

Delphi XE2



- Визуальная библиотека компонентов
- Сетевые приложения на базе именованных каналов и механизма сокетов
- Клиент-серверные приложения на основе технологии COM
- Управление процессами, потоками и службами
- Многоуровневые приложения DataSnap
- Технология живого связывания LiveBindings
- Введение в кроссплатформенную разработку на основе библиотеки FireMonkey



Материалы
на www.bhv.ru

Наиболее
полное
руководство

В ПОДЛИННИКЕ®

УДК 681.3.068+800.92Delphi XE2

ББК 32.973.26-018.1

O-74

Осипов Д. Л.

O-74 Delphi XE2. — СПб.: БХВ-Петербург, 2012. — 912 с.: ил. —
(В подлиннике)

ISBN 978-5-9775-0825-4

Книга посвящена одному из самых совершенных языков программирования Delphi XE2. В ней излагаются основы программирования на языке Delphi XE2, подробно рассматривается визуальная библиотека компонентов (VCL), описывается порядок разработки программного обеспечения для 32- и 64-разрядных версий Windows с использованием функций Win API, предоставляется обзор новейшей кроссплатформенной библиотеки FireMonkey, позволяющей создавать программное обеспечение не только для ОС Microsoft Windows, но и для Mac OS X. Примеры проектов из книги размещены на сайте издательства.

Для программистов и студентов

УДК 681.3.068+800.92Delphi XE2

ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Марины Дамбиевой</i>
Зав. производством	<i>Николай Тверских</i>

Подписано в печать 05.03.12.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 73,53.

Тираж 1200 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0825-4

© Осипов Д. Л., 2012

© Оформление, издательство "БХВ-Петербург", 2012

Оглавление

Введение.....	1
ЧАСТЬ I. ОСНОВЫ ЯЗЫКА DELPHI	3
Глава 1. Знакомство	5
Структура консольного приложения.....	6
Комментарии.....	7
Перевод листинга программы в машинные коды	8
Первая программа.....	10
Глава 2. Типы данных Delphi	11
Переменные.....	11
Константы.....	13
Строки-ресурсы.....	13
Правила объявления идентификаторов.....	14
Типы данных	14
Простые типы.....	16
Целые числа.....	16
Символьный тип.....	16
Логический тип	17
Перечисления	18
Диапазоны	19
Обслуживание данных порядкового типа	19
Действительные типы	20
Строковый тип	21
Указатели.....	22
Вариантный тип	24
Глава 3. Структурные типы	25
Множества.....	26
Записи	27
Вариантные поля.....	29
Усовершенствованная запись	30

Массивы.....	31
Объявление массива.....	32
Обращение к ячейкам массива.....	32
Динамический массив.....	33
Вариантные массивы	34
Глава 4. Операторы и выражения.....	36
Оператор присваивания.....	36
Арифметические операторы	36
Оператор конкатенации строк	37
Логические операторы	38
Операторы поразрядного сдвига	39
Операторы отношения.....	40
Операторы множеств.....	40
Составной оператор <i>begin..end</i>	41
Условный оператор <i>if..then..else</i>	42
Оператор-селектор <i>case</i>	43
Оператор перехода <i>goto</i>	45
Оператор <i>with..do</i>	45
Организация циклов	46
Цикл с параметром <i>for..do</i>	46
Инструкция <i>in</i> в цикле <i>for..do</i>	48
Цикл с предусловием <i>while..do</i>	48
Цикл с постусловием <i>repeat..until</i>	49
Вложенные циклы.....	49
Операторы <i>break</i> и <i>continue</i>	50
Глава 5. Процедуры и функции	52
Процедуры.....	53
Функции.....	55
Особенности передачи параметров	58
Директивы для процедур и функций.....	62
Перегрузка функций: директива <i>overload</i>	62
Опережающее объявление: директива <i>forward</i>	63
Внешнее объявление: директива <i>external</i>	65
Встроенная функция: директива <i>inline</i>	65
Рекурсивная функция	65
Процедурный тип данных	66
Анонимные функции	66
Глава 6. Файлы и каталоги.....	68
Типизированные файлы	68
Пример работы с типизированным файлом.....	72
Особенности удаления записей из больших файлов.....	76
Текстовые файлы	77
Двоичные файлы	79
Управление дисками, каталогами и файлами.....	80
Работа с дисками компьютера	81
Организация поиска каталогов и файлов.....	83
Проверка существования файла и каталога	85

Расположение системных каталогов	85
Создание, удаление, копирование и перемещение	86
Размер файла	87
Дата и время создания файла и каталога	88
Атрибуты файла и каталога	89
Глава 7. Концепция ООП.....	90
Основные понятия ООП	90
Абстрагирование	91
Инкапсуляция	92
Модульность	92
Наследование	93
Класс Delphi	93
Проектирование класса	94
Управление жизненным циклом объекта	96
Опережающее объявление класса	97
Ограничение видимости членов класса	99
Свойства объекта	100
Особенности объявления методов	101
Поля класса и методы класса	101
Иерархия наследования	102
Полиморфизм	103
Операторы класса	105
Аннотация класса	106
Создание и подключение атрибутов с аннотацией	106
Извлечение аннотации	107
Глава 8. Шаблоны.....	109
Обобщенный тип данных в полях записей	109
Обобщения в процедурах и функциях	111
Обобщенные типы данных в шаблонах классов	112
Наследование шаблона класса	113
Перегрузка методов с параметром обобщенного типа	114
Шаблон массива, класс <i>TArray</i>	115
ЧАСТЬ II. ПЛАТФОРМА VCL.....	117
Глава 9. Опорные классы VCL	119
Класс <i>TObject</i>	119
Управление жизненным циклом объекта	121
Информирование о классе	123
Класс <i>TPersistent</i>	125
Основа компонента, класс <i>TComponent</i>	126
Владение компонентом	127
Глава 10. Массивы указателей, наборы строк и коллекции.....	129
Массив указателей, класс <i>TList</i>	130
Контейнер объектов, класс <i>TObjectList</i>	133
Контейнер компонентов, класс <i>TComponentList</i>	133

Наборы строк, класс <i>TStrings</i>	134
Редактирование списка и управление данными	135
Загрузка и сохранение строк	137
Объединение строк	138
Поиск строки и объекта	139
Обслуживание данных "параметр — значение"	139
Оптимизация производительности	139
Особенности класса <i>TStringList</i>	140
Коллекции	141
Элемент коллекции, класс <i>TCollectionItem</i>	142
Создание и уничтожение коллекции	143
Доступ к элементу коллекции	143
Манипуляции элементами коллекции	143
Поиск элемента коллекции	144
Сравнение коллекций	144
Глава 11. Классы потоков данных	145
Прототип потоков данных, класс <i>TStream</i>	145
Потоки с дескриптором, класс <i>THandleStream</i>	147
Файловый поток данных, класс <i>TFileStream</i>	147
Пример работы с файловым потоком данных	149
Потоки данных в памяти	151
Поток данных в памяти <i>TMemoryStream</i>	152
Поток байтов <i>TBytesStream</i>	152
Поток строк <i>TStringStream</i>	153
Поток с возможностью сжатия данных	154
Сжатие данных <i>TZCompressionStream</i>	154
Восстановление данных <i>TZDecompressionStream</i>	155
Глава 12. Визуальные элементы управления и класс <i>TControl</i>	157
Принадлежность к родительскому контейнеру	157
Размещение и размеры элемента управления	158
События, связанные с изменением размеров	160
Пересчет клиентских и экранных координат	160
Выравнивание элемента управления в контейнере	161
Видимость и активность элемента управления	162
Внешний вид	163
Вывод текста	163
Оперативная подсказка	163
Контекстное меню	164
Командный объект	164
Поддержка естественного ввода	165
Обработка событий мыши	165
Щелчки кнопками мыши	165
Перехват щелчков мыши	169
Перемещение указателя мыши	170
Вращение колесика мыши	172
Операция перетаскивания drag and drop	173
Пример реализации операции drag and drop	175

Глава 13. Оконные элементы управления и класс <i>TWinControl</i>	178
Дескриптор окна	178
Управление подчиненными элементами.....	180
Выравнивание подчиненных элементов	181
Фокус ввода	182
Обработка событий клавиатуры	183
Операция буксировки <i>drag and dock</i>	185
Буксировка программным способом	189
Глава 14. Приложение VCL	191
Приложение VCL, класс <i>TApplication</i>	193
Оконная процедура	193
Общие возможности <i>Application</i>	195
Доступ к основным объектам приложения	197
Обработка поступающих сообщений	197
Управление процессом создания приложения	199
Завершение работы приложения	201
Сворачивание и восстановление размеров окна приложения	202
Диалоговое окно приложения	202
Осуществление оперативной подсказки	203
Подключение к справочной системе	204
События приложения, класс <i>TApplicationEvents</i>	205
Значок в области уведомлений	206
Пример работы с компонентом <i>TTrayIcon</i>	208
Стили оформления приложения	209
Менеджер стилей <i>TStyleManager</i>	211
Утилита создания стилей оформления	213
Глава 15. Форма, фрейм и модуль данных	215
Форма проекта VCL, класс <i>TForm</i>	215
Описание формы в <i>dfm</i> -файле	216
Стиль, поведение и оформление формы	218
Состояние формы.....	219
Создание, отображение и уничтожение форм	220
Вывод формы в модальном режиме	221
Закрытие формы.....	222
Уничтожение формы	223
Подключение меню.....	223
Фокус ввода	223
Полосы прокрутки	224
Особенности графического вывода формы	225
Выбор монитора для вывода формы	225
Масштабирование шрифта.....	226
Эффект прозрачности	226
Дескрипторы окна.....	228
Прикрепление формы к границам экрана	228
Обработка событий формы	228
Жизненный цикл формы	229
Нажатие быстрых клавиш	231

Пользовательские интерфейсы SDI и MDI	232
Особенности проекта MDI	233
Фрейм <i>TFrame</i>	237
Создание простого фрейма	237
Диалоги-помощники на основе фреймов	239
Модуль данных <i>TDataModule</i>	242
Глава 16. Исключительные ситуации	244
Защищенные от ошибок секции	244
Конструкция <i>try..except</i>	245
Конструкция <i>try..finally</i>	245
Вложенные конструкции <i>try</i>	246
Объектная модель исключительных ситуаций	247
Базовый класс <i>Exception</i> исключительной ситуации	249
Тихая исключительная ситуация <i>EAbort</i>	251
Исключительная ситуация отладки <i>EAssertionFailed</i>	252
Определение новых классов ИС	253
Расширенные возможности <i>try..except</i>	254
Централизованная обработка ошибок в приложении	255
Настройка поведения Delphi при обработке исключительных ситуаций	256
Глава 17. Компоненты отображения и редактирования текста	257
Компоненты отображения текста	257
Метка <i>TLabel</i>	257
Статический текст <i>TStaticText</i>	260
Метка-ссылка <i>TLinkLabel</i>	260
Компоненты редактирования текста	261
Основа текстовых редакторов, класс <i>TCustomEdit</i>	262
Ограничения на ввод	263
Выделение части текста	264
Взаимодействие с буфером обмена	265
Отмена изменений	265
Строка ввода <i>TEdit</i>	266
Строка ввода с кнопками <i>TButtonEdit</i>	266
Строка ввода с меткой <i>TLabelEdit</i>	266
Строка ввода с маской <i>TMaskEdit</i>	267
Многострочный текстовый редактор <i>TMemo</i>	268
Редактор расширенного текстового формата <i>TRichEdit</i>	269
Форматирование абзаца — класс <i>TParaAttributes</i>	270
Атрибуты текста — класс <i>TTextAttributes</i>	271
Особенности обработки событий	271
Глава 18. Кнопки и компоненты выбора значений	274
Кнопка <i>TButton</i>	274
Кнопка с рисунком <i>TBitBtn</i>	277
Кнопка-флажок <i>TCheckBox</i>	278
Кнопки изменения значения <i>TUpDown</i>	279
Кнопка выбора <i>TRadioButton</i>	280
Группа переключателей <i>TRadioGroup</i>	280

Группа кнопок <i>TButtonGroup</i>	281
Категории кнопок <i>TCategoryButtons</i>	284
Глава 19. Меню приложения	290
Опорный класс меню <i>TMenu</i>	291
Главное меню <i>TMainMenu</i>	292
Контекстное меню <i>TPopupMenu</i>	293
Элемент меню <i>TMenuItem</i>	294
Элемент меню в виде флажка	295
Элементы меню в виде группы выбора	296
Родительские и дочерние элементы меню	297
Присвоение элементам меню значков	298
Динамическое создание элементов меню	298
Удаление элементов меню	300
Элементы-разделители	300
Особенности прорисовки пункта меню	301
Глава 20. Управление приложением с помощью команд	303
Команда <i>TAction</i>	304
Связь с элементом управления	307
Выполнение команды	307
Установка в актуальное состояние	308
Связь команды с контейнером	308
Менеджеры команд	308
Общие черты менеджеров команд	309
Список команд <i>TActionList</i>	310
Менеджер команд <i>TActionManager</i>	310
Командные панели	311
Класс <i>TActionClientItem</i>	312
Опорный класс командных панелей <i>TCustomActionBar</i>	314
Панель главного меню <i>TActionMainMenuBar</i>	315
Инструментальная панель <i>TActionToolBar</i>	316
Контекстное командное меню <i>TPopupMenuActionBar</i>	316
Настройка интерфейса во время выполнения приложения, диалог <i>TCustomizeDlg</i>	317
Редактор "горячих" клавиш <i>THotKey</i>	319
Глава 21. Списки	321
Опорный класс списков <i>TCustomListControl</i>	323
Общие черты списков, список <i>TListBox</i>	324
Замедление перебора элементов списка	327
Особенности обработки событий	327
Список с флажками выбора <i>TCheckListBox</i>	330
Список выбора цвета <i>TColorListBox</i>	331
Комбинированные списки, <i>TComboBox</i>	333
Улучшенный комбинированный список <i>TComboBoxEx</i>	335
Список просмотра <i>TListView</i>	337
Стиль представления данных	338
Особенности работы списка со стилем <i>vsReport</i>	339
Колонка <i>TListColumn</i>	340

Коллекция элементов списка <i>TListItems</i>	340
Элемент списка <i>TListItem</i>	342
Редактирование заголовка элемента	343
Выбор элементов списка	343
Упорядочивание элементов	345
Поиск элементов	345
Группировка элементов	346
Операции перерисовки	346
Пример работы с <i>TListView</i>	348
Глава 22. Сетки	353
Общие черты сеток, сетка <i>TDrawGrid</i>	354
Адресация ячейки	356
Обработка событий	356
Расширенные возможности по оформлению сетки	358
Сетка строк <i>TStringGrid</i>	359
Редактор списка значений <i>TValueListEditor</i>	363
Глава 23. Иерархические данные и компонент <i>TTreeView</i>	367
Сохранение и загрузка дерева	369
Выбор узла в дереве	369
Одновременный выбор нескольких узлов	370
Узел дерева <i>TTreeNode</i>	371
Положение узла в дереве	372
Родительские узлы	372
Дочерние узлы	373
Методы перехода между узлами дерева	373
Перемещение узла	374
Удаление узла	375
Значок узла	375
Свертывание и разворачивание узла	376
Хранилище узлов класс <i>TTreeNodeList</i>	377
Добавление узлов	378
Сортировка узлов	381
Удаление узлов из коллекции	382
Редактирование текста узла	383
Оформление дерева	383
Глава 24. Панели-контейнеры	387
Простые панели	387
Простая панель <i>TPanel</i>	388
Панель <i>TFlowPanel</i>	390
Панель-сетка <i>TGridPanel</i>	390
Область группировки <i>TGroupBox</i>	392
Контейнеры с возможностью скроллинга	393
Область с полосами прокрутки <i>TScrollBar</i>	393
Страница с кнопками прокрутки <i>TPageScroller</i>	394
Разделитель панелей, компонент <i>TSplitter</i>	395

Глава 25. Инструментальные планки	397
Инструментальная планка <i>TToolBar</i>	398
Кнопка <i>TToolButton</i>	398
Управление кнопками.....	402
Пользовательские настройки	403
Оформление.....	403
Планка <i>TCoolBar</i>	404
Дочерняя полоса <i>TCoolBar</i>	405
Планка управления <i>TControlBar</i>	406
Панель состояния <i>TStatusBar</i>	408
Глава 26. Наборы закладок и блокноты	412
Набор закладок, <i>TTabControl</i>	413
Закладки <i>TTabSet</i> и <i>TDockTabSet</i>	416
Блокнот <i>TPageControl</i>	416
Страница блокнота <i>TTabSheet</i>	418
Глава 27. Работа с датой и временем	420
Отсчет времени, таймер <i>TTimer</i>	421
Компоненты-календари, базовый класс <i>TCommonCalendar</i>	422
Календарь <i>TMonthCalendar</i>	424
Выбор даты/времени, компонент <i>TDateTimePicker</i>	425
Глава 28. Диалоговые окна.....	427
Окна вывода сообщений.....	427
Окна выбора действия	428
Создание многоразового окна выбора действия	431
Окна ввода данных	431
Окна выбора файлов и папок	432
Компоненты-диалоги.....	433
Диалоги открытия и сохранения файлов	435
Универсальные диалоги <i>TOpenDialog</i> и <i>TSaveDialog</i>	435
Особенности графических диалогов <i>TOpenPictureDialog</i> и <i>TSavePictureDialog</i>	441
Особенности текстовых диалогов <i>TOpenTextFileDialog</i> и <i>TSaveTextFileDialog</i>	441
Диалоги поиска и замены текста	441
Выбор шрифта <i>TFontDialog</i>	445
Выбор цвета <i>TColorDialog</i>	446
Параметры страницы <i>TPageSetupDialog</i>	447
Настройка печати <i>TPrinterSetupDialog</i>	449
Отправка задания на печать <i>TPrintDialog</i>	450
Диалог управления задачей <i>TTaskDialog</i>	451
Глава 29. Технология естественного ввода	454
Описание жеста.....	454
Реакция элементов управления на жест	456
Пример обработки стандартных жестов	458
Компоненты поддержки естественного ввода.....	459
Менеджер жестов <i>TGestureManager</i>	459
Доступ к жестам и их сохранение	460

Просмотр жестов, <i>TGestureListView</i> и <i>TGesturePreview</i>	462
Область ввода жеста <i>TGestureRecorder</i>	462
Виртуальная клавиатура <i>TTouchKeyboard</i>	463
Глава 30. Управление графическим выводом	465
Получение сведений об устройствах видеовывода	465
Изменение настроек дисплея	468
Исследование текущего состояния устройства	470
Взаимодействие с экраном, класс <i>TScreen</i>	471
Информация о рабочем столе	472
Управление видом указателя мыши	472
Информация о шрифтах системы	472
Информация о формах проекта	473
Информация об устройствах видеовывода	474
Реакция на события	474
Взаимодействие с дисплеем, класс <i>TMonitor</i>	474
Глава 31. Холст <i>TCanvas</i>	476
Представление цвета	477
Кисть <i>TBrush</i>	479
Перо <i>TPen</i>	481
Шрифт <i>TFont</i>	484
Холст <i>TCanvas</i> в VCL	487
Закраска области	487
Градиентная заливка	488
Графические примитивы	489
Линии	490
Простейшие геометрические фигуры	491
Дуги	492
Сплайн Безье	492
Копирование части холста	494
Глава 32. Растровая и векторная графика	496
Абстрактный базовый класс <i>TGraphic</i>	497
Значок <i>TIcon</i>	499
Формат BMP, класс <i>TBitmap</i>	500
Формат JPEG, класс <i>TJPEGImage</i>	504
Формат GIF, класс <i>TGifImage</i>	506
Управление фреймами рисунка GIF	508
Оптимизация рисунка GIF	509
Обработка событий	511
Формат PNG, класс <i>TPngImage</i>	511
Векторная графика, метафайл <i>TMetaFile</i>	514
Холст метафайла <i>TMetafileCanvas</i>	515
Универсальный контейнер <i>TPicture</i>	516
Универсальный контейнер <i>TWICImage</i>	517
Коллекция изображений <i>TImageList</i>	518
Загрузка образов в контейнер	519

Особенности отображения значков	521
Прозрачность.....	522
Экспорт значков из контейнера	523
Глава 33. Сложные графические задачи	524
Растровые операции	524
Управление прозрачностью	527
Системы координат и режимы отображения.....	530
Перенос начала координат	531
Управление страничными координатами	532
Мировые координаты и аффинные преобразования.....	534
Глава 34. Управление печатью	539
Описание принтера в Delphi, класс <i>TPrinter</i>	540
Выбор принтера	540
Управление страницей документа	541
Формирование и отправка задания на печать.....	541
Отмена задания	542
Печать многострочного текста	543
Особенности печати изображений	544
Пример печати изображений	544
Окно предварительного просмотра	545
Отправка задания на печать	550
ЧАСТЬ III. VCL И WINDOWS API.....	551
Глава 35. Реестр Windows	553
Класс <i>TRegistryIniFile</i>	554
Чтение из реестра.....	555
Запись в реестр.....	556
Удаление подраздела	557
Пример	557
Класс <i>TRegistry</i>	559
Создание и уничтожение экземпляра реестра	559
Работа с удаленным реестром.....	559
Доступ к разделам реестра	559
Чтение и запись значений в параметры	561
Получение информации о разделе.....	562
Получение сведений о параметре	563
Экспорт и импорт разделов реестра	563
Глава 36. Управление процессами	565
Создание процесса	566
Доступ к процессу.....	568
Приоритет процесса.....	569
Время выполнения процесса	570
Завершение процесса	571
Сбор информации о процессах Windows.....	572
Получение сведений о версии ОС	575

Глава 37. Многопоточные приложения	576
Поток <i>TThread</i>	576
Метод ожидания	580
Управление приоритетом потока	581
Время выполнения потока	582
Синхронный и асинхронный вызовы внешнего метода	582
Пример многопоточного приложения	582
Синхронизация потоков	586
Синхронизация событием <i>TEvent</i>	587
Критическая секция <i>TCriticalSection</i>	590
Мьютекс <i>TMutex</i>	590
Семафор <i>TSemaphore</i>	592
Глава 38. Взаимодействие процессов	594
Обмен данными через буфер обмена	594
Регистрация пользовательского формата буфера обмена	597
Обмен сообщениями	601
Поиск окна	602
Регистрация пользовательских сообщений	604
Пример обмена сообщениями между процессами	605
Файловое отображение	608
Глава 39. Сетевое взаимодействие	613
Почтовые слоты	613
Определение имени почтового слота	614
Управление почтовым слотом	615
Получение и отправка корреспонденции	615
Пример почтового приложения	616
Именованные каналы	619
Определение имени именованного канала	619
Создание именованного канала	620
Управление соединением с клиентом	622
Состояние канала	623
Подключение к каналу клиентского приложения	624
Разработка класса сервера именованного канала	625
Разработка класса клиента именованного канала	628
Сокеты	629
Классы сокетов в VCL	629
Общие черты сокетов, опорный класс <i>TIPSocket</i>	630
Отправка и получение данных	633
Сервер, компонент <i>TTCPServer</i>	634
Клиенты, компоненты <i>TTCPClient</i> и <i>TUDPSocket</i>	636
Пример приложения	636
Сокет-клиент	637
Сокет-сервер	638
Глава 40. Сервисы Windows	641
Менеджер управления сервисами	642
Управление сервисом	644
Состояние службы	646

Конфигурирование службы.....	647
Удаление службы.....	647
Сервис в VCL, класс <i>TService</i>	647
Идентификация	647
Тип сервиса.....	648
Определение прав на управление сервисом.....	648
Загрузка и запуск службы.....	648
Статус службы.....	649
Сбой при старте сервиса.....	650
Остановка и возобновление службы	650
Инсталляция и деинсталляция сервиса	651
Выполнение службы, поток <i>TServiceThread</i>	652
Ведение протокола службы.....	653
Приложение-сервис <i>TServiceApplication</i>	654
Пример	654
Регистрация сервиса в ручном режиме	657
Апплеты Панели управления	658
Апплет Панели управления, класс <i>TAppletModule</i>	659
Приложение Панели управления <i>TAppletApplication</i>	660
Пример апплета управления сервисом Windows.....	661
Приложение управления сервисом.....	661
Апплет Панели управления.....	662
Глава 41. Динамически подключаемые библиотеки	665
Создание проекта DLL	666
Объявление и экспорт функций в DLL	667
Соглашение о вызовах.....	667
Пример экспорта функций	668
Пример хранения форм в библиотеке	669
Вызов библиотеки из приложения	670
Неявное подключение DLL.....	671
Явное подключение DLL.....	673
Глава 42. Многокомпонентная модель COM.....	675
COM-объект	676
Понятие интерфейса	676
Базовый интерфейс <i>IUnknown</i>	678
Реализация интерфейса.....	679
Порядок вызова сервера COM.....	680
Интерфейс <i>IClassFactory</i> и библиотека COM.....	681
Реализация фабрики класса, класс <i>TComObjectFactory</i>	683
Реализация COM-объекта в Delphi.....	685
Класс <i>TComObject</i>	686
Класс <i>TTypedComObject</i>	686
Класс <i>TComServer</i>	687
Пример COM-проекта	688
COM-сервер.....	688
Помощник настройки COM-объекта.....	689
Шаблон кода с описанием класса.....	690
Библиотека типов.....	691

Главная форма сервера	696
Регистрация сервера	697
COM-клиент	698
Импорт библиотеки типов.....	698
Обращение к COM-объекту	700
Глава 43. Автоматизация	701
Интерфейс <i>IDispatch</i>	702
Диспінтерфейсы и дуальные интерфейсы	703
Контроллер автоматизации без применения библиотеки типов	703
Контроллер автоматизации с поддержкой библиотеки типов	705
Сервер автоматизации, базовый класс <i>TAutoObject</i>	708
Регистрация сервера автоматизации в таблице ROT	710
События автоматизации	711
Фабрика класса объекта автоматизации	714
Пример проекта автоматизации с поддержкой событий	714
Сервер автоматизации	715
Клиент автоматизации	722
Глава 44. Интерфейс <i>IShellFolder</i>	728
Идентификация объекта Shell	729
Диалоговое окно получения PIDL	729
Получение пути к системным папкам	731
Интерфейс <i>IShellFolder</i>	732
Получение PIDL из файлового пути.....	733
Получение интерфейса дочерней папки.....	733
Получение названия объекта по PIDL.....	734
Изменение названия объекта.....	735
Сбор дочерних объектов папки, интерфейс <i>IEnumIDLst</i>	735
Атрибуты объекта	737
Сравнение объектов папки	738
Глава 45. DataSnap.....	739
Архитектура проекта DataSnap	740
Компоненты сервера.....	742
Сервер <i>TDSServer</i>	742
Обработка событий	742
Класс сервера <i>TDSServerClass</i>	745
Транспортные компоненты <i>TDSTCPSTransport</i> и <i>TDSHTTPService</i>	746
Менеджер аутентификации <i>TDSAuthenticationManager</i>	748
Компоненты клиента	749
Соединение <i>TSQLConnection</i>	750
Проект DataSnap с использованием мастера	752
Подготовка клиентского приложения	756
Создание нового метода на сервере DataSnap	758
Доступ к новому методу из клиентского приложения	759
Проект DataSnap на основе пользовательского класса.....	760
Сервер	760
Клиент	762
Механизм обратного вызова	764

Глава 46. LiveBindings	767
Вводный пример LiveBindings.....	767
Класс <i>TBindExpression</i>	770
Выражение LiveBindings.....	772
Программная связь, класс <i>TBindings</i>	773
ЧАСТЬ IV. FIREMONKEY	777
Глава 47. Платформа FireMonkey	779
Опорный класс <i>TFmxObject</i>	779
Создание и уничтожение экземпляра класса.....	781
Сохранение объекта в памяти.....	781
Управление дочерними объектами.....	781
Сопоставление дополнительных данных.....	782
Элемент управления FMX — класс <i>TControl</i>	782
Размещение и выравнивание элемента управления.....	783
Выравнивание объекта.....	784
Масштабирование и вращение объекта.....	785
Видимость и прозрачность элемента управления.....	786
Обработка событий.....	786
Простейшие события — щелчок.....	786
Клавиатурные события.....	787
События мыши.....	787
События получения и утраты фокуса ввода.....	788
Событие изменения размера.....	788
События перетаскивания drag and drop.....	788
Особенности прорисовки элемента управления.....	790
Глава 48. Приложение FireMonkey	791
Выбор целевой платформы для проекта.....	791
Приложение <i>FMX.Forms.TApplication</i>	793
Общие черты форм HD и 3D.....	794
Форма HD <i>FMX.Forms.TForm</i>	795
Стили оформления формы, компонент <i>TStyleBook</i>	795
Трехмерная форма <i>FMX.Forms.TForm3D</i>	796
Пример 3D-проекта.....	798
Глава 49. Обзор компонентов для проектов HD	801
Панель-выноска <i>TCalloutPanel</i>	801
Разворачивающаяся панель <i>TExpander</i>	802
Компонент <i>TArcDial</i>	803
Компонент <i>TNumberBox</i>	803
Компонент <i>TComboTrackBar</i>	803
Компонент <i>TPopupBox</i>	804
Сетки <i>TGrid</i> и <i>TStringGrid</i>	805
Глава 50. Анимация	808
Анимация.....	808
Простой пример анимации.....	809

Общие черты компонентов-аниматоров, класс <i>TAnimation</i>	810
Индивидуальные особенности компонентов-аниматоров	812
Цветовая анимация, компонент <i>TColorAnimation</i>	813
Градиентная анимация, компонент <i>TGradientAnimation</i>	813
Анимированная картинка, компонент <i>TBitmapAnimation</i>	813
Анимированный ряд, компонент <i>TBitmapListAnimation</i>	813
Анимация числовых свойств, компонент <i>TFloatAnimation</i>	814
Анимация прямоугольной области, компонент <i>TRectAnimation</i>	814
Анимация траектории, компонент <i>TPathAnimation</i>	814
Управление графической производительностью	815
ПРИЛОЖЕНИЯ	817
Приложение 1. Математика, статистика и тригонометрия	819
Приложение 2. Работа со строками и символами.....	825
Системные настройки форматирования и класс <i>TFormatSettings</i>	829
Приложение 3. Работа с датой и временем	831
Представление даты и времени в текстовом формате	840
Приложение 4. Работа с памятью	843
Приложение 5. Управление ходом выполнения программы	845
Приложение 6. Работа с именами папок и файлов.....	846
Приложение 7. Модуль <i>IOUtils</i>	848
Приложение 8. Константы <i>CSIDL</i>	855
Приложение 9. Холст <i>FMX.Types.TCanvas</i>	859
Управление холстом	860
Кисть <i>FMX.Types.TBrush</i>	861
Внешний вид линий	862
Шрифт <i>FMX.Types.TFont</i>	863
Заливка замкнутых областей.....	863
Вывод простейших фигур	865
Вывод текста	865
Отображение рисунков.....	866
Отсечение	867
Сохранение и восстановление состояния холста	867
Приложение 10. Описание электронного архива	868
Предметный указатель	869



ГЛАВА 1

Знакомство

Часть I книги призвана научить читателя основам программирования на языке Delphi — потомке знаменитого языка Pascal, созданного известным швейцарским ученым, профессором Цюрихской высшей технической школы, обладателем премии Тьюринга (высшей награды для специалистов по информационным технологиям) Никлаусом Виртом (Niklaus Wirth).

Появившийся на свет в начале 1970-х годов язык Pascal предназначался для обучения студентов основам структурного программирования. За более чем четыре десятилетия своего развития Pascal не просто доказал свою жизнеспособность, он стал одним из ведущих языков разработки программного обеспечения. Во многом своему первому успеху язык Pascal обязан компании Borland, в стенах которой в 1992 г. появилась среда разработки Borland Pascal with Objects 7.0, и немного позднее, в 1995 г., — знаменитая Borland Delphi 1.0 для разработки программного обеспечения (ПО) для 16-разрядной ОС Microsoft Windows 3.x.

ЗАМЕЧАНИЕ

Имя Pascal было выбрано не случайно — язык был назван в честь французского математика Блеза Паскаля.

Успех первой версии Delphi и ее *библиотеки визуальных компонентов* (Visual Components Library, VCL) был столь оглушителен, что несколько лет спустя, опираясь на библиотеки Delphi, компания Borland создала еще один шедевр — среду C++Builder (на этот раз с базовым языком Си). Так, уже почти 20 лет, Delphi и C++Builder идут вместе. Эти два языка функционируют в одной среде разработки, обладают практически идентичными возможностями и отличаются лишь особенностями построения синтаксических конструкций, поэтому программист Delphi легко адаптируется к программам, написанным на C++Builder, и, наоборот, разработчик, имеющий опыт работы в среде C++Builder, поймет программу на Delphi.

Сегодня права на язык Delphi принадлежат компании Embarcadero Technologies. Новые владельцы Delphi не просто воспользовались успешными наработками программистов Borland, но и внесли в язык и среду разработки многочисленные улучшения и усовершенствования. В частности, современная Delphi позволяет разрабатывать 32- и 64-разрядные приложения для Windows на базе популярной библиотеки VCL и способна создавать кросс-платформенные приложения (для Windows, Mac OS X и iOS) на основе новейшей библиотеки FireMonkey. На момент выхода в свет Delphi XE2 осенью 2011 г. подобными возможностями не обладала ни одна среда разработки!

В настоящее время существуют две базовые разновидности Delphi:

- ◆ Delphi XE2 (16-я версия языка), предназначенная для создания программного обеспечения для Microsoft Windows, Mac OS X и iOS;
- ◆ Delphi Prism XE2 — язык разработки для платформы .NET (Microsoft Windows).

Эта книга посвящена наиболее новой версии языка — Delphi XE2, однако более половины примеров из книги сохраняют работоспособность и в более ранних версиях среды разработки.

Структура консольного приложения

Для начала следует обратиться к фундаментальным основам языка Delphi, поэтому чтобы не "оглушить" читателя обилием пока непонятных терминов, мы, насколько возможно, постараемся абстрагироваться от сложной для начинающего программиста визуальной библиотеки компонентов (VCL). Девизом этой части книги можно считать разумный минимализм. Именно поэтому первые шаги изучения языка разработки мы сделаем в элементарных консольных приложениях, где нет отвлекающих новичка элементов управления и код максимально прост и линеен.

Для создания заготовки консольного приложения Delphi в главном меню среды разработки выберите пункт меню **File | New | Other** (Файл | Создать | Другое). Если все сделано верно, то перед нами появится окно **New Items** (Новый элемент) с открытым набором проектов **Delphi Projects**. Найдите на этой странице значок **Console Application** (Консольное приложение) и щелкните по кнопке **ОК** (рис. 1.1). За этот труд Delphi отблагодарит нас заготовкой для самого простейшего приложения — консольного окна Windows.

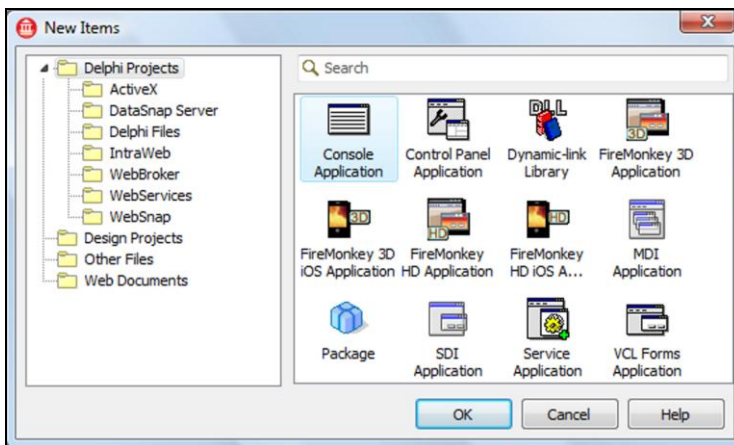


Рис. 1.1. Окно выбора типа проекта Delphi XE2

Сразу приучим себя к одному из ключевых правил программистов — исходный код программы должен регулярно сохраняться на жестком диске компьютера. Поэтому, прежде чем мы приступим к изучению заготовки консольного приложения, немедленно сохраним только что созданный проект. Для этого находим пункт главного меню среды разработки **File | Save All** (Файл | Сохранить все), в появившемся диалоге сохранения выбираем целевую (желательно пустую) папку, в которую отправятся файлы проекта. Среда проектирования обязательно спросит, под каким именем следует сохранить проект. Присвоим своему пер-

вому проекту имя (для первого раза вполне подойдет название по умолчанию — Project1.dproj). В завершение нажмем кнопку **ОК**.

Обязательно откройте целевую папку в любом файловом менеджере, например в Проводнике Windows, и посмотрите, какие файлы были созданы Delphi во время сохранения нашего проекта:

- ◆ Project1.dproj — файл в формате расширяемого языка разметки (eXtensible Markup Language, XML), содержащий основные конфигурационные и справочные сведения о нашем проекте;
- ◆ Project1.dpr — головной файл с кодом проекта на языке Delphi;
- ◆ Project1.res — файл с ресурсами приложения.

После компиляции проекта появится еще ряд файлов, в том числе исполняемый файл с расширением exe. Но сейчас для нас наибольшую ценность представляет головной файл проекта — Project1.dpr. Именно в нем находится самое важное — исходный код будущего приложения. Этот же исходный код отобразится в редакторе Delphi (рис. 1.2).

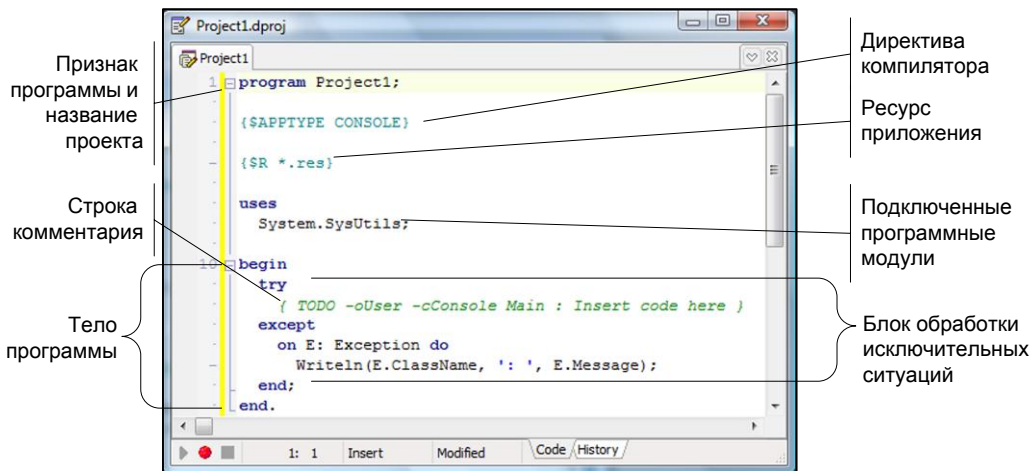


Рис. 1.2. Код консольного приложения Delphi XE2

Простейшее консольное приложение содержит немногим больше десяти строк (см. рис. 1.2). Листинг начинается с зарезервированного слова `program`. Это признак того, что перед нами листинг будущей программы. За словом `program` следует имя программы (в нашем случае `Project1`). Строкой ниже вы найдете директиву, определяющую порядок компиляции проекта. Еще ниже, после ключевого слова `uses`, приводится перечень подключаемых к проекту дополнительных программных модулей: в примере это всего лишь один модуль системных утилит — `SysUtils`, принадлежащий *пространству имен* `System`. Основной код программы располагается между ключевыми словами — `begin` и `end`. Пока здесь мы обнаружим только секцию обработки ошибок `try..except` и строку комментария.

Комментарии

Ни один, даже очень профессиональный программист, при разработке программного обеспечения не может обойтись без услуг комментариев. *Комментарий* — это не что иное, как краткое пояснение того, что происходит в тех или иных строках листинга. Человеческая

память не идеальна, и нередко нас подводит. А теперь представьте себе, что вы решили до-работать код годовалой давности. Попробуйте быстро сообразить, для чего предназначалась переменная с "мудреным" названием `x` или что произойдет после вызова процедуры `MySuperProcedure()`?

Другими словами, в комментариях разработчик кратко поясняет смысл рожденных в его голове команд. В результате листинг программы становится более понятным и доступным для изучения.

Для того чтобы при компиляции программы текст комментариев не воспринимался Delphi в качестве команд и не служил источником ошибок, приняты следующие соглашения. Комментарием считается (листинг 1.1):

- ◆ отдельная строка, начинающаяся с двух наклонных черт `//`;
- ◆ весь текст, заключенный в фигурные `{ }` или в круглые скобки с символами звездочек `(* *)`.

Листинг 1.1. Примеры комментариев

```
//Одна строка комментария
```

```
{Текст  
многострочного  
комментария}
```

```
(*Это также  
комментарий*)
```

Будьте внимательны! Если внутри фигурных скобок на первой позиции окажется символ `$`, то это не строка комментария, а директива компилятора. В шаблоне только что созданного нами приложения (см. рис. 1.2) такая директива имеется — `{$APPTYPE CONSOLE}`. Это означает, что наш проект должен стать консольным приложением.

ВНИМАНИЕ!

Директивы компилятора начинаются с символа открывающейся фигурной скобки и символа доллара `{$. . .}`. Начинающему программисту ни в коем случае не следует удалять, редактировать или изменять местоположение директив компилятора, в противном случае вы рискуете привести свой проект в негодность.

Перевод листинга программы в машинные коды

Для того чтобы листинги программы превратились в полноценное приложение, необходимо сделать их понятными для центрального процессора компьютера. Процессор не понимает языка Delphi (как, впрочем, и любых других языков высокого уровня), вместо этого процессор работает с машинными командами.

Чтобы листинг программы превратился в машинные коды, необходима помощь трех специальных подпрограмм: *препроцессора*, *компилятора* и *компоновщика*.

Препроцессор подготавливает программу к компиляции. Для этого он изучает и выполняет имеющиеся в программе директивы.

Компиляция — это процесс преобразования команд языка программирования в машинный код, понятный процессору компьютера. Во время компиляции программа-компилятор просматривает исходный код программы на предмет синтаксических ошибок и выполняет смысловой анализ кода.

На финальной стадии в дело вступает *компоновщик*. Он подключает к нашей программе все необходимые для ее работы модули, в первую очередь модули дополнительных библиотек.

В результате работы препроцессора, компилятора и компоновщика Delphi получается исполняемый файл с расширением `exe`. Полученный файл можно переносить на другой компьютер и использовать как самостоятельную программу для Windows.

ЗАМЕЧАНИЕ

Далее в книге процесс перевода листинга программы в машинные коды мы станем называть *компиляцией*.

Компилятор Delphi является высокотехнологичной программой и представляет собой ядро всей среды разработки. Для того чтобы полученная в результате компиляции программа стала высокопроизводительной, профессиональные программисты *оптимизируют параметры компиляции*. Для вызова окна с опциями компилятора следует воспользоваться пунктом меню **Project | Options** (Проект | Параметры) и в появившемся окне настроить параметры компилятора, выбрав элемент **Delphi Compiler** (рис. 1.3).

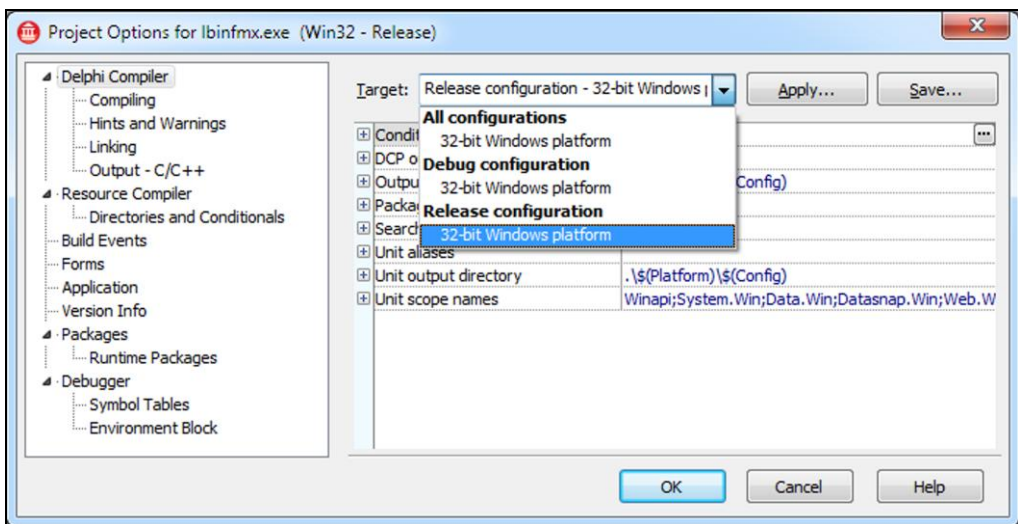


Рис. 1.3. Настройка параметров компиляции проекта

Без особой на то причины начинающему программисту не стоит экспериментировать с параметрами компилятора, однако необходимо знать следующее. По умолчанию компилятор настроен на работу в *режиме отладки* (debug). Этот режим используется во время проектирования приложения Delphi и позволяет эффективно выявлять и устранять ошибки в исходном коде. После окончательной победы программиста над всеми ошибками отладка программы завершается, самая последняя компиляция проекта осуществляется в режиме *выпуска конечного продукта* (release). Заказчику программного обеспечения ваши программы должны передаваться откомпилированными в режиме конечного продукта. Для этого следует изменить конфигурацию компилятора (**Build Configuration**), выбрав в раскрывающемся списке (см. рис. 1.3) вариант **Release configuration**.

Для старта компиляции проекта Delphi достаточно выбрать пункт меню **Run | Run** или (что еще проще) нажать клавишу <F9>. Если компилятор работает в режиме настроек по умолчанию, то в результате этих действий в папке с проектом появится дочерняя папка Debug, "порывшись" в которой вы обнаружите готовую к работе программу с расширением exe.

Первая программа

Настал тот час, когда мы готовы написать свою первую консольную программу на языке Delphi. Создайте новый проект консольного приложения и сохраните его в отдельную папку под любым именем (я выбрал имя Hello.dproj).

По давно сложившейся традиции (у ее истоков стояли известные программисты Брайан Керниган (Brian Kernighan) и Деннис Ритчи (Dennis Ritchie)) самой первой программой, открывающей книгу по программированию, станет программа-приветствие "Hello, world!", ее код представлен в листинге 1.2.

Листинг 1.2. Программа "Hello, world!"

```
program Hello;

{$APPTYPE CONSOLE} //это директива компилятора, которую мы не трогаем!

uses System.SysUtils; (*строка подключения внешних библиотек подпрограмм, хотя,
между нами говоря, в этой программе внешние модули нам не нужны*)

begin
  WriteLn('Hello, world!'); //выводим текст приветствия
  ReadLn; //ожидаем ввод — нажатие любой клавиши завершит работу консоли
end.
```

Введя код программы, обратитесь к главному меню Delphi и выберите пункт **Run | Run** или воспользуйтесь клавишей <F9>. Если вы были внимательны и переписали строки листинга, не совершив ни одной ошибки, то за считанные доли секунды на экране появятся плоды нашего коллективного творчества — консольное окно со строкой "Hello, world!". Если же была допущена ошибка, то компилятор просигнализирует о ней, выделив в листинге подозрительную строку или следующую за ней строку.

Для вывода текстовой строки на экран компьютера нам понадобились услуги процедуры WriteLn(). Эта процедура еще не раз встретится нам в других примерах к этой книге, пока вам достаточно запомнить то, что WriteLn() позволяет отобразить в окне консольной программы текст, но при условии, что символы текста заключены в одинарные кавычки.

Завершая вводную главу, попрошу запомнить вас еще одну особенность синтаксиса языка Delphi — каждое выражение должно завершаться *точкой с запятой*, а признаком окончания всей программы является ключевое слово *end с точкой на конце*.



ГЛАВА 2

Типы данных Delphi

Функциональное назначение любой программы, будь это простейшее консольное приложение, выводящее на экран приветствие "Hello, world", или сложный математический пакет, моделирующий термоядерную реакцию, заключается в получении, обработке и возврате пользователю определенных результатов. Обслуживаемые программой данные размещаются в памяти компьютера в виде разнообразных структур, в простейшем случае это обычная последовательность байтов, а в более сложных ситуациях структура представляет собой весьма неординарную конструкцию, которая способна не просто хранить, но и управлять своими данными.

Сложность программы состоит в прямой зависимости от способов представления данных в памяти и задействованных для обслуживания этих данных алгоритмов. Николаусу Вирту принадлежит меткое определение программы:

Программа = структуры + алгоритмы.

Заметьте, что в определении профессора Вирта на первом месте стоят структуры, а алгоритмы только на втором. Это не случайность. Дело в том, что даже самый совершенный и глубоко проработанный алгоритм окажется малоэффективным, если подлежащие обработке данные предоставляются алгоритму в ненадлежащем виде. Здесь все как в автомобиле, никакой даже лучший на свете двигатель (читай — алгоритм) не в состоянии сдвинуть автомобиль с места, если к нему не прикручены колеса.

В этой главе мы начнем обсуждение первого ингредиента программы — структур. Здесь мы рассмотрим простейшие способы представления данных в программах Delphi, более сложные решения будут вынесены в следующую главу книги.

Переменные

Наиболее распространенный способ хранения данных реализуется при посредничестве *переменных*. Переменные позволяют размещать в памяти значения различных типов (целые и вещественные числа, символы, текстовые строки). Термин "переменная" (variable) подсказывает, что в ходе выполнения программы содержимое памяти может изменяться.

Объявление переменной включает упоминание слова `var` и указание имени и типа переменной.

```
var имя_переменной : тип данных;
```

Если необходимо объявить несколько одноименных переменных, то разрешается перечислить их, разделяя имена запятыми так, как представлено в листинге 2.1.

Листинг 2.1. Объявление переменных

```
var a: integer; //переменная для хранения целого числа
    b: boolean; //переменная для хранения логического значения
    c, d : real; //переменные для обслуживания действительных чисел
```

Более подробно о типах данных мы поговорим немного позднее, а пока достаточно понимать, что тип данных определяет особенность хранения и представления значений в переменных. Если необходимо оперировать целыми числами, то, скорее всего, вам пригодится тип данных `integer`, для обслуживания вещественных чисел воспользуйтесь типом `real`, для манипуляций отдельными символами понадобится `Char`.

ПРИМЕЧАНИЕ

Пока, при встрече с термином "тип данных", нам достаточно понимания того, что тип данных определяет число битов памяти, отдаваемых в распоряжение переменной. Чем больше битов, тем данные большего размера могут быть переданы в переменную. Добравшись до последних страниц главы, читатель увидит, что тип данных представляет собой более сложное понятие.

Современные версии Delphi позволяют проинициализировать содержимое переменной в момент ее объявления (листинг 2.2).

Листинг 2.2. Инициализация переменной во время объявления

```
var i: integer=10; //целочисленная переменная i содержит значение 10
    b: boolean=false; //логическая переменная b содержит false
    c: char='!'; //символьная
```

Унаследованный от языка Pascal классический стиль программирования строго устанавливает области кода, в которых допускается объявление переменных. В консольных приложениях переменные должны объявляться перед телом программы, которое, в свою очередь, заключается внутрь составного оператора `begin..end`. При описании функции (процедуры, метода) все объявления локальных переменных должны быть сосредоточены между заголовком процедуры и началом ее реализации (листинг 2.3).

Листинг 2.3. Объявление локальной переменной в функции

```
function MyFunction: integer;
var i: integer; //локальная переменная i
begin
    i:=0; //переменная i будет доступна только внутри функции
    ...
end;
```

В зависимости от места объявления переменной можно говорить о ее локальном или глобальном статусе. *Глобальные переменные* проекта доступны или, как говорят программисты, видимы из всех областей программы. К такой переменной можно обратиться из любой процедуры или функции. В противовес глобальным переменным, область видимости их локальных "коллег" существенно уже. Например, переменная, объявленная внутри функции (см. листинг 2.3), может быть задействована только в границах этой функции. Попытка

прочитать содержимое локальной переменной извне области ее видимости обречена на неудачу.

Константы

В отличие от переменных *константы* (constants) предназначены для хранения заранее известных и неизменяющихся во время выполнения программы значений. Объявление константы начинается с ключевого слова `const` и включает имя, необязательное указание типа и значение константы.

```
const имя_константы: [необязательный тип] = значение;
```

Пример объявления констант предложен в листинге 2.4. Обратите внимание на то, что для определения значения константы могут применяться выражения, например константа `c` хранит результат операции деления 5 на 3.

Листинг 2.4. Порядок объявления констант

```
const a = 3.14;  
      b = 65535;  
      c : extended = 5/3;
```

Так же как и в случае с переменными, в Delphi константа должна быть объявлена в строго отведенном для этого месте программы (идентичном местам объявления переменных). По аналогии с переменными константы обладают областью видимости, поэтому различают *глобальные* и *локальные константы*.

Строки-ресурсы

К одной из разновидностей констант стоит отнести *строки-ресурсы* (resource strings). Строки-ресурсы состоят из текстовой информации и специальных символов форматирования (применяемых в функциях `Format()` и `FormatDateTime()`), вместо которых позднее можно вставить дополнительную информацию.

Листинг 2.5 демонстрирует один из вариантов применения строк-ресурсов.

Листинг 2.5. Порядок объявления констант

```
resourcestring ResDateTime = 'Дата d.mm.yyyy Время hh:mm';  
var s:string;  
begin  
    S := FormatDateTime(ResDateTime, Now);  
    WriteLn(S);  
end.
```

В предложенном примере символы `d.mm.yyyy` замещаются текущей датой, а символы `hh:mm` — текущим временем.

Правила объявления идентификаторов

Все используемые в программах Delphi переменные, константы, процедуры, функции, объекты должны обладать уникальным именем — *идентификатором*.

В языке Delphi длина идентификатора не ограничена, но значащими являются только первые 255 символов. Идентификатор может содержать любые символы латинского алфавита, цифры и символ нижнего подчеркивания. Первый символ идентификатора обязан быть буквой. Само собой, в роли идентификаторов не допускается применять зарезервированные слова.

Различают *неквалифицированные* и *квалифицированные идентификаторы*. Синтаксис последних выглядит следующим образом:

идентификатор1.идентификатор2

Подобное обращение необходимо в тех ситуациях, когда требуемая переменная (функция, метод, свойство, объект и т. п.) принадлежит другому модулю (объекту, классу и т. д.). Таким образом, квалифицированный идентификатор знает всю цепочку своих владельцев.

Неквалифицированные идентификаторы не требуют явного указания своих владельцев. Это допускается в том случае, когда в принадлежности идентификатора сомнений не возникает.

ЗАМЕЧАНИЕ

В отличие от идентификаторов языка Си, идентификаторы языка Delphi *не чувствительны к регистру символов*, другими словами, объявив переменную `myvariable`, программисту разрешено обращаться к ней, не реагируя на регистр символов, например `MYVARIABLE` или `MyVariable`.

Типы данных

Отправной точкой процесса построения любой системы хранения и обработки данных по праву считается выбор *типа данных*. Вне зависимости от того, какой язык программирования выбран для разработки проекта, первым шагом становится рассмотрение типов данных, имеющихся в распоряжении системы.

В первую очередь тип данных определяет порядок хранения данных в памяти компьютера. Физическая концепция хранения данных зависит от особенностей конкретной платформы, на базе которой функционирует программное обеспечение. В первую очередь это утверждение относится к указателям. Например, в 32-разрядных ОС размер указателя равняется 4 байтам. В 64-битных системах это утверждение ошибочно, здесь для указателя требуется 8 байт.

ВНИМАНИЕ!

В Delphi XE2 появилась возможность создавать 64-разрядные приложения для Windows. Во всех предыдущих версиях Delphi эта опция отсутствовала.

Типизация хранимых значений не просто указывает на размер в байтах, которую должна выделить система для размещения в памяти того или иного значения. Преследуемая цель еще более значима. Типизация определяет, какие операции могут быть осуществлены с теми или иными данными. Delphi не позволит новичку передать результаты деления в целочисленную переменную, ведь в этом случае есть риск потерять дробную часть результата. Delphi станет отчаянно сопротивляться, если мы попробуем просуммировать символьный и вещественный типы данных, пусть даже в символьной переменной хранится числовое зна-

чение. В последнем случае перед проведением математической операции необходимо выполнить преобразование данных.

Программы функционируют в интересах человека, поэтому результаты их работы должны представляться в удобном для оператора виде. Во многом это достигается благодаря концепции типа данных, например при выводе на экран действительных чисел отображается целая и дробная части, разделенные запятой, а при выводе значений, специализирующихся на работе с датой и временем, дата и время форматируются так, как в принято в вашей стране.

Подытожим все вышесказанное. Понятие "тип данных" интегрирует в себе три компонента:

- ◆ ограничение множества значений, принадлежащих типу;
- ◆ дефиниция¹ набора операций, применяемых к типу;
- ◆ определение способа отображения (внешнего представления) значений типа.

Язык Delphi предоставляет в распоряжение программиста весьма широкий спектр *предопределенных типов данных* и, кроме того, разрешает конструировать собственные *пользовательские типы*. Вариант классификации фундаментальных типов данных Delphi представлен на рис. 2.1.

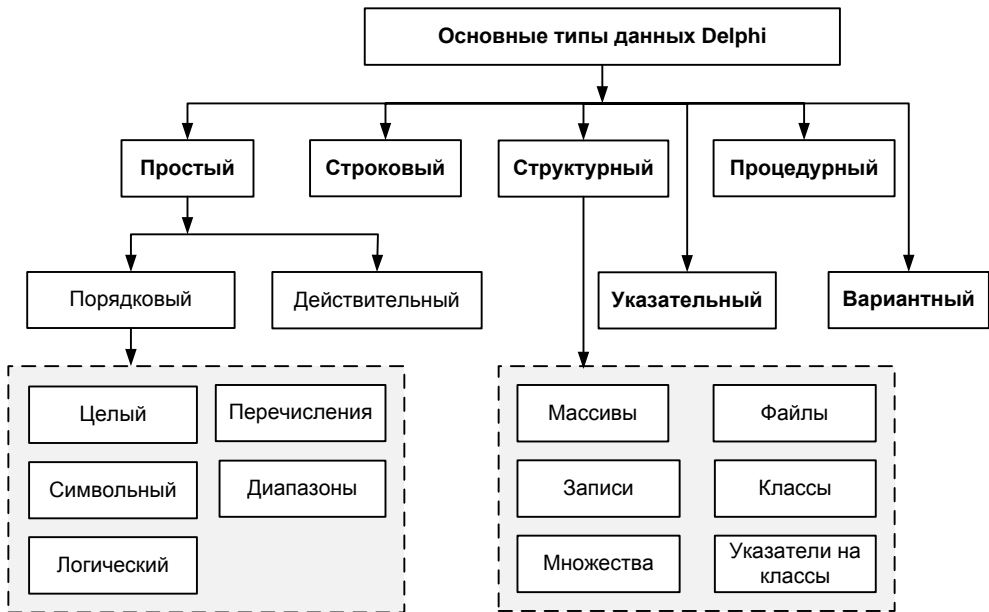


Рис. 2.1. Классификация типов данных Delphi

Все существующие в языке Delphi типы данных можно разделить на шесть ветвей:

- ◆ простые;
- ◆ строковые;
- ◆ структурные;
- ◆ указательные;
- ◆ процедурные;
- ◆ варианты типов данных.

¹ Краткое определение какого-либо понятия, содержащее наиболее существенные его признаки.

В этой главе мы изучим простые, строковые и указательные типы данных, с оставшимися типами мы познакомимся в других главах книги.

Простые типы

Из представленных на рис. 2.1 типов данных наиболее востребованным является *простой*. Простой тип специализируется на обслуживании числовых данных и упорядоченных последовательностей, тип разделяется на два направления: *порядковые* и *действительные* типы.

Своим названием порядковый тип обязан тому, что между элементами этого типа существуют отношения порядка, другими словами, они могут расположиться в виде упорядоченной последовательности. К еще одной особенности порядковых типов данных стоит отметить то, что все без исключения простые типы по своей сути являются целыми числами.

Целые числа

Характерным представителем порядкового типа считаются целые числа, их характеристики вы найдете в табл. 2.1.

Таблица 2.1. Целые числа

Тип	Диапазон значений	Размер (байт)
Shortint, Int8	-128...127	1
Smallint, Int16	-32 768...32 767	2
Longint, Integer, Int32	-2 147 483 648...2 147 483 647	4
Int64	$-2^{63} \dots 2^{63} - 1$	8
Byte, UInt8	0...255	1
Word, UInt16	0...65 535	2
Longword, UInt32	0...4 294 967 295	4
UInt64	$0 \dots 2^{64} - 1$	8

Символьный тип

Второй по счету представитель простых типов — символьный тип данных — специализируется на хранении кода символа. В современных версиях Delphi по умолчанию используется кодировка Unicode. Один символ в такой кодировке может занимать до 4 байт, что позволяет закодировать все символы национальных алфавитов. Вместе с тем, Delphi поддерживает и более старую однобайтовую кодировку ANSI (табл. 2.2).

Таблица 2.2. Символьный тип данных

Тип	Кодировка	Размер в байтах
AnsiChar	ANSI	1
WideChar, Char	Unicode	От 1 до 4