКТИЧЕСКАЯ РАБОТА №13 125

13.1. Создание собственного видеопроигрывателя 125
13.2. Вывод видео на стандартные компоненты Delphi. Описание свойства DeviceType компонента MediaPlayer1 128

ПРАКТИЧЕСКАЯ РАБОТА №14 132

14.1. Классы и объекты 132
14.2. Наследование классов, класс TObject, назначение конструктора Create. Статические и виртуальные методы. 140

ПРАКТИЧЕСКАЯ РАБОТА №15 157

15.1. Замена виртуальных методов. Цепочки Виртуальных методов. Директива reintroduce. Таблица виртуальных методов VMT 157
15.2. Перегрузка статических и виртуальных методов в разных взаимонаследующих классах .. 165
15.3. Инкапсуляция. Спецификаторы доступа public, private, protected 178
15.4. Свойства класса 181

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 187

ПРАКТИЧЕСКАЯ РАБОТА №1

1.1. Основы работы в среде объектно-ориентированного программирования Delphi7

Для обучения и освоения объектно-ориенитрованной среды программирования Delphi наиболее оптимальными версиями являются Delphi 6 или Delphi7. Установите какую-либо из этих версий на свой компьютер и можно приступать к освоению этой популярной, обладающей значительными возможностями, среде программирования. Если вы имеете некоторые навыки работы с языком программирования Pascal, то обучение пойдет гораздо быстрее, поскольку в среде Delphi программы разрабатываются на языке Object Pascal. Запускается среда объектно-ориентированного программирования Delphi7 также, как и обычное приложение. Из общего меню Windows последовательно выберите: «Программы» → «Borland Delphi7», а затем просто щелкните на надписи «Delphi7» из раскрывшегося списка (рис. 1.1).

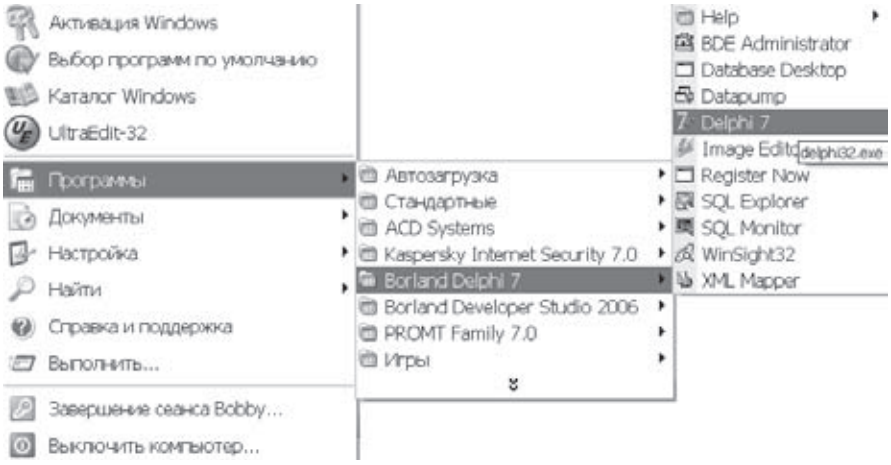


Рис. 1.1 Запуск Delphi7

Как только среда Delphi запустится, на экране сразу же появятся пять следующих окон (рис. 1.2):
главное окно – Delphi7;
окно стартовой формы – Form1;
окно редактора свойств и событий объектов – Object Inspector;
окно просмотра списка объектов – Object TreeView;
окно редактора кода – Unit1.pas.

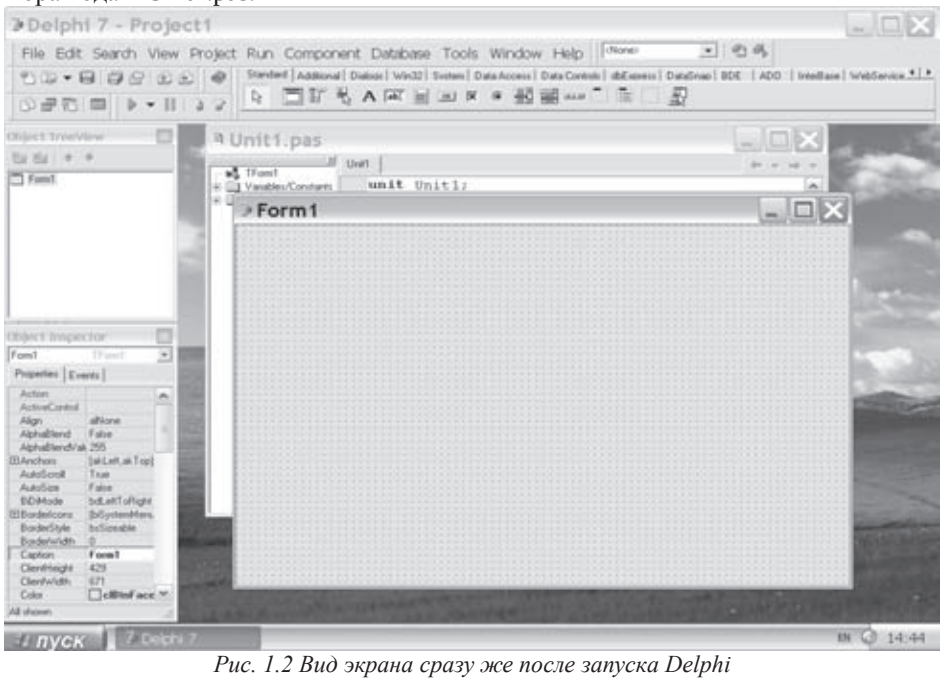


Рис. 1.2 Вид экрана сразу же после запуска Delphi

В главном окне (рис. 1.3) находится меню команд, панели инструментов и палитра компонентов (вкладки → «Standard», «Additional», «Win32» и т.д.). Палитра компонентов содержит компоненты, необходимые для разработки приложения. Часто компонент называют еще и объектом.

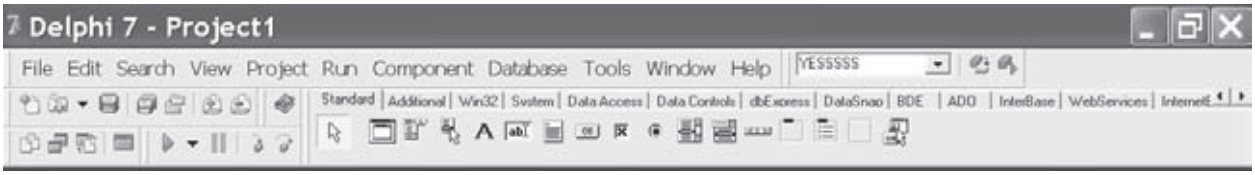


Рис. 1.3. Главное окно

Задание 1.1.0 **Закроем проект и создадим его заново в среде** Delphi.

В данном случае, после запуска Delphi, появившееся у вас на экране окно с заголовком Form1 и окно редактора кода с заголовком Unit1.pas представляют собой проект по умолчанию.

Закройте проект, последовательно выбрав «File» → «Close All», а затем снова создайте новый проект (приложение). Для этого последовательно выберите «File» → «New» → «Application».

Перед вами снова появится окно стартовой формы Form1, которое представляет собой заготовку главного окна разрабатываемого приложения (проекта) и окно с заголовком Unit1.pas.

Каждый объект в объектно-ориентированном программировании (например, объект с именем Form1) описывается определенным классом (например форма Form1 описывающая окно приложения, описывается классом TForm1). Понятие «класс» является логической абстракцией, описывающей некоторый объект.

Именно объект, созданный на основе некоторого класса, физически занимает определенную область в оперативной памяти компьютера и производит определенные действия с помощью своих методов над своими данными.

Фактически класс – это тип данных, определенный пользователем, а объект – его экземпляр. Часто говорят, что объект является экземпляром класса. В некотором смысле объект – это «воплощение класса в реальность».

Для сравнения можно сказать, что класс – это форма для печенья, а объект – само печенье [1]. Часто объектом в Delphi выступает обыкновенное окно операционной системы windows, именно такое окно и описывает класс TForm, то есть внешний вид окна, цвет заливки, надпись заголовка в окне и т.д.

У класса имеются свои поля и методы. В сущности, поля класса – это данные любого определенного типа, принадлежащие классу, а методы класса – это процедуры и функции класса. Методы непосредственно исполняют некоторые действия, в том числе могут работать с полями.

Например некоторые поля класса TForm содержат информацию о внешнем виде окна (ширина, высота, цвет и т. д.). Поля могут содержать текстовую информацию, некоторый адрес в памяти, просто целое или вещественное число, то есть могут содержать значение почти любого типа определенного в Delphi.

Чтобы объявить в программе два объекта (две переменные объектного типа) – Form1 и Form2 класса TForm1, требуется записать следующее объявление: var Form1, Form2: TForm1.

В данном случае объекты Form1 и Form2 полностью описываются классом TForm1. На самом деле Form1 и Form2 не являются объектами, они являются лишь ссылками на объекты, то есть содержат адрес объектов в оперативной памяти, но для упрощения рассуждений их часто можно считать самими объектами.

Во многих случаях понятия «тип» и «класс» можно считать синонимами.

Классы могут наследоваться. Это значит, что, если мы хотим создать новый класс TForm1, являющийся наследником TForm, то следует написать инструкцию: TForm1 = class (TForm).

Таким образом, новый класс TForm1 вбирает в себя все свойства, возможности и функциональность класса TForm, а также может, и скорее всего будет (специально для этого он и создан), дополнять их своими новыми.

В данном случае говорят, что класс TForm1 является производным классом от класса TForm, а класс TForm – базовым по отношению к классу TForm1.

Можно сказать, что класс TForm является родительским для класса TForm1, который, в свою очередь, является его дочерним классом.

Все компоненты в Delphi **являются объектами, следовательно, описываются определенными классами**.

В среде Delphi имеется очень важное окно Object Inspector (рис. 1.4) – окно редактора свойств и событий объектов, предназначенное для редактирования значений свойств объектов [2, 3] (**вызывается при нажатии на** «F11»). В терминологии компонентно-визуального проектирования объектами являются диалоговые окна и элементы управления (поля ввода и вывода, командные кнопки, переключатели и др.).

Например, сама форма Form1 является объектом. И все компоненты, что находятся в палитре компонентов, также являются объектами. Именно их свойства и события может отображать окно Object Inspector. На протяжении всего времени разработки приложений в Delphi вам придется очень интенсивно им пользоваться.

Properties (**Свойства объекта**) – это характеристики, определяющие вид, положение и поведение объекта. Находятся в Object Inspector слева.

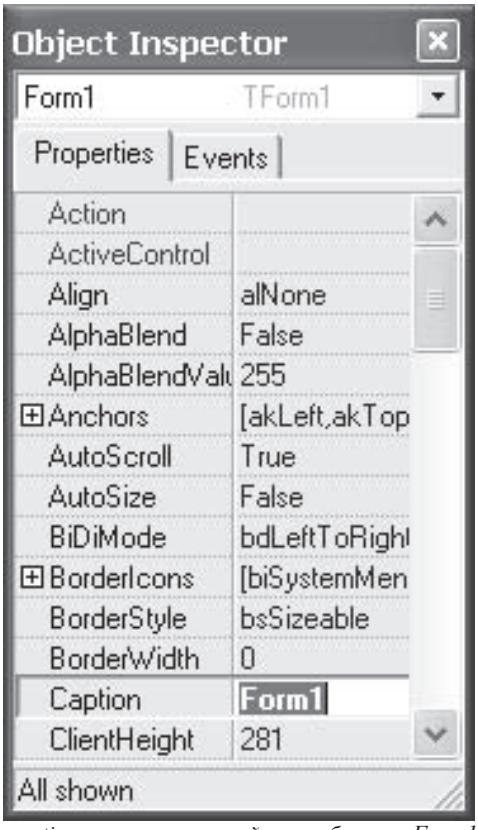


Рис. 1.4 На вкладке Properties перечислены свойства объекта-Form1 и указаны их значения

Задание 1.1.1 **Изменим некоторые свойства формы** – Form1

Активируйте окно Object Inspector, нажав «F11», и выберете там из выпадающего списка форму Form1 или просто один раз щелкните по форме (в Object Inspector установится значение Form1, как на рис. 1.4). Поменяйте в «Object Inspector» → «Properties» значения свойств Width и Height (ширину и высоту) формы, свойства Top и Left – положение формы на экране.

Измените значение свойства Caption (заголовок) на “Работа в Delphi”.

Обратите внимание на то, что изменяя значение свойства Caption вы не меняйте имя самого объекта. Посмотрите на свойство Name – вы обнаружите там «Form1». Нужно отметить особое свойство Name, имеющееся у всех компонентов Delphi. Name – это имя компонента, его идентификатор.

Аналогично настроив Object Inspector на любой другой компонент, можно менять значения его свойств.

Задание 1.1.2 **Кратко рассмотрим редактор кода**.

Помимо формы, имеется еще и редактор кода (рис. 1.5), там, где непосредственно находится текст программы. **Редактор кода предназначен для редактирования модуля**, **который содержится в файле с расширением** *.pas. **По умолчанию имя файла модуля** «Unit1.pas».

Нажмите «F12» и вы будете переключаться между редактором кода и формой.

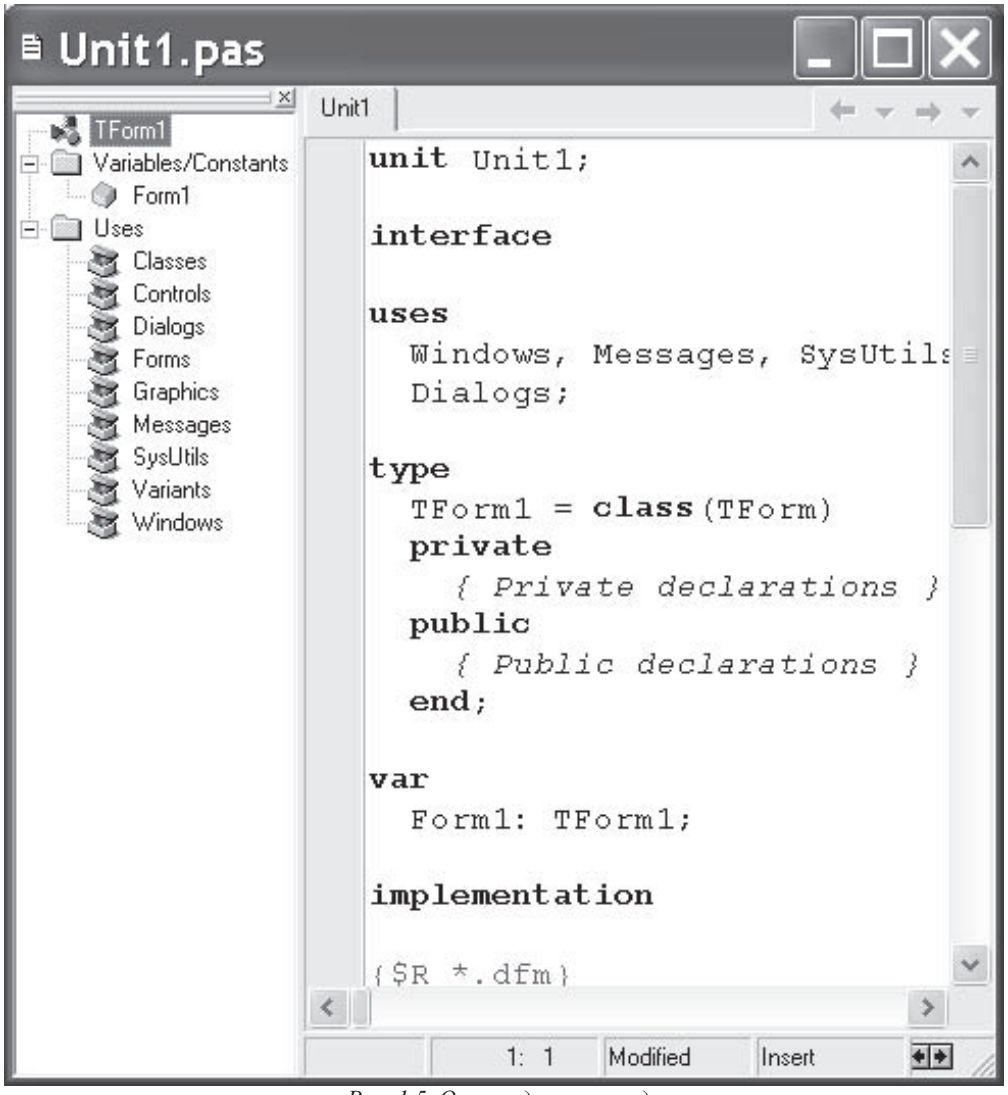


Рис. 1.5. Окно редактора кода

В окне редактора кода, которое можно увидеть, отодвинув в сторону окно формы, следует набирать текст программы. В начале работы над новым проектом окно редактора кода содержит сформированный средой Delphi шаблон программы.

В Delphi в основном создаются Windows-приложения, **часто одному приложению сопоставляется один проект, который хранится в одной папке**.

Задание 1.1.3 **Сохраним проект**.

Проект Delphi – **это папка, в которой хранится несколько файлов**.

Из них основные два файла – файл проекта (*.dpr), **в котором хранится информация о проекте в целом и файл модуля** (*.pas), **в котором находится программный код**.

Сохраняется проект путем последовательного сохранения этих двух файлов (файла проекта и файла модуля) в одну папку:

Последовательно выберите «File» → «Save All» или один раз щелкните мышкой на соответствующей пиктограмме на панели инструментов, обведенной кружком (рис. 1.6а)



а) Сохраняем проект б) Открываем проект

Рис. 1.6. Пиктограммы сохранения и открытия проекта

Перед вами появится диалоговое окно, предлагающее сохранить файл модуля unit1.pas в месте, определенном по умолчанию (рис. 1.7).

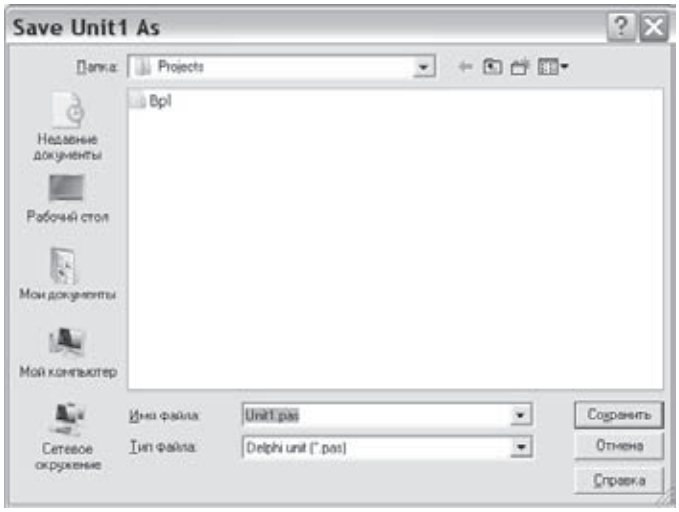


Рис. 1.7. Диалоговое окно, предлагающее сохранить вам файл модуля unit1.pas в месте, определенном по умолчанию. Не сохраняйтесь в этом месте, лучше выберите другое место

Однако вы, с помощью данного диалогового окна, выберете какое-нибудь другое место на диске, где должен располагаться ваш проект (например, «Рабочий стол» или «Диск С»), и создайте в этом месте – папку проекта, например, с именем «Proekt» *(щелкните на иконке «создание новой папки» или правой кнопкой мыши «создать папку»).* Затем зайдите в эту папку «Proekt» и, уже там, сохраните файл модуля unit1.pas, щелкнув один раз на кнопке «Сохранить» (рис. 1.8).

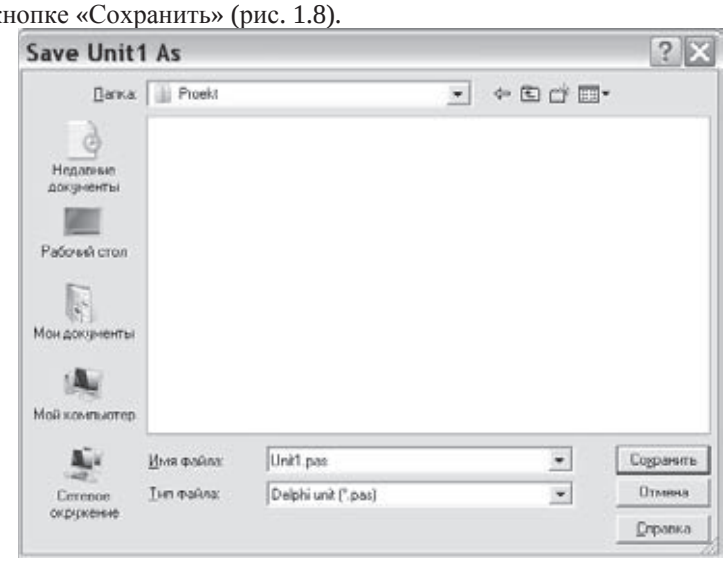


Рис. 1.8. Диалоговое окно сохранения файла модуля unit1.pas

После этого, сразу же, появится следующее диалоговое окно с предложением сохранить файл проекта project1.dpr в этой же папке.

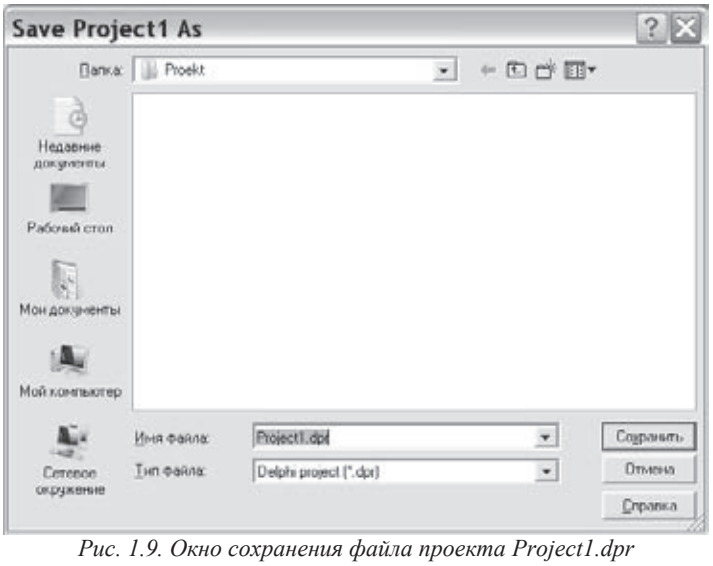


Рис. 1.9. Окно сохранения файла проекта Project1.dpr

Сохраните его в этой же папке.

Теперь, когда проект создан и основные файлы сохранены в одной папке, на форму можно добавлять остальные компоненты, менять программный код и для сохранения изменений уже просто нажимать на пиктограмме Save All.

Чтобы посмотреть содержимое файла проекта *.dpr и перемещаться между файлами модулей и файлом проекта, в окне редактора кода нажмите сочетание «Ctrl+F12» – появится следующее окно:

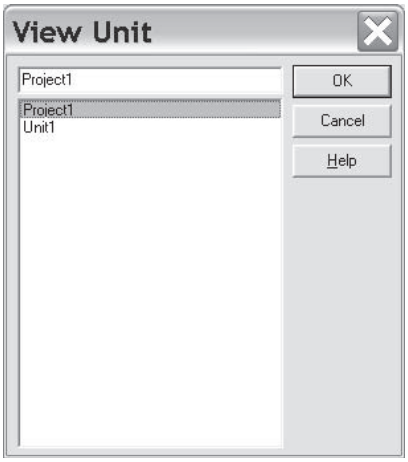


Рис. 1.10. Окно выбора файла модуля или файла проекта для его отображения в окне редактора кода

Выберите Project1 и нажмите OK. Перед вами в окне редактора кода появится содержимое файла проекта Project1.dpr.

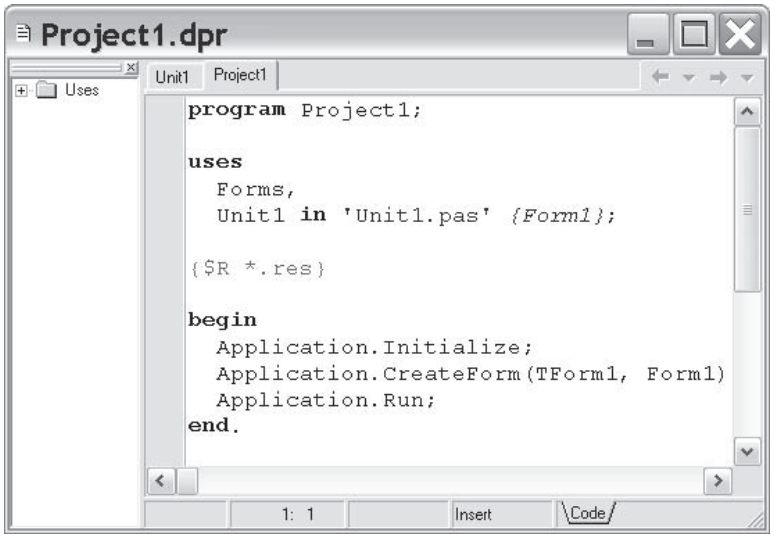


Рис. 1.11 Содержимое файла проекта

В этом файле, как и в unit1.pas, можно изменять программный код, но в основном при разработке несложных приложений его изменение не требуется. Например, при добавлении дополнительной формы в приложение среда Delphi сама внесет необходимый код в данный файл.

Задание 1.1.4 **Переместим на форму три компонента** Edit.

Наиболее часто используемые компоненты находятся на вкладке Standard (рис. 1.12).

Компонент Edit предназначен для ввода значений в его текстовое поле.



Рис. 1.12 Вкладка Standard содержит наиболее часто используемые компоненты

Для того чтобы добавить в форму компонент Edit1, необходимо в палитре компонентов Standard выбрать его, щелкнув один раз левой кнопкой мыши на его пиктограмме, далее подвести указатель мыши в ту точку

формы, в которой должен быть размещен левый верхний угол компонента, и еще раз щелкнуть левой кнопкой мыши по форме. В результате на форме появится компонент стандартного размера.



Рис. 1.13 Компонент Edit1 размещенный на форме

Каждый компонент в Delphi является объектом, и следовательно описывается классом. Подведите указатель мыши к компоненту Edit1, который уже на форме, и задержите его на пару секунд. Вы увидите надпись Edit1: TEdit. Следовательно, объект с именем Edit1 описывается классом TEdit.

Таким образом, имя переменной-объекта помещенного на форму состоит из 2-х частей:

«имя класса компонента без первой буквы T» + «порядковый номер», то есть в случае с Edit1 будет «Edit» + «1». А если добавить второй компонент класса Tedit, будет «Edit» + «2», то есть Edit2.

Добавьте компонент Edit2 и Edit3 на форму.

Название класса по принятым соглашениям предваряется буквой «T», так как оно происходит от слова «Type» – что означает тип или класс.

Программист путем изменения значения свойства Name может изменять имя компонента. В простых программах имена компонентов, как правило, не изменяют.

Задание 1.1.5 **Переместим компонент** Button1 **(кнопка) на форму**.

Выберете из палитры компонентов Standard, компонент Button1 и поместите его на форму. Этот компонент является кнопкой и описывается классом TButton. По умолчанию кнопка будет иметь имя Button1, это можно увидеть в свойстве name окна Object Inspector.

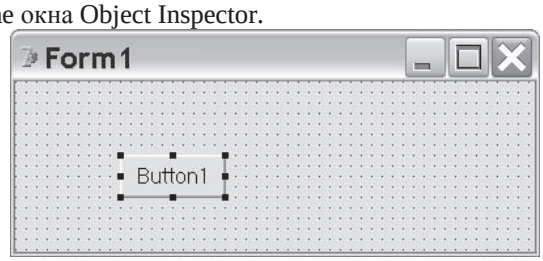
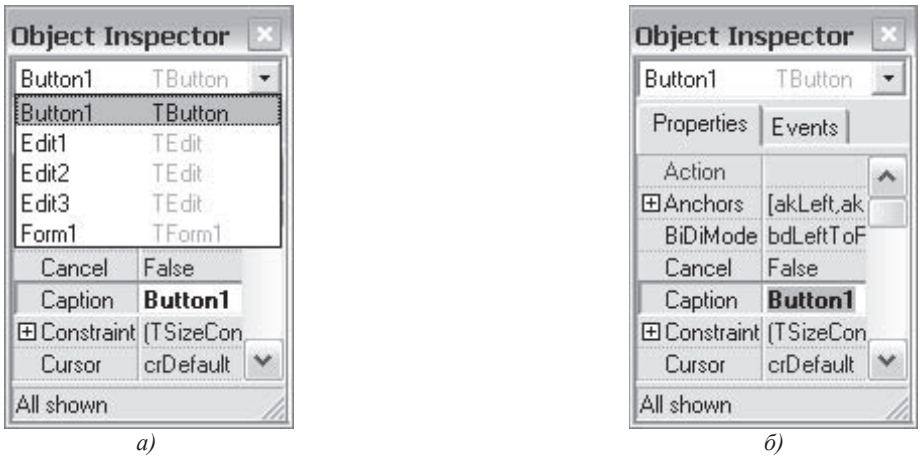


Рис. 1.14 Компонент Button1 размещенный на форме

Настройте Object Inspector (F11) на Button1. Для этого в верхней части Object Inspector из раскрывающегося списка просто выберете Button1.



а) *б)*

Рис. 1.15.
а) Раскрывающийся список Object Inspector; б) Object Inspector настроен на Button1

Или выберите одинарным щелчком Button1 на форме, что также сразу настроит Object Inspector на Button1.

Растяните кнопку пошире в длину. Посмотрите, как при этом изменится значение свойства Width кнопки, а также в высоту (свойство Height) в Object Inspector. Перемещайте Button1 по форме. Посмотрите, что

изменяются значения свойств Left и Top – определяющие координаты Button1 относительно левого верхнего угла формы. Попробуйте поменять эти значения вручную.

Поменяйте значения свойства Caption с «Button1» на «PUSK». Тем не менее, имя кнопки останется Button1, вы меняете лишь надпись на ней.

Некоторые свойства компонента Button1

Свойство	Описание
Name	**Имя компонента. Используется в программе для доступа к компоненту и его свойствам**
Caption	**Текст на кнопке**
Enabled	**Признак доступности кнопки. Кнопка доступна, если значение свойства равно** True, **и недоступна, если значение свойства равно** False
Left	**Расстояние от левой границы кнопки до левой границы формы**
Top	**Расстояние от верхней границы кнопки до верхней границы формы**
Height	**Высота кнопки**
Width	**Ширина кнопки**

Задержите курсор мыши на кнопке (не щелкаете) – и вы увидите ее имя; или посмотрите значения свойства name кнопки в Object Inspector.

ВАЖНО! У всякого объекта (например, Button1), **помимо** Properties (**свойств**), **имеются еще и** Events (**События**). **С объектом могут происходить события, и он должен на них как-то реагировать. Например, с кнопкой самое частое событие -это щелчок на ней.**

Для каждого события можно определить реакцию объекта на него, поэтому существуют процедуры и функции, которые обрабатывают реакцию объекта на событие – они называются обработчиками события.

Задание 1.1.6. **Создадим обработчик события «Щелчок на кнопке».**

Настройте Object Inspector на Button1, затем выберете вкладку Events. Подведите указатель мышки к событию onClick и щелкните двойным щелчком чуть справа от надписи «onClick» на белом фоне, где мигает курсор (рис. 1.16).

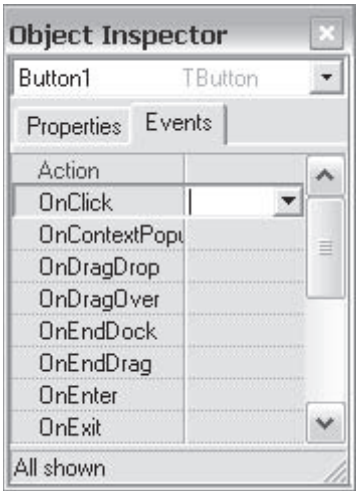


Рис. 1.16. Вкладка Events в Object Inspector

Вы переместитесь в окно редактора кода, и появится заготовка обработчика события onClick:

```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;
```

Вот и все, заготовка обработчика создана. Таким способом **можно создать обработчик события для любого компонента. Для этого нужно настроить** Object Inspector **на компонент (выбрать этот компонент одинарным щелчком мышкой или в** Object Inspector **из выпадающего списка), затем выбрать вкладку** Events **и щелкнуть двойным щелчком чуть справа от надписи (на белом фоне) нужного собы-**

тия, где мигает курсор. Появится заготовка обработчика, которую можно заполнять программным кодом.

Ту же самую заготовку обработчика события onClick можно сделать, если просто двойным щелчком быстро щелкнуть на Button1. Проверьте это. Но это формирует обработчик только для события – щелчок на кнопке. Если нужно создать обработчик только щелчка на кнопке, просто два раза быстро щелкните на Button1. Если же нужно создать обработчик для какого-либо другого события, то это нужно делать именно с помощью Object Inspector.

В данном случае обработчиком события «Щелчок на кнопке» является метод-процедура с именем Button1Click, принимающая параметр Sender типа TObject и принадлежащая к классу TForm1.

Между begin и end заготовки обработчика события onClick напишите следующее:

```
ShowMessage('Моя первая программа на Дельфи');
```

Оператор ShowMessage выдает текст, заключенный между кавычками, на экран в виде текстового сообщения. В этом случае обработчик щелчка на Button1 полностью должен выглядеть так:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
ShowMessage('Моя первая программа на Дельфи');
end.
```

Сохраните в проекте все изменения, щелкнув на пиктограмме Save All. Теперь можно приступать к запуску программы на выполнение.

Откомпилируйте программу на предмет ошибок, нажав «CTRL+F9».

Если в тексте вашей программы есть ошибки, Delphi **отметит их, выделив красной чертой**.

Если ошибок нет, **запустите программу на выполнение, нажав** «F9», **либо выберете последовательно** «Run» → « Run», **либо щелкните по зеленому треугольнику в главном окне**.

Перед вами появится ваш проект, но уже в режиме выполнения, а не режиме разработки. **В режиме выполнения вы уже не можете изменять проект или программировать, а в режиме программирования можете.**

Щелкните по кнопке, и вы получите то, что запрограммировали. При нажатии на Button1 будет выдаваться текстовое сообщение «Моя первая программа на Дельфи».

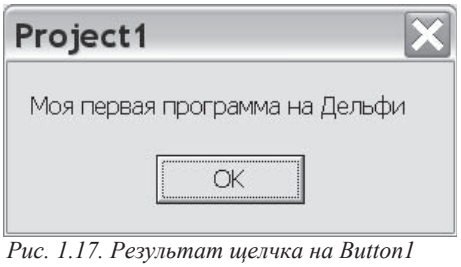


Рис. 1.17. Результат щелчка на Button1

Вот и все, ваше первое приложение в Delphi готово!

Затем, чтобы иметь возможность программировать и изменять проект, вернитесь в **режим программирования (режим разработки)**, для этого просто закройте окно вашего приложения одинарным щелчком мыши или нажмите «Alt»+«F4» как для обыкновенного окна.

После этого в режиме программирования просто закроем проект.

Выберете «File»→»Close All» и проект будет закрыт, затем закройте и саму среду Delphi.

Задание 1.1.7. **Найдите исполняемый** Project1.exe **файл программы**.

Найдите папку вашего проекта «Proekt». Помимо project1.dpr и unit1.pas, вы найдете полноценный исполняемый файл Project1.exe, готовый к выполнению, независимо, есть ли Delphi на данном компьютере или нет. Запустите его. Запустится ваше приложение, которое должно работать нормально. Таким образом, у вас готово стандартное оконное приложение (программа) Windows. После закройте приложение.

Задание 1.1.8. **Откроем уже готовый файл проекта** Project1.dpr **в среде** Delphi.

Здесь же в папке проекта «Proekt» щелкните двойным щелчком по файлу Project1.dpr и автоматически загрузится Delphi с вашим проектом. Вы также можете открыть проект из самого Delphi; сначала закройте имеющийся открытый проект, если он есть, последовательно выбрав «File» → «Close All», а затем последо-

вательно выберите «File» → «Open» или «File» → «Open Project» либо щелкните на эквивалентной пиктограмме (рис. 1.6б). В результате появится диалоговое окно с предложением открыть нужный файл проекта с расширением dpr (рис. 1.18).

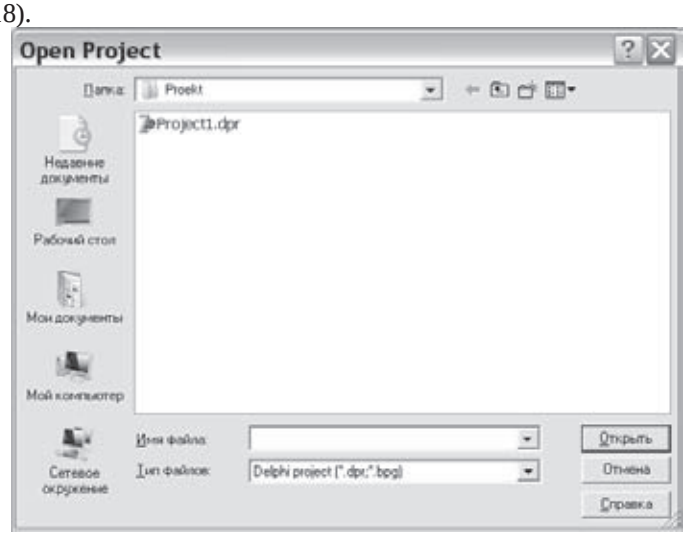


Рис. 1.18. Диалоговое окно предлагающее открыть проект

Выбирите с помощью данного окна, например, файл вашего проекта «Project1.dpr» в папке «Proekt» и щелкните на кнопке «Открыть». Выбранный проект нормально загрузится в среде Delphi в режиме программирования.

1.2. **Создание проекта «Квадратное уравнение» с рассмотрением операторов типа** IntToStr. **Объявление глобальной переменной**. **Работа с таймером и компонентом** Image.

Задание 1.2.1 **Создадим программу «Квадратное уравнение», рассчитывающую корни квадратного уравнения** $ax^2+bx+c=0$.

Создайте новый проект в папке KvadrUrovn, сразу же сохранив данный проект.
Установите значение свойства Caption формы в – Квадратное Уравнение
Перенесите по очереди 5 компонентов Label и 5 компонентов Edit, а также одну кнопку Button1, из палитры Standard, на форму (рис. 1.19).



Рис. 1.19. Окно программы «Квадратное уравнение»

Компонент Label используется как поле для вывода текста, а Edit в основном как поле для ввода текста (но туда также можно выводить текст).

Установите для компонента Label1 значение свойства Caption – “ a ”
Установите для компонента Label2 значение свойства Caption – “ b ”
Установите для компонента Label3 значение свойства Caption – “ c ”
Установите для компонента Label4 значение свойства Caption – “ x1 ”
Установите для компонента Label5 значение свойства Caption – “ x2 ”
Установите для компонента Button1 значение свойства Caption – “ Посчитать ”
Для каждого компонента Edit удалите текст из его свойства Text, чтобы там стало пусто (рис. 1.20).



Рис. 1.20. Окно программы «Квадратное уравнение»

Далее войдите в обработчик события onClick нажатия кнопки Button1 через вкладку Events или просто щелкните двойным щелчком по кнопке Button1.

Добавьте над словом begin ключевое слово var.

Учитывайте, что в тексте приводятся комментарии – это текст, который вы пишите для себя, он не воспринимается компилятором. Комментарии – это либо два слэша «//» – они действуют только на одну строку, либо фигурные скобки {} – они могут распространяться на несколько строк.

Напишите в обработчике следующий текст, в результате чего обработчик будет выглядеть следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b,D,x1,x2:real; c:integer;
begin
a:=StrToFloat(Edit1.Text); //в Edit1 можем вводить или вещественное или целое значение
b:=StrToFloat(Edit2.Text); // в Edit2 можем вводить или вещественное или целое значение
c:=StrToInt(Edit3.Text); // в Edit3 можем вводить только целое значение
D:=b*b-4*a*c;
if D<0 then
   begin
   ShowMessage('Корней        нет');
   halt;// немедленное прекращение программы
   end;

x1:=(-b+sqrt(D))/(2*a);//вычислили первый корень
x2:=(-b-sqrt(D))/(2*a);// вычислили второй корень
edit4.Text:=FloatToStr(x1);//выводим значение x1 на экран в Edit4
edit5.Text:=FloatToStr(x2);// выводим значение x2 на экран в Edit5
end;
```

Запустите программу, например, введите в поле Edit1 значения 9, в поле Edit2 значение 5, в поле Edit3 значения -4. Убедитесь, что она работает правильно и выдает корни x1 и x2.

Проанализируйте текст в комментариях кода. **Вернитесь в режим программирования**.

Проанализируем выражение a:=StrToFloat(Edit1.Text). Здесь компонент Edit1 является полем для ввода текста в его свойство Edit1.text строкового типа, поэтому введенное туда значение будет интерпретироваться программой как текстовый тип, строка, набор символов. Поэтому мы не можем просто написать a:= Edit1.Text, так как это ошибка. В противном случае компилятор начнет «возмущаться», поскольку нельзя присвоить переменной «a» вещественного типа real, текстовое или строковое значение типа string, хотя бы и в виде цифр. Мы должны указать компилятору, как интерпретировать введенное значение, и применяем оператор StrToFloat(‘тут текстовое представление числа’), его можно трактовать как «Строка_В_Вещественное число». Оператор StrToFloat преобразует текстовое представление числа в вещественный тип.

При этом, если текстовый тип преобразуем в вещественный тип, пишем StrToFloat, а если текстовый тип преобразуем в целый тип, пишем StrToInt – «Строка_В_Целое число».

Сравните, например, a:=StrToFloat(Edit1.Text) и c:=StrToInt(Edit3.Text).

Далее мы выводим значение переменной x1 вещественного типа real на экран в Edit4.text и поэтому пишем edit4.Text:=FloatToStr(x1). Здесь уже, наоборот, с помощью оператора FloatToStr преобразуем вещественный тип real в строковый тип: «Вещественное число_В_Строку», ситуация обратная предыдущей.

Задание 1.2.2 **Объявим глобальную переменную** X **типа** integer.

Обратите внимание, что все переменные - a, b, D, x1, x2 и c в обработчике нажатия кнопки procedure TForm1.Button1Click(Sender: TObject), идущие после слова var, – являются локальными по отношению к обработчику и не будут видны из других мест модуля Unit1.pas кроме этой процедуры, в отличие от глобальных переменных.

Чтобы создать глобальную переменную X, **в окне редактора кода найдите ключевое слово** var, **оно расположено над ключевым словом** implemantation, **рядом с ним найдете запись** Form1: TForm1; **объявите там имя** X **и тип глобальной переменной** integer (**рис**. 1.21).

```
var // Это место условимся называть «главный var модуля»
Form1: TForm1; X:integer; // глобальная переменная X видна во всех
                          // процедурах и функциях модуля Unit1.pas
```

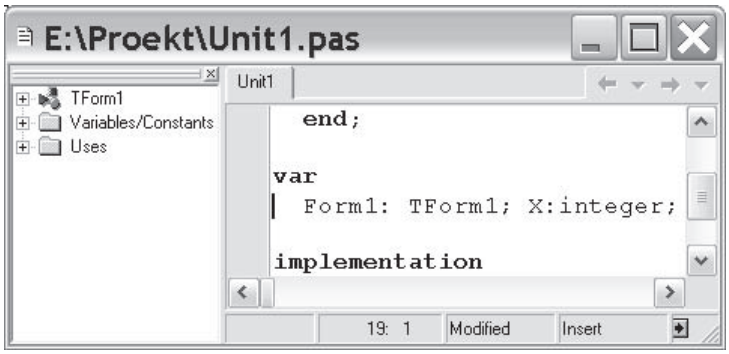


Рис. 1.21. Окно редактора кода. Объявляем глобальные переменные Form1 и X.

Таким образом, объявление переменной X в этом месте модуля делает ее глобальной по отношению ко всему модулю, то есть переменная X будет доступна в любом месте модуля приложения.

Задание 1.2.3 **Используя компонент** Timer1, **запрограммируем так**, **чтобы через** 1 **секунду значение переменной целого типа увеличивалось и отображалась на форме**.

Создайте новый проект в папке с названием «Timer».

Переместите на форму Label1, checkbox1, а также таймер timer1 из вкладки System палитры компонентов (рис. 1.22).

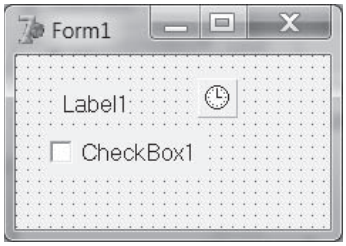


Рис. 1.22. Окно проекта с названием «Timer»

Установите следующие значения свойств компонентов приведенные в таблице:

Компонент	**Свойство** (properties)	**Значение свойства**
Label1	Caption	0
checkbox1	Checked	False
	Caption	Запуск
Timer1	Interval	1000
	Enabled	False

Здесь свойство Checked компонента checkbox1 является – булевским (логическим) значением, отвечающим на вопрос, поставлена (True) галочка в checkbox1 или нет (False).

Свойство Interval компонента timer1 определяет промежуток времени (в миллисекундах, 1000 миллисекунд это 1 секунда), через который будет генерироваться событие onTimer и, следовательно, будет вызываться обработчик данного события. У таймера всего одно событие -onTimer.

Создайте теперь глобальную переменную X типа integer (как ее создавать, сказано выше).

Затем настройте Object Inspector на форму Form1, выберите вкладку Events и создайте обработчик события onCreate.

Рядом с ключевым словом var **имеется запись** Form1:TForm1, **поэтому в памяти компьютера может быть создан объект (переменная-указатель на объект** Form1) **класса** TForm1. **При этом каждый объект имеет определенный специфичный метод** – constructor, **который вызывается единственный раз и формирует объект в оперативной памяти. Можно сказать, что конструктор – это главный метод, который создает объект. При этом часто конструктор используется для инициализации (присвоения начальных значений) полей объекта. В** Delphi **в большинстве случаев имя конструктора называется** «Create», что в переводе означает – «создать».

Поэтому в начале, когда создается объект Form1 – генерируется событие onCreate и в обработчике данного события onCreate мы присвоим начальное нулевое значение переменной X.

Используя объект инспектор перейдите в данный обработчик и напишите в этом обработчике всего одну строчку:

```
X:=0;//Инициализируем X начальным значением 0
```

Полностью данный обработчик будет выглядеть так:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
X:=0; //Инициализируем X начальным значением 0
end;
```

Далее создайте обработчик события onClik (щелчка) на компоненте checkbox1 и напишите там следующее:

```
if CheckBox1.Checked=true then timer1.Enabled:=true
else timer1.Enabled:=false.
```

Если у компонента CheckBox1 поставлена галочка (CheckBox1.Checked=true), то нужно запустить таймер, другими словами, активировать таймер – timer1.Enabled:=true, а иначе – сделать таймер недоступным – timer1.Enabled=false, не выполняющим никаких действий.

Теперь пропишем обработчик таймера. Используя «Object Inspector» → «Events», создадим обработчик события onTimer для таймера Timer1. Полностью обработчик должен выглядеть так:

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
X:=X+1; //увеличиваем значение X на 1 при каждом сообщении от таймера
Label1.Caption:=IntToStr(X); // выводим это значение в Label1
end.
```

Запустите программу, ставьте и снимайте галочку у Checkbox1. **Вернитесь в режим программирования**.

Задание 1.2.4. **Загрузим в графический компонент** Image1 **картинку из графического файла**.

В этом же проекте скиньте на форму графический компонент Image1 из вкладки Additional. Растяните его пошире, сбросьте рядом кнопку SpeedButton1, а также компонент OpenDialog1 из вкладки Dialogs, который представляет собой диалоговое окно выбора фалов.

Для Speed Button1 измените значение свойства Caption на «Загрузить картинку».

В обработчике щелчка на SpeedButton1 напишите:

```
if OpenDialog1.Execute=true then
Image1.Picture.LoadFromFile(OpenDialog1.FileName);
```

Запустите программу, нажмите кнопку, в появившемся диалоговом окне выберите какой-нибудь *.bmp файл и нажмите OK. Картинка должна нормально загрузиться в компонент Image1.



Рис. 1.23. Картинка помещенная в Image1

В данном случае, если диалоговое окно нормально загрузилось в память (if OpenDialog1.Execute=true), то в свойство Picture графического контейнера Image1 загружается рисунок из файла методом LoadFromFile. Имя файла содержится в свойстве FileName объекта OpenDialog1, мы этот файл выбираем мышкой. Пока вы не можете открыть *.jpg файл, как это сделать, будет рассказано ниже. **Вернитесь в режим программирования**.

Чтобы перемещаться между файлами модуля и главным файлом проекта, следует пользоваться сочетанием клавиш «Ctrl+F12».

ПРАКТИЧЕСКАЯ РАБОТА №2

2.1. **Работа с буфером обмена, рисование мышью, контекстное меню, компонент** Memo1 **и** Image1

Часто в приложениях нужно открыть какой – нибудь графический файл, изменить его содержимое, затем поместить это содержимое в буфер обмена, чтобы оно могло быть доступно другим приложениям и т.д. Также полезно иметь возможность рисования мышкой в окне созданного приложения. Часто при различных расчетах нужно в оперативном порядке вывести куда-нибудь рассчитанные программой результаты. Очень удобно и просто сделать это, используя стандартный компонент Memo1, очень похожий на стандартное приложение «Блокнот» операционной системы Windows. Описанные вещи кратко рассматриваются в данной практической работе.

Задание 2.1.1 **Научимся работать с буфером обмена помещая и извлекая оттуда графические изображения**.

Поместите на форму следующие компоненты: Image1 и Image2 (палитра Additional), OpenDialog1 (палитра Dialogs), а также 3 кнопки класса TButton, с заголовками на них так, как указано на рис. 2.1.



Рис. 2.1 Окно приложения «Буфер обмена»

Назначение кнопок данного приложения будет следующим:

Обработчик щелчка Button1 – Откроет картинку в Image1.
Обработчик щелчка Button2 – Скопирует картинку в буфер обмена – объект Clipboard.
Обработчик щелчка Button3 – Загрузит картинку в Image2.

Далее напишите имя подключаемого модуля, который описывает класс объекта Clipboard, т.е. буфера. Для этого в редакторе кода в самом верху после описания стандартных подключаемых модулей, идущих после ключевого слова uses, подпишите через запятую ClipBrd [2] (для буфера) и Jpeg для возможности открытия файлов *.jpg (рис. 2.2).

Также для обоих компонентов Image с помощью Object Inspector установите их свойство Stretch в значение True, чтобы рисунок подстраивался под их размер.

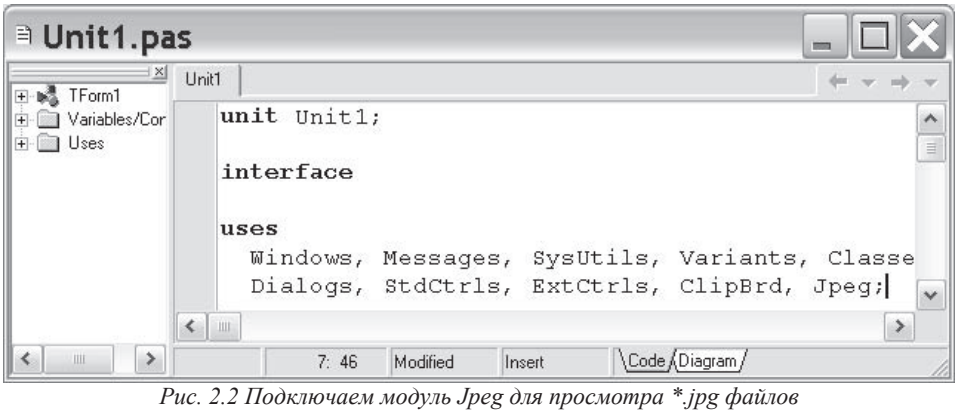


*Рис. 2.2 Подключаем модуль Jpeg для просмотра *.jpg файлов*