Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра «Вычислительные методы и программирование»

С.В.Колосов

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В СРЕДЕ DELPHI

Лабораторный практикум для студентов всех специальностей

Минск 2001

УДК 681.3 (075)

ББК 32.973.26-018 я 73

К-61

Рецензент: канд. техн. наук, доц. кафедры «Вычислительная техника»
А.И. Шакирин, БАТУ

Колосов С. В.

Объектно-ориентированное программирование в вреде Delphi. Лабораторный практикум для студентов всех специальностей. Мн.: БГУИР, 2001. – 47 с.

ISBN 985-444-296-9

В лабораторном практикуме приведены краткие теоретические сведения по основам программирования в среде Delphi и созданию программ работы с графическими объектами и базами данных.

В практикум вошло 4 лабораторные работы.

УДК 681.3 (075)
ББК 32.973.26-018 я 73

ISBN

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА 1. ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ В DELPHI

Цель лабораторной работы: Изучение основ объектно-ориентированного программирования.

1.1 Объекты и классы

Класс - это тип объекта. Он характеризует объект вместе с его свойствами и правилами поведения. Объект есть экземпляр класса. В Delphi все объекты динамические. Для выделения памяти под объект используется специальный метод данного класса Constructor Create. Освобождение памяти, выделенной ранее объекту, осуществляется методом Dectructor Destroy. Для объектов и классов сделано одно исключение из общих правил работы на Паскале с динамическими переменными – для них не используется символ ^ (тильда) для указателей и содержимого указываемой памяти. Переменные, описанные в классе, называют полями. Любая процедура или функция, описанная в классе, является уже методом. При вызове любого метода ему неявным образом первым параметром передается параметр Self, который является указателем на объект, который вызвал данный метод. Если процедура описана вне класса, но за ее описанием следуют слова of object, то это тоже будет метод. Классы могут быть описаны или в интерфейсной части модуля Unit или в самом начале секции реализации Implementation. Не допускается их описание внутри процедур и функций. Если перед описанием метода стоит ключевое слово class, то это классовый метод и его можно вызывать даже в том случае, если объекту еще не была выделена память. Типичный пример класса:

```
Type TmyObject=class(Tobject)
   x,y:integer;
   Constructor Create;
   Destructor Destroy;virtual;
   Procedure Show;
   End;
```

В скобках после ключевого слова class указывается наследуемый класс. Объект Tobject является прародителем всех классов и не наследует никаких других классов.

1.2 Области видимости класса

1. Private – личная, внутренняя область класса. Поля и методы, описанные в этой области, доступны только внутри модуля Unit, где описан данный класс. Для всех других модулей, которые подсоединяют данный модуль и наследуют этот класс, они недоступны.
2. Protected – защищенная область. Поля и методы этой области доступны только внутри классов, наследующих данный класс.
3. Public – общедоступная область. Поля и методы этой области не имеют ограничений на видимость.
4. Published – область публикаций. Поля и методы этой области имеют такую же видимость, как для области public, но они еще видны инспектору объектов

на этапе разработки программы. В дочерних классах можно переносить методы и свойства из области Protected в область Published и обратно.

1.3 Свойства (Property) и инкапсуляция

Объектно - ориентированное программирование (ООП) основано на трех принципах – инкапсуляция, наследование и полиморфизм. Классическое ООП утверждает, что чтение и обновление полей должно производиться только специальными методами и не допускается прямое обращение к полям класса. Это правило и называется инкапсуляцией, а такие поля - свойствами. Свойство определяется полем и двумя методами, которые осуществляют чтение и запись заданных значений в поле. Пример определения свойства:

```
Type TmyObject=class(Tobject)
        Privete
                FmyField:String;
        Protected
                Procedure SetMyField(Value:String);
        Published
                Property MyProp:String Read FmyField
                        Write SetMyField
                        Default ‘Начальное значение’;
        End;
```

Здесь в классе TmyObject определено свойство MyProp строкового типа. В качестве метода чтения выступает само значение строки, а запись осуществляется методом SetMyField. Само поле FmyField определено в области Privete и к нему поэтому нет прямого доступа из других модулей. Метод чтения этого поля находится в защищенной области, а свойство MyProp - в области публикаций и доступно инспектору объектов во время проектирования программы.

1.4 Методы, наследование и полиморфизм

Наследование означает, что при создании нового класса он наследует все поля, свойства и методы, определенные в родительском классе. В новом классе только добавляются новые поля, методы и свойства. Унаследованные от предка поля и методы доступны в дочернем классе, но с учетом областей видимости. Если имеет место совпадение имен, то говорят, что они перекрываются. В Delphi допускается только последовательное единичное наследование классов. Методы подразделяются на 4 группы:

статические (Static),
виртуальные (Virtual),
динамические (Dynamic) и
абстрактные (Abstract).

Адрес вызова статического метода определяется на этапе трансляции проекта и вызов этих методов осуществляется быстрее всех остальных методов. Такие методы можно без ограничений перекрывать, при этом можно менять

список передаваемых параметров. По умолчанию методы объектов являются статическими.

Адреса виртуальных и динамических методов определяются во время выполнения программы и находятся в специальных таблицах: таблице виртуальных методов (VMT) и таблице динамических методов (DMT). В таблицу VMT включаются адреса всех определенных в данном классе виртуальных методов и всех наследуемых методов. В таблицу DMT включаются адреса динамических методов определенных только в данном классе. Поэтому виртуальные методы вызываются быстрее динамических, но размеры таблиц VMT существенно больше таблиц DMT. Динамические методы позволяют экономить память, но их вызов осуществляется медленнее всех остальных методов, т.к. приходится для поиска адреса метода проходить по всем таблицам DMT родительских классов пока не найдем нужный нам динамический метод.

Для перекрытия виртуальных и динамических методов используется ключевое слово Override. Список параметров перекрываемых виртуальных и динамических методов не должен отличаться от списка параметров этих методов в родительском классе.

Абстрактные методы определяют только интерфейсную часть метода, такие методы нельзя использовать без перекрытия в дочерних классах, где должна находиться и реализация такого метода.

Рассмотрим следующий пример.

```
Type Tpoint=Class(Tobject)
                Constructor Create;
                Destructor Destroy;virtual;
                X,y:Integer;
                C:Tcolor;
                Procedure Show;virtual;
                End;
        Tcircle=Class(Tpoint)
                Constructor Create;
                Destructor Destroy;override;
                R:Integer;
                Procedure Show;override;
                End;
……….
Var     Point1:Tpoint;
        Circle1:Tcircle;
Begin
        Point1.Create;
        Circle1.Create;
        With Point1 do Begin
                X:=100;y:=50; C:=clRed;
                Show;
                End;
```

```
With Circle1 do Begin
        X:=200;Y:=100:C:=clBlue;
        R:=50;
        Show;
        End;
……….
```

В данном примере определены два класса Tpoint и Tcircle. Класс Tpoint наследует класс Tobject. В классе Tpoint определены поля X, Y, которые задают точку на экране дисплея, и C – цвет этой точки. В нем также описаны методы по выделению и освобождению памяти под объект и виртуальный метод Show - рисования точки на экране. Класс Tcircle наследует класс Tpoint и все его поля и методы. В нем дополнительно описаны поле R (радиус) и метод Show, который перекрывает аналогичный метод класса Tpoint. Метод Show класса Tcircle рисует уже окружность. В результате два метода Show рисуют разные картинки, в зависимости от того, какому классу они принадлежат. Это и называется полиморфизмом объектов.

1.5 События (Events)

События в Delphi - это свойства процедурного типа, предназначенные для создания пользовательской реакции на те или иные входные воздействия. Пример объявления события.

```
Property OnMyEvent:TmyEvent    Read FOnMyEvent
                               Write FonMyEvent;
```

Присвоить такому свойству значение – это значит указать адрес метода, который будет вызываться в момент наступления события. Такие методы называются обработчиками событий. События имеют разные типы, но общим для всех является параметр Sender – указатель на объект источник события. Самый простой тип события это тип

```
TnotifyEvent=procedure(Sender:Tobject) of Object;
```

Здесь «of object» означает, что данный тип определяет именно метод, а не обычную процедуру. Приставка On в имени свойства означает, что данное свойство является событием, хотя не каждое событие может иметь такую приставку. Для определения события необходимо в разделе Private объявить поле указателя на метод. В разделе Protected нужно объявить методы чтения и записи адреса обработчика события в это поле и объявить событие, как свойство процедурного типа. В инспекторе объектов есть две страницы: страница свойств (Properties) и страница событий (Events). Двойной щелчок левой клавишей мыши по событию приводит к появлению обрамления обработчика события в тексте программного модуля Unit.

Далее приведем основной список событий. Общими для всех компонентов являются события (наследники класса TControl):

OnClick – нажатие левой клавиши мыши,

OnDblClick – двойной щелчок левой клавиши мыши,

OnMouseDown – нажатие любой клавиши мыши,

OnMouseMove – перемещение курсора мыши по компоненту,

OnMouseUp – отжатие кнопки мышки.

Общими для оконных элементов управления являются события (наследники класса (TWinControl):

OnEnter – перемещение фокуса ввода на компонент, данный компонент становится активным,

OnExit – потеря активности компонентом,

OnKeyDown – нажатие клавиши или комбинации клавиш,

OnKeyPress – нажатие каждой одиночной клавиши,

On KeyUp – отпускание клавиши.

Вопросы

1. Чем класс отличается от объекта?
2. Объекты бывают статическими или динамическими?
3. Для каких целей используется метод Create?
4. Что собой представляет неявно передаваемый в объект параметр Self?
5. Области видимости класса.
6. Что такое свойства объектов?
7. Что обозначает принцип инкапсуляции в ООП?
8. Чем метод отличается от обычной процедуры?
9. Какие вы знаете типы методов?
10. Что означает принцип наследования классов?
11. Что такое полиморфизм в ООП?
12. Что такое событие и чем оно отличается от свойства класса?
13. Приведите примеры основных событий компонентов?
14. Чем динамические методы отличаются от виртуальных?
15. Где можно давать определение классу?

ЛАБОРАТОРНАЯ РАБОТА 2. СТАНДАРТНЫЕ И ТИПОВЫЕ ДИАЛОГИ, МНОГОСТРАНИЧНЫЕ БЛОКНОТЫ

Цель лабораторной работы: научиться использовать стандартные и типовые диалоги и многостраничные блокноты.

2.1. Стандартные диалоги

Основные стандартные диалоги описаны в [1, С.81-85].

2.2. Типовые диалоги

Типовые диалоги вызываются, как обычные процедуры. Далее приведены основные из них.

Procedure ShowMessage(const Msg:String);

- вывод сообщения Msg по центру экрана в виде диалогового окна с одной кнопкой “Ok” .

Procedure ShowMessagePos(const Msg:String; X,Y:integer);

- вывод сообщения Msg в заданную точку экрана.

Function MessageDlg(const Msg:String; Atype:TmsgDlgType;
Abutton:TmsgDlgButton; HelpCtr:Longint):word;

- вывод сообщения Msg с возможностью выбора ответа на сообщение.

Здесь TmsgDlgType=(MtWarning, предупреждение
MtError, ошибка
MtInformation, информация
MtConfirmation подтверждение
MtCustom); без иконки
- тип пиктограммы (иконки) в диалоговом окне.

TmsgDlgButtons=set of TMsgDlgBtn; множество кнопок.

TmsgDlgBtn=(MbYes, да
MbNo, нет
MbOk, хорошо
MbCancel, закончить
MbAbort, прервать
MbRetry, повторить
MbIgnore, игнорировать
MbAll, для всех
MbHelp); помощь
- возможные значения кнопок с рисунками.

Например, оператор:

```
Case MessageDlg('Удалить файл? Да или нет', mtWarning, [MbYes,MbNo], 0) of
    MrYes: Begin            // получен ответ да
        End;
    MrNo: Begin             // получен ответ нет
        End;
    End;
```

На экране возникает диалоговое окно с иконкой предупреждения, текстом «Удалить файл? Да или нет» и двумя кнопками со словами «Yes» и «No».

2.3. Многостраничные компоненты.

Многостраничные компоненты используются в случае, если на одну форму нельзя поместить все требуемые компоненты.

Компонент TtabbedNotebook со страницы Win3.1.

Он представляет собой многостраничный блокнот с закладками. Основные свойства этого компонента:

Property ActivePage:String; - имя активной страницы,

Property PageIndex:Integer; - номер активной страницы. Страницы нумеруются с нуля.

Property Pages:Tstrings; - список имен всех страниц.

На этапе проектирования программы с помощью свойства PageIndex делается активной нужная страница и в нее помещаются нужные визуальные компоненты. Затем активной делается другая страница и в нее помещаются другие компоненты и т.д.

Компонент TpageControl со страницы Win32.

Он подобен предыдущему, но имеет свои особенности. На этапе проектирования для добавления новой страницы нужно щелкнуть правой кнопкой мыши и выбрать один из пунктов меню – New Page (новая страница), Next Page (следующая страница) или Previous Page (предыдущая страница).

2.4. Дополнительные компоненты

Компонент TTtimer со страницы System.

Это невизуальный компонент. Он используется для отсчета интервалов реального времени. Его свойство

Property Interval:word; - задает интервал времени в миллисекундах, который проходит между событиями OnTimer. Величина этого интервала реально всегда будет кратна 55 миллисекундам. Это определяется системными часами, которые вызывают прерывания по времени через каждые 55 миллисекунд. Свойство:

Property Enable:Boolean; - включает и выключает таймер.

Компонент Tgauge.

Он предназначен для отображения процесса изменения какой-либо величины. Он имеет несколько вариантов отображения, что определяется свойством

Property Kind:TgaugeKind;

Кроме этого свойства есть следующие:

Property MinValue:LongInt; - минимальное значение диапазона изменения свойства Progress.

Property MaxVakue:LongInt; - максимальное значение диапазона изменения свойства Progress.

Property Progress:LongInt; - текущее значение отображаемой в компоненте изменяющейся величины.

Компонент TColorGrid со страницы Samples.

Он предназначен для выбора цветов из 16 - цветной палитры. Основной цвет выбирается щелчком левой кнопки мыши и отображается символами FG, фоновый цвет выбирается правой кнопкой мыши и отображается символами BG, при совпадении цветов отображаются символы FB.

Property BackGroundColor:TColor; - цвет фона,

Property ForeGroundColor:Tcolor; - основной цвет.

При смене цветов вызывается событие OnChange.

2.5. Оперативные подсказки

Они определяются свойствами:

Property Hint:string; - текст подсказки,

Property ShowHint:Boolean; - активность подсказки,

При попадании курсора на компонент через определенное время рядом с этим компонентом может возникать оперативная подсказка, которая находится в свойстве Hint. Текст подсказки может разделяться на две части вертикальной чертой. При этом первая часть подсказки передается в окно подсказки, а вторая часть присваивается свойству Hint объекта Application. Этот объект имеет еще несколько полезных свойств.

Property HintColor:Tcolor; - цвет окна подсказки.

Property HintPause:Integer; - время нахождения курсора мыши на компоненте, необходимое для начала показа подсказки (стандартно 800мс.).

Property HintHidePause:Integer; - время показа подсказки в миллисекундах (стандартно – 2500мс).

У объекта Application значение свойства ShowHint нужно устанавливать в True при первоначальном запуске программы, например, в обработчике события OnCreate главной формы приложения. При смене текста в свойстве Hint, что происходит при смене текущего элемента управления, у Application возникает событие

Property OnHint:TnotifyEvent;

Пример:

```
Procedure Tform1.AppHint(Sender:Tobject);
  Begin
    Panel1.Caption:=Application.Hint;
    End;
Procedure Tform1.FormCreate(Sender:Tobject);
  Begin
    Application.OnHint:=AppHint;
    Application.ShowHint:=True;
    End;
```

В этом примере вторая часть подсказки будет выводиться в компонент Panel1.

2.6 Управление курсорами

Каждому визуальному компоненту можно поставить в соответствие определенный вид курсора с помощью свойства

Property Cursor:Tcursor;

Type Tcursor=-32768..+32767;

Общим для всех компонентов вначале является вид курсора для глобального объекта Screen.

Стандартные виды курсоров Delphi и соответствующие им константы приведены ниже.

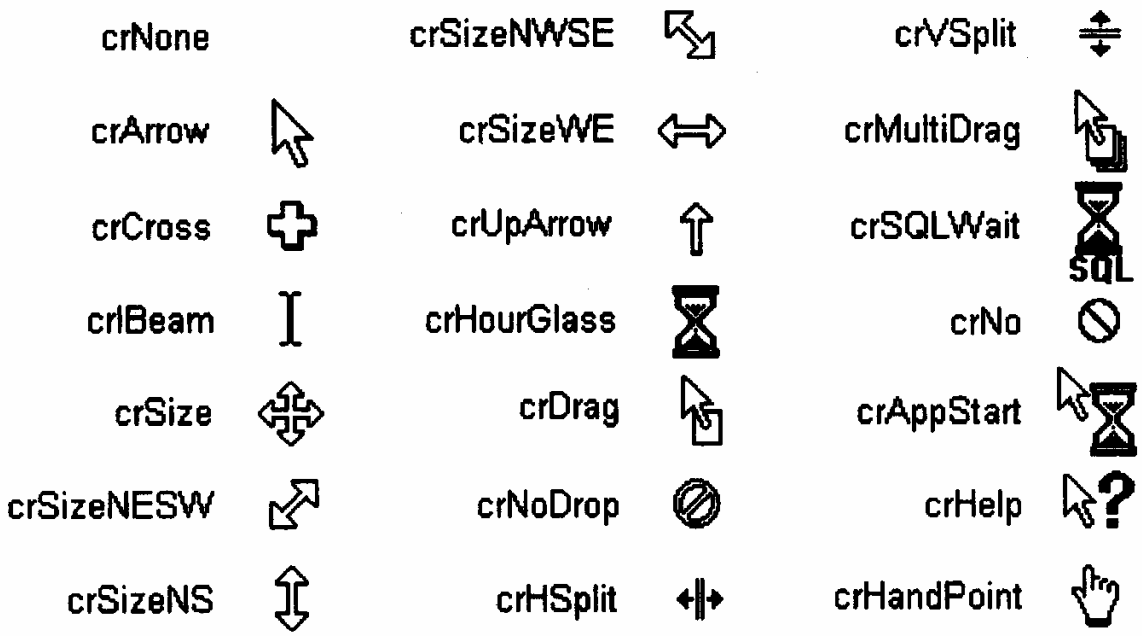


Рис.2.1 Стандартные виды курсора

Вид курсора может задать сам программист в графическом редакторе Delphi и сохранить его в ресурсном файле (расширение *.res). В тексте программы подключение этого файла осуществляется оператором {$R *.RES}. При создании главной формы следует загрузить новый курсор из ресурсного файла оператором

Screen.Cursors[1]:=LoadCursor(Hinstance,'MyCursor');

После этого ваш курсор становится доступным и имеет номер 1. Стандартные курсоры имеют номера от –22 до 0. Пользователь для своих курсоров может использовать все остальные номера.

2.7 Компоненты визуализации процесса

Компонент TGauge предназначен для представления процесса изменения какой-нибудь величины различным образом.

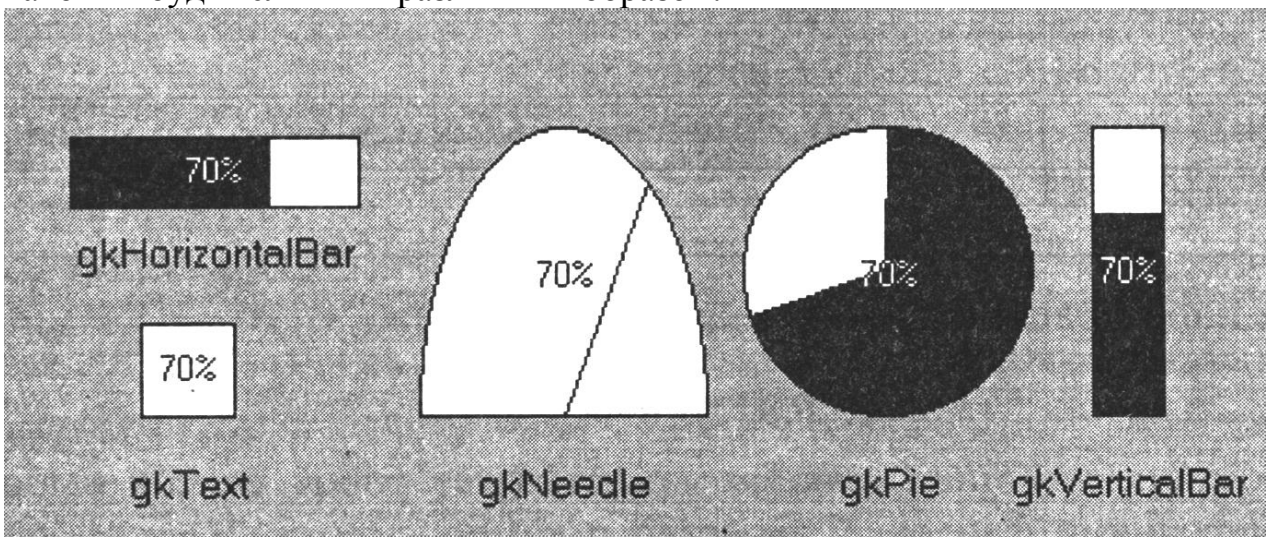


Рис.2.2 Различные формы представления компонента Tgauge

Свойства компонента Таблица 1

Property BackColor: TColor;	Цвет незакрашенной части индикатора
Property ForeColor: TColor;	Цвет закрашенной части индикатора
TGaugeKind = (gkText, gkHorizontalBar, gkVerticalBar, gkPie, gkNeedle); Property Kind: TGaugeKind;	Определяет форму индикатора (см. рис. 2.2)
Property MaxValue: Longint;	Определяет максимальное значение диапазона изменения свойства *Progress*
Property MinValue: Longint;	Определяет минимальное значение диапазона изменения свойства *Progress*
Property PercentDone: Longint;	Содержит текущее значение *Progress* в процентах от его диапазона изменения

Продолжение табл. 1

Property Progress: Longint;	Содержит текущее значение изменяющейся числовой величины
Property ShowText: Boolean;	Если содержит *True,* в центре компонента выводится строковое представление значения *PercentDone*

Метод Procedure AddProgress(Value: Longint); добавляет к текущему значению Progress величину Value. Аналогичные функции выполняют компоненты TProgressBar и TtrackBar.

Задания

1. На форму поместить во всю клиентскую область компонент TTabbedNoteBook, содержащий две страницы. На первой странице поместить компоненты Tmemo, TbitBtn и TopenDialog. Компонент TbitBtn должен иметь кроме текста пиктограмму для каждого состояния кнопки. При нажатии на кнопку TbitBtn нужно запускать диалог открытия текстового файла и поместить его содержимое в компонент Tmemo, при этом на дисплее должно появляться сообщение об открытии файла с помощью типового диалога ShowMessage. На второй странице компонента TtabbedNoteBook поместить компоненты Ttimer, Tgauge и TcolorDrid. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой горизонтально заполняемый прямоугольник с текстом процентного заполнения этого прямоугольника. С помощью компонента TcolorGrid предусмотреть изменение цвета фона и заполнения в компоненте Tgauge. Все визуальные компоненты должны иметь всплывающие подсказки (Hint).
2. На форму поместить во всю клиентскую область компонент TTabbedNoteBook, содержащий две страницы. На первой странице поместить компоненты Tmemo, два TbitBtn, TopenDialog и TSaveDialog. Компонент TbitBtn должен иметь кроме текста пиктограмму для каждого состояния кнопки. При нажатии на кнопку TbitBtn1 нужно запускать диалог открытия текстового файла и поместить его содержимое в компонент Tmemo, при этом на дисплее должно появляться сообщение об открытии файла с помощью типового диалога ShowMessage. При нажатии на кнопку TbitBtn2 нужно запускать диалог сохранения содержимого Tmemo в текстовом файле, при этом на дисплее должно появляться сообщение об сохранении файла с помощью типового диалога ShowMessagePos. На второй странице компонента TtabbedNoteBook поместить компоненты Ttimer, Tgauge и TcolorDrid. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой вертикально заполняемый прямоугольник с текстом процентного заполнения этого прямоугольника. С помощью компонента TcolorGrid предусмотреть изменение цвета фона и заполнения в компоненте Tgauge. Все визуальные компоненты должны иметь всплывающие подсказки (Hint).

3. На форму поместить во всю клиентскую область компонент TTabbedNoteBook, содержащий две страницы. На первой странице поместить компоненты TImage, TbitBtn и TopenPictureDialog. Компонент TbitBtn должен иметь кроме текста пиктограмму для каждого состояния кнопки. При нажатии на кнопку TbitBtn нужно запускать диалог открытия файла с рисунком и поместить его содержимое в компонент TImage, при этом на дисплее должно появляться сообщение об открытии файла с помощью типового диалога ShowMessage. На второй странице компонента TtabbedNoteBook поместить компоненты Ttimer, Tgauge и TcolorDrid. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой панель спидометра с текстом процентного заполнения этой области. С помощью компонента TcolorGrid предусмотреть изменение цвета фона и заполнения в компоненте Tgauge. Все визуальные компоненты должны иметь всплывающие подсказки (Hint).
4. На форму поместить во всю клиентскую область компонент TTabbedNoteBook, содержащий две страницы. На первой странице поместить компоненты TImage, два TbitBtn, TopenPictureDialog и TsavePictureDialog. Компоненты TbitBtn должены иметь кроме текста пиктограмму для каждого состояния кнопки. При нажатии на кнопку TbitBtn1 нужно запускать диалог открытия файла с рисунком и поместить его содержимое в компонент TImage, при этом на дисплее должно появляться сообщение об открытии файла с помощью типового диалога MessageDlg. На второй странице компонента TtabbedNoteBook поместить компоненты Ttimer, Tgauge и TcolorDrid. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой сектор окружности с текстом процентного заполнения этой области. С помощью компонента TcolorGrid предусмотреть изменение цвета фона и заполнения в компоненте Tgauge. Все визуальные компоненты должны иметь всплывающие подсказки (Hint).
5. На форму поместить во всю клиентскую область компонент TTabbedNoteBook, содержащий две страницы. На первой странице поместить компоненты: два TlistBox и три TbitBtn. При нажатии на первую кнопку должен запускаться диалог открытия текстового файла и следует поместить текст выбранного файла в первый TlistBox1. Выбранные строки из TlistBox1 при нажатии на вторую кнопку должны переноситься во второй TlistBox2. Третья кнопка предназначена для вызова диалога сохранения содержимого второго TlistBox2 в другом файле. На второй странице TtabbedNoteBook следует разместить компоненты Ttimer, Tgauge, TcolorDialog и две кнопки TspeedButton. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой горизонтально заполняемый прямоугольник с текстом процентного заполнения этого прямоугольника. Две кнопки предназначены для вызова диалога TcolorDialog, который должен управлять цветом фона или цветом заполнения компонента Tgauge. Все визуальные компоненты должны иметь всплывающие подсказки (Hint).

6. На форму поместить во всю клиентскую область компонент TTabbedNoteBook, содержащий две страницы. На первой странице поместить компонент Tpanel. В нем разместить три кнопки TspeedButton с зависимой фиксацией. На эту же страницу поместить компонент Tshape. В зависимости от того, какую из кнопок TspeedButton нажали в компоненте Tshape, должна меняться картинка в пределах возможных значений свойства Shape. На второй странице компонента TtabbedNoteBook поместить компоненты Ttimer, Tgauge и TcolorDrid. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой сектор окружности с текстом процентного заполнения этой области. С помощью компонента TcolorGrid предусмотреть изменение цвета фона и заполнения в компоненте Tgauge. Каждый из визуальных компонентов второй страницы должен иметь свой вид курсора.
7. На форму поместить во всю клиентскую область компонент TTabbedNoteBook, содержащий две страницы. На первой странице поместить компоненты: два TstringGrid, два TbitBtn и Tlabel. Нужно создать подобие кассового аппарата. В первом TstringGrid в первой фиксированной строке ввести надписи «Товар» и «Цена». В этих двух столбцах нужно описать товары и их цены. В Tedit можно будет вводить количество выбранного товара. Кнопка TbitBtn означает суммирование. При этом во второй TstringGrid вносится наименование товара, цена, количество и полная стоимость данного товара. В Tlabel выводится полная стоимость покупок. Вторая кнопка TbitBtn должна очищать TstringGrid2 и Tlabel. На второй странице компонента TtabbedNoteBook поместить компоненты Ttimer, Tgauge и TcolorDrid. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой горизонтально заполняемый прямоугольник с текстом процентного заполнения этого прямоугольника. С помощью компонента TcolorGrid предусмотреть изменение цвета фона и заполнения в компоненте Tgauge. Все визуальные компоненты должны иметь всплывающие подсказки (Hint).
8. На форму поместить во всю клиентскую область компонент TTabbedNoteBook, содержащий две страницы. На первой странице поместить компоненты TopenDialog, TsaveDialog, TlistBox и два TbitBtn. При нажатии на первую кнопку TbitBtn должен открываться диалог TopenDialog и выбранные имена файлов помещаются в TlistBox. Выбрав нужные файлы в TlistBox и нажав на вторую кнопку TbitBtn, следует открыть диалог TsaveDialog и сохранить файлы в выбранном каталоге. На второй странице компонента TtabbedNoteBook поместить компоненты Ttimer, Tgauge и TcolorDrid. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой вертикально заполняемый прямоугольник с текстом процентного заполнения этого прямоугольника. С помощью компонента TcolorGrid предусмотреть изменение цвета фона и заполнения в компоненте Tgauge. Все визуальные компоненты должны иметь всплывающие подсказки (Hint).

9. На форму поместить во всю клиентскую область компонент TPageControl, содержащий две страницы. На первой странице поместить компоненты TdrawGrid, TopenPictureDialog и TbitBtn. При нажатии на кнопку TbitBtn должен вызываться диалог TopenPictureDialog и после выбора картинки должны помещаться в ячейки компонента TdrawGrid. На второй странице компонента TtabbedNoteBook поместить компоненты Ttimer, Tgauge и TcolorDrid. Компонент Ttimer должен управлять изменением движка в компоненте Tgauge, который должен представлять собой сектор окружности с текстом процентного заполнения этой области. С помощью компонента TcolorGrid предусмотреть изменение цвета фона и заполнения в компоненте Tgauge. Все визуальные компоненты должны иметь свой вид курсора.
10. Внизу формы поместить компонент Tpanel. В оставшуюся клиентскую область формы поместить компонент TPageControl, содержащий две страницы. На первой странице поместить компоненты, реализующие простейший калькулятор с памятью и выполняющий следующие действия: +, -, *, /, √. На второй странице поместить компоненты Ttimer, Tgauge и кнопку BitBtn со своей пиктограммой и словом «Старт». Эта кнопка должна запускать Ttimer, который должен управлять текущим состоянием Tgauge в виде спидометра. TTemer также должен обеспечивать случайный выбор цветов для стрелки спидометра и фона. Все визуальные компоненты должны иметь составную подсказку, вторую часть которой нужно выводить в Tpanel.
11. На форму поместить во всю клиентскую область компонент TPageControl, содержащий две страницы. На первой странице поместить компоненты реализующие простейший калькулятор с памятью и выполняющий следующие действия: +, -, *, /, sin, cos, arctan. На второй странице разместить компоненты Ttimer, TProgressBar, Tedit и TupDown. Ttimer должен управлять заполнением TprogressBar, а Tedit и TupDown должны задавать скорость этого заполнения. Все компоненты должны иметь свои подсказки и вид курсора.
12. На форму поместить во всю клиентскую область компонент TPageControl, содержащий две страницы. На первой странице поместить компоненты Tchart и BitBtn. При нажатии на кнопку BitBtn в Tchart нужно нарисовать две кривые функций Sin и Cos для $a<x<2\pi+a$. При каждом нажатии эту кнопку «a» увеличивать на 0.5. На вторую страницу поместить компоненты TtrackBar, Ttimer и TspinEdit. Ttimer должен управлять перемещением движка в TtrackBar, а TspinEdit управлять скоростью этого перемещения. Все компоненты должны иметь подсказки.
13. На форму поместить во всю клиентскую область компонент TPageControl, содержащий две страницы. На первой странице поместить компонент TVTChart со страницы ActiveX. С его помощью построить две кривые для функций Sin и Cos и $0<x<2\pi$. На вторую страницу поместить компоненты TtrackBar, Ttimer и TspinEdit. Ttimer должен управлять перемещением движка в TtrackBar, а TspinEdit управлять скоростью этого перемещения. Все компоненты должны иметь подсказки.

14. На форму поместить во всю клиентскую область компонент TPageControl, содержащий две страницы. На первой странице поместить компонент Tchartfx со страницы ActiveX. С его помощью построить две кривые для функций Sin и Cos и $0<x<2\pi$. На вторую страницу поместить компоненты TprogressBar, Ttimer и TspinEdit. Ttimer должен управлять перемещением движка в TprogressBar, а TspinEdit управлять скоростью этого перемещения. Все компоненты должны иметь подсказки.
15. На форму поместить во всю клиентскую область компонент TPageControl, содержащий две страницы. На первой странице поместить компоненты, реализующие простейший калькулятор с памятью и выполняющий следующие действия: +, -, sin, cos, arctan. Необходимо предусмотреть возможность задания аргументов тригонометрических функций в радианах и градусах, а целые числа возможно задавать как в десятичной форме представления, так и шестнадцатеричной. На второй странице поместить компоненты, реализующие часы в числовом представлении с возможностью изменения текущего времени и дата. Все компоненты должны иметь свой собственный вид курсора.

ЛАБОРАТОРНАЯ РАБОТА 3. ГРАФИКА В DELPHI

Цель лабораторной работы: научиться работать с графическими объектами.

3.1 Класс Tcanvas

Основу графики в Delphi представляет класс Tcanvas. Это холст (контекст GDI в Windows) с набором инструментов для рисования. Основные свойства холста:

Property Pen:Tpen; - карандаш,

Property Brush:Tbrush; - кисть,

Property Font:Tfont; - шрифт,

Property PenPos:Tpoint; - текущая позиция карандаша в пикселях относительно левого верхнего угла канвы,

Property Pixels[x,y:Integer]:Tcolor.

Property CopyMode:TcopyMode; Это свойство определяет, как графический рисунок копируется в канву. Оно используется при вызове метода CopyRect и при копировании объектов TbitMap.

Возможные значения этого свойства приведены ниже.

Таблица 2

cmBlackness	Заполняет область рисования черным цветом
cmDest	Заполняет область рисования цветом фона
cmMergeCopy	Объединяет изображение на канве и копируемое изображение операцией AND
cmMergePaint	Объединяет изображение на канве и копируемое изображение операцией OR
cmNotSrcCopy	Копирует на канву инверсное изображение источника
cmNotSrcErase	Объединяет изображение на канве и копируемое изображение операцией OR и инвертирует полученное
cmPatCopy	Копирует шаблон источника
cmPatInvert	Комбинирует шаблон источника с изображением на канве с помощью операции XOR
cmPatPaint	Комбинирует инверсное изображение источника с исходным шаблоном, используя операцию OR. Смешивает результат этого действия с изображением на холсте, используя логическую операцию OR
cmSrcAnd	Объединяет изображение источника и канвы с помощью операции AND
cmSrcCopy	Копирует изображение источника на канву
cmSrcErase	Инвертирует изображение на канве и объединяет результат с изображением источника операцией AND
cmSrcInvert	Объединяет изображение на канве и источнике операцией XOR. Отметим, что повторное объединение восстанавливает первоначальное изображение на канве. Это используется в играх, при движении какого-то объекта по фону
cmSrcPaint	Объединяет изображение на канве и источнике операцией OR
CmWhiteness	Заполняет область рисования белым цветом

Канва не является компонентом, но во многих компонентах является свойством. С помощью свойства Pixels все пиксели канвы представляются в виде двумерного массива точек. Изменяя цвет пикселей, можно прорисовывать изображение по отдельным точкам.

Методы класса Таблица 3

Procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);	Чертит дугу эллипса в охватывающем прямоугольнике (X1, Y1)- (X2, У2). Начало дуги лежит на пересечении эллипса и луча, проведенного из его центра в точку (X3, Y3), а конец - на пересечении с лучом из центра в точку (X4, У4). Дуга чертится против часовой стрелки (рис.3.1)
Procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer) ;	Чертит сегмент эллипса в охватывающем прямоугольнике (XI, VI) - (X2, У2). Начало дуги сегмента лежит на пересечении эллипса и луча, проведенного из его центра в точку (X3, Y3), а конец - на пересечении с лучом из центра в точку (X4, У4), Дуга сегмента чертится против часовой стрелки, а начальная и конечная точки дуги соединяются прямой (рис. 3.2)
Procedure CopyRect(Dest: Trect; Canvas: Tcanvas; Source: Trect);	Копирует изображение Source канвы Canvas в участок Dest текущей канвы. При этом свойство CopyMode определяет различные эффекты копирования
Procedure Draw(X, Y: Integer; Graphic: TGraphic);	Осуществляет вывод на канву графического объекта Graphic так, чтобы левый верхний угол объекта расположился в точке (X, Y)
Procedure E11ipse(X1, Y1, X2, Y2: Integer);	Чертит эллипс в охватывающем прямоугольнике (X1, Y1) -(X2, Y2). Заполняет внутреннее пространство эллипса текущей кистью
Procedure FillRect(const Rect: Trect) ;	Заполняет текущей кистью прямоугольную область Rect, включая ее левую и верхнюю границы, но не затрагивая правую и нижнюю границы
Procedure FloodFill(X, Y: Integer; Co1or: TCo1or; FillStyle: TFillStyle) ;	Производит заливку канвы текущей кистью. Заливка начинается с точки (X, У) и распространяется во все стороны от нее. Если FillStyle=fsSurface, заливка распространяется на все соседние точки с цветом Co1or. Если FillStyle=fsBorder, наоборот, заливка прекращается на точках с этим цветом

Продолжение табл. 3

Procedure FrameRect(const Rect:TRect) ;	Очерчивает границы прямоугольника Rect текущей кистью толщиной в 1 пиксель без заполнения внутренней части прямоугольника
Procedure LineTo (X, У: Integer);	Чертит линию от текущего положения пера до точки (X. У)
Procedure MoveTo(X, У: Integer) ;	Перемещает карандаш в положение (X, У) без вычерчивания линий
Procedure Pie(X1, У1, X2, У2, X3, У3, X4, У4: Integer);	Рисует сектор эллипса в охватывающем прямоугольнике (X1, У1) - (X2, у2). Начало дуги лежит на пересечении эллипса и луча, проведенного из его центра в точку (X3, У3), а конец - на пересечении с лучом из центра в точку (X4, У4). Дуга чертится против часовой стрелки. Начало и конец дуги соединяются прямыми с ее центром (см. рис. 3.3,в)
Procedure Po1ygon (Points: аггау of TPoint) ;	Вычерчивает карандашом многоугольник по точкам, заданным в массиве Points. Конечная точка соединяется с начальной и многоугольник заполняется кистью. Для вычерчивания без заполнения используйте метод Po1yline.
Procedure Po1y1ine (Points: array of TPoint) ;	Вычерчивает карандашом ломаную прямую по точкам, заданным в массиве Poimts.
Procedure Rectangle (X1, У1, X2, У2: Integer);	Вычерчивает и заполняет прямоугольник (X1, У1) - (X2, У2). Для вычерчивания без заполнения используйте FrameRect или Po1yLine.
Procedure ReFresh;	Устанавливает в канве умалчиваемые шрифт, карандаш и кисть.
Procedure RoundRect (X1, У1, X2, У2, X3, У3: Integer) ;	Вычерчивает и заполняет прямоугольник (XI, У1) - (X2, У2) со скругленными углами. Прямоугольник (X1, Y1) - (X3, У3) определяет дугу эллипса для округления углов (рис.13.3,г)
Procedure StretchDraw (const Rect: TRect; Graphic: TGraphic) ;	Вычерчивает и при необходимости масштабирует графический объект Graphic так, чтобы он полностью занял прямоугольник Rect
Procedure TextOut (X, У: Integer; const Text: String) ;	Выводит текстовую строку Text так, чтобы левый верхний угол прямоугольника, охватывающего текст, располагался в точке (X, У)

Procedure TextRect (Rect: TRect; X, У: Integer; const Text: String) ;	Выводит строку Text так, чтобы левый верхний угол прямоугольника, охватывающего текст, располагался в точке (X, У). Если при этом какая-либо часть надписи выходит из границ прямоугольника Rect, она отсекается и не будет видна

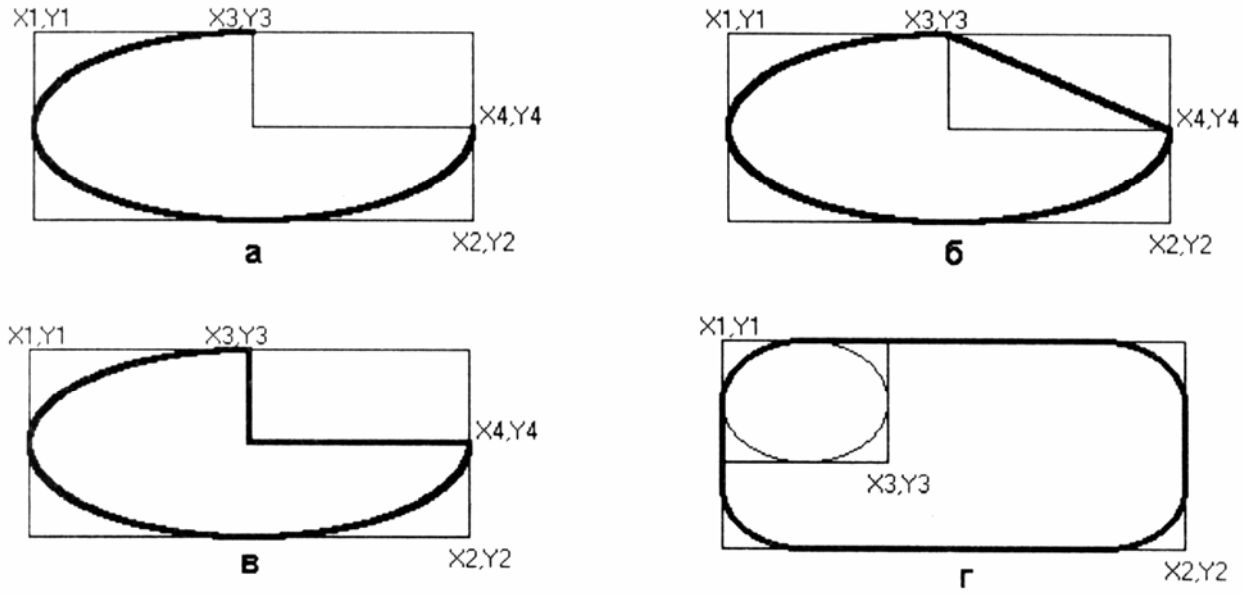


Рис. 3.1. Параметры обращения: а) к методу Arc; б) к методу Chord; в) к методу Pie; г) к методу RoundRect

3.2. Классы TGraphic и TPicture

Важное место в графическом инструментарии Delphi занимают классы TGraphic и Tpicture.

Tgraphic - это абстрактный класс, инкапсулирующий общие свойства и методы трех своих потомков: пиктограммы (TIcon), метафайла (TmetaFile) и растрового изображения (TBitMap). Общей особенностью потомков TGraphic является то, что обычно они сохраняются в файлах определенного формата. Пиктограммы представляют собой небольшие растровые изображения, снабженные специальными средствами, регулирующими их прозрачность. Для файлов пиктограмм обычно используется расширение ICO. Метафайл - это изображение, построенное на графическом устройстве с помощью специальных команд, которые сохраняются в файле с расширением WMF или EMF. Растровые изображения - это произвольные графические изображения в файлах со стандартным расширением BMP.

Свойства класса TGraphic Таблица 4

Property Empty: Boo1ean;	Содержит *Tгue,* если с объектом не связано графическое изображение.

Окончание табл. 4

Property Height: Tnteger;	Содержит высоту изображения в пикселях
Property Modefied: Boo1ean;	Содержит *Тгие,* если графический объект изменялся
Property Palette: HPALETTE;	Содержит цветовую палитру графического объекта
Property PaletteModified: Boolean;	Содержит *True,* если менялась цветовая палитра графического объекта
Property Transparent: Boolean;	Содержит *True,* если объект прозрачен для фона, на котором он изображен
Property Width: Integer;	Содержит ширину изображения в пикселях

Методы класса TGraphic Таблица 5

Procedure LoadFromClipboardFormat (AFormat: Word; AData: THandle; Apalette: HPALETTE) ;	Ищет в буфере межпрограммного обмена *Clipboard* зарегистрированный формат *AFormat* и, если формат найден, загружает, из буфера изображение *AData* и его палитру *APalette*
Procedure LoadFromFile(const FileName: String) ;	Загружает изображение из файла *FileName*
Procedure LoadFromStream (Stream: TStream) ;	Загружает изображение из потока данных *Stream*
Procedure SaveToClipboardFormat (var AFormat: Word; var AData: THandle; var APalette: HPALETTE);	Помещает графическое изображение *AData* и его цветовую палитру *APalette* в буфер меж-программного обмена в формате *Aformat*
Procedure SaveToFile(const FileName:String) ;	Сохраняет изображение в файле *FileName*
Procedure SaveToStream(Stream: TStream);	Сохраняет изображение в потоке *Stream*

Полнофункциональный класс *TPicture* инкапсулирует в себе все необходимое для работы с готовыми графическими изображениями - пиктограммой, растром или метафайлом. Его свойство *Graphic* может содержать объект любого из этих типов, обеспечивая нужный полиморфизм методов класса.

Свойства класса TPicture Таблица 6

property Bitmap: TBitmap;	Интерпретирует графический объект как растровое изображение

Окончание табл. 6

Property Graphic: TGraphic;	Содержит графический объект
Property Height: Integer;	Содержит высоту изображения в пикселях
Property Icon: TIcon;	Интерпретирует графический объект как пиктограмму
Property Metafile: TMetafile;	Интерпретирует графический объект как метафайл
Property Width: Integer;	Содержит ширину изображения в пикселях

Методы класса *TPicture* Таблица 7

Procedure Assign(Source: TPersistent) ;	Связывает собственный графический объект *Graphic* с объектом *Source*
Procedure LoadFromClipboardFormat (AFormat: Word; AData: THandle; APalette: HPALETTE);	Ищет в буфере межпрограммного обмена *Clipboard* зарегистрированный формат *AFormat* и, если формат найден, загружает из буфера изображение *AData* и его палитру *Apalette*
Procedure LoadFromFile(const FileName: String) ;	*Загружает* изображение из файла *FileName*
class Procedure RegisterClipboard-Format(AFormat: Word; AGraphicClass: TGraphicClass) ;	Используется для регистрации в *Clipboard* нового формата изображения
class Procedure RegisterFileFormat (const AExtension, ADescription: String; AGraphicClass: TGraphicClass);	Используется для регистрации нового файлового формата
class Procedure RegisterFileFormat-Res(const AExtension: String; ADescriptionResID: Integer; AGraphicClass: TGraphicClass);	Используется для регистрации нового формата ресурсного файла
Procedure SaveToClipboardFormat (var AFormat: Word; var AData: THandle; var APalette: HPALETTE) ;	Помещает графическое изображение *AData* и его цветовую палитру *APalette* в буфер межпрограммного обмена в формате *AFormat*

Procedure SaveToFile(const FileName: String) ;	Сохраняет изображение в файле *FileName*
class Function SupportsClipboard-Format(AFormat: Word): Boolean;	Возвращает *True,* если формат *AFormat* зарегистрирован в буфере межпрограммного обмена *Clipbord*
class Procedure UnregisterGraphic-Class(AClass: TGraphicClass);	Делает недоступными любые графические объекты класса *AClass*

3.3 Классы TFont, TPen и TBrush

Класс TFont определяет объект-шрифт для любого графического устройства (экран, принтер и т.д.).

Свойства класса Таблица 8

Property Charset: TFontCharSet;	Набор символов. Для русскоязычных программ это свойство обычно имеет значение *DEFAULTCHARSET* или *RUSSIAN CHARSET*. Используйте значение *OEMCHARSET* для отображения текста *MS-DOS* (альтернативная кодировка)
Property Color: TColor;	Цвет шрифта
Property FontAdapter: IChangeNotifier;	Поставляет информацию о шрифте в компоненты *ActveX*
Property Handle: hFont;	Дескриптор шрифта. Используется при непосредственном обращении к API-функциям *Windows*
Property Height: Integer;	Высота шрифта в пикселях экрана
Property Name: TFontName;	Имя шрифта. По умолчанию имеет значение *MS Sans Serif*
Property Pitch: TFontPitch;	Определяет способ расположения букв в тексте: значение *fpFixed* задает моноширинный текст, при котором каждая буква имеет одинаковую ширину; значение *fpVariabel* определяет пропорциональный текст, при котором ширина буквы зависит от ее начертания; *fpDectault* определяет ширину, принятую для текущего шрифта

Окончание табл. 8

Property PixelPerInch: Integer;	Определяет количество пикселей экрана на один дюйм реальной длины. Это свойство не следует изменять, т.к. оно используется системой для обеспечения соответствия экранного шрифта шрифту принтера
Property Size: Integer;	Высота шрифта в пунктах (1/72 дюйма). Изменение этого свойства автоматически изменяет свойство *Height* и наоборот
Property Style: TFont-Styles;	Стиль шрифта. Может принимать значение как комбинацию следующих *признаков: fsBold* (жирный), *fsItalic (Курсив)*,*fsUnderline (подчеркнутый), fsStrikeOut* (перечеркнутый)

Класс TPen определяет объект перо для рисования линий.

Свойства класса Таблица 9

Property Color: TColor;	Цвет вычерчиваемых пером линий
Property Handle : Integer;	Дескриптор пера. Используется при непосредственном обращении к API-функциям *Windows*
Property Mode: TPenMode ;	Определяет способ взаимодействия линий с фоном (см. ниже)
Property Style: TPenStyle;	Определяет стиль линий. Учитывается только для толщины линий 1 пиксель. Для толстых линий стиль всегда *psSolid* (сплошная)
property Width: Integer;	Толщина линий в пикселях экрана

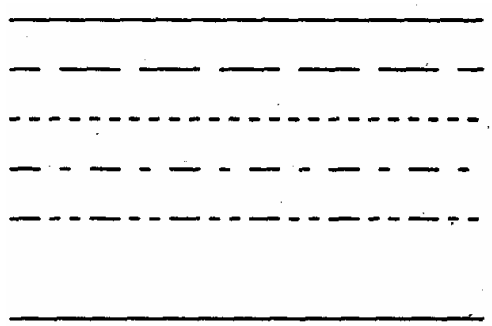
Значения свойства *Mode* Таблица 10

pmBlack	Линии всегда черные. Свойства *Color* и *Style* игнорируются
pmWhite	Линии всегда белые. Свойства *Color* и *Style* игнорируются
pmNop	Цвет фона не меняется (линии не видны)
pmNot	Инверсия цвета фона. Свойства *Color* и *Style* игнорируются
pmCopy	Цвет линий определяется свойством *Color* пера
pmMotCopy	Инверсия цвета пера. Свойство *Style* игнорируется
pfflMergePenNot	Комбинация цвета пера и инверсионного цвета фона
pmMaskPenNot	Комбинация общих цветов для пера и инверсионного цвета фона. Свойство *Style* игнорируется
pmMergeNotPen	Комбинация инверсионного цвета пера и фона

Окончание табл. 10

pmMaskNotPen	Комбинация общих цветов для инверсионного цвета пера и фона. Свойство *Style* игнорируется
pmMerge	Комбинация цветов пера и фона
pmNotMerge	Инверсия цветов пера и фона. Свойство *Style* игнорируется
pmMask	Общие цвета пера и фона
pieNotMask	Инверсия общих цветов пера и фона
pmXor	Объединение цветов пера и фона операцией *XOP*
pmMotXor	Инверсия объединения цветов пера и фона операцией *XOR*

Стили линий показаны ниже.

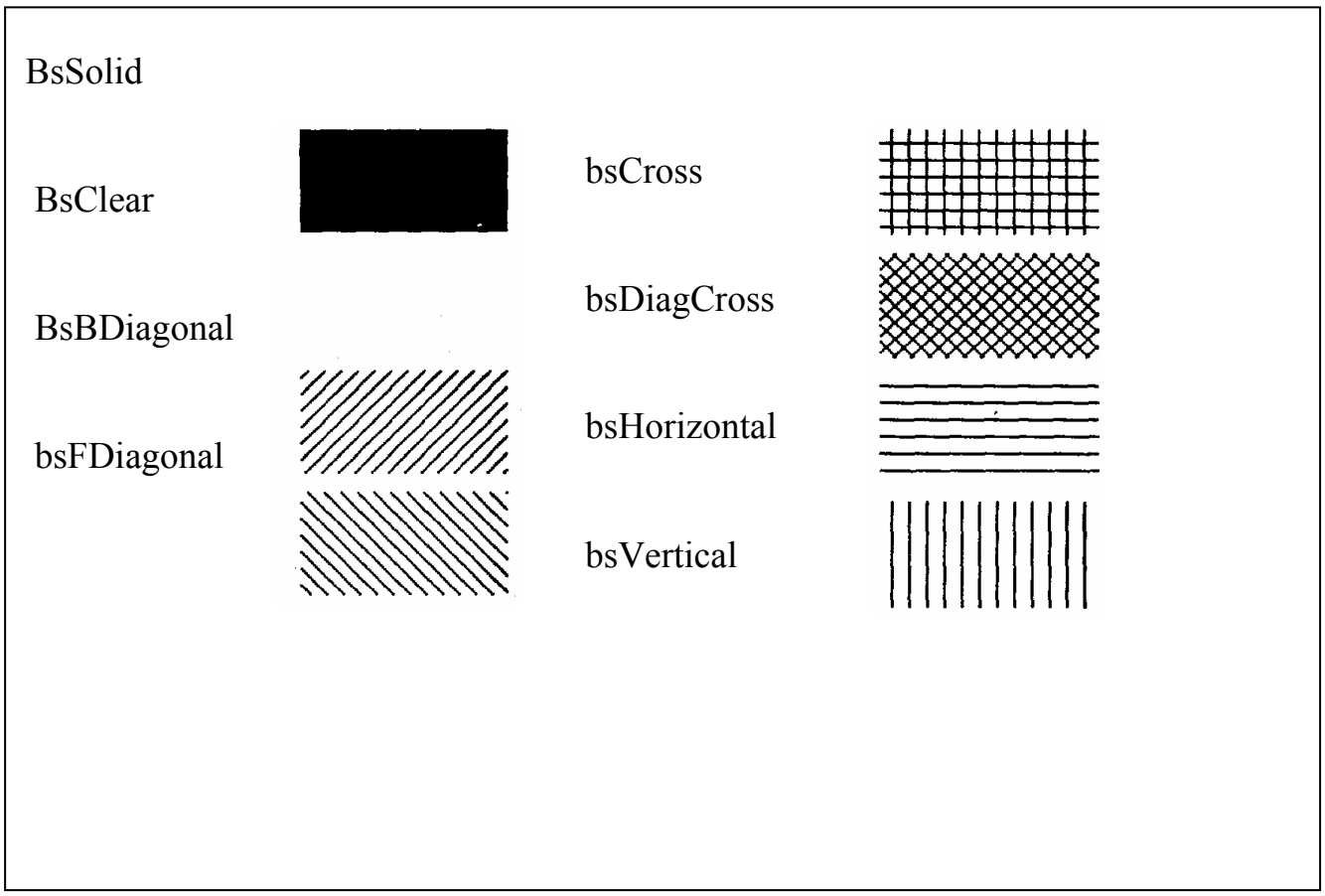


psSolid
psDash
psDot
psDashDot
psDashDotDot
psClear
psInsideFrame

Класс TBrush служит для описания параметров кисти для заполнения внутреннего пространства замкнутых фигур.

Свойства класса Таблица 11

Property Bitmap: TBitmap;	Содержит растровое изображение, которое будет использоваться кистью для заполнения. Если это свойство определено, свойства *Color* и *Style* игнорируются
Property Color: TColor;	Цвет кисти
Property Handle: Integer;	Дескриптор кисти. Используется при непосредственном обращении к API-функциям *Windows*.
Property Style:TBrushStyle;	Стиль кисти



Стили кисти:

Задания

1. Разработать программу, реализующую игру «Бега лошадей по кругу ипподрома». Предусмотреть возможность устанавливать ставки на лошадей и расчета выигрыша. Скорость движения лошадей должна задаваться случайным образом функцией Random.
2. Разработать программу, реализующую игру «Бега лошадей по прямой». Предусмотреть возможность устанавливать ставки на лошадей и расчета выигрыша. Скорость движения лошадей должна задаваться случайным образом функцией Random.
3. Разработать программу игры в крестики – нолики. В основу положить компонент DrawGrid.
4. Разработать программу игры «Минер» по подобию такой же игры в системе Windows. Начальная расстановка мин должна выполняться случайным образом. В основу положить компонент DrawGrid.
5. Разработать программу игры «Стрельба из подводной лодки по кораблю», используя вид из перископа. На заднем плане должен периодически проплывать кораблик с постоянной поперечной скоростью. С помощью клавиш влево-вправо следует менять вид в перископе. Клавиша «Ввод» должна запускать торпеду. В перископе должна отображаться траектория движения торпеды с уменьшением скорости движения при приближении к кораблю. Попадание должно сопровождаться видимым взрывом и исчезновением корабля.

6. Разработать программы игры «Бомбометание с самолета по наземной цели». С летящего с постоянной скоростью самолета клавишей «Ввод» производить бомбометание. Траектория движения бомбы должна соответствовать физическим законам падения тел на землю. Попадание в цель должно сопровождаться видимым взрывом и исчезновением цели. Самолет должен периодически вылетать из-за края канвы компонента рисования.
7. Разработать программу игры «Морской бой». Программа должна случайным образом на сетке 10х10 расставлять корабли: один четырехклеточный, два трехклеточных, три двухклеточных и четыре одноклеточных. Они не могут изгибаться и соприкасаться друг с другом. Игрок выбирает определенный квадрат и как бы стреляет в него. Программа должна сообщать, попал ли игрок в корабль или нет. Она также должна отображать все старые выстрелы и показывать ячейки, куда уже не имеет смысла стрелять. Аналогично строится и вторая таблица, где игрок располагает свои корабли, по которым уже случайным образом стреляет программа. Выигрывает тот, кто быстрей потопит корабли неприятеля. Предусмотреть в конце игры показ расположения кораблей программы. Для таблиц использовать компоненты TdrawGrid.
8. Разработать программы игры «Стрельба из пушки». Пушка должна стрелять через гору по какой-то цели. Траектория полета снаряда должна подчиняться законам физики. Игрок может управлять углом подъема ствола относительно горизонта и начальной скоростью снаряда в дискретных величинах (определяется типом снаряда). При попадании должен происходить видимый взрыв и исчезновение цели.
9. Разработать программу показа в форме текущего времени в виде обычных стрелочных часов со стрелками часов, минут и секунд.
10. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента TpaintBox произвольные кривые с помощью мыши. Предусмотреть возможность:
    а) изменения толщины кривых,
    б) изменение цвета кривых,
    в) сохранение рисунка в графическом файле.
11. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента TpaintBox ломанные линии с помощью нажатия на клавиши мыши. Предусмотреть возможность:
    а) изменения толщины линий,
    б) изменение цвета линий,
    в) сохранение рисунка в графическом файле.
12. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента TpaintBox с помощью мыши прямоугольники. Предусмотреть возможность:
    а) изменения толщины линий,

б) изменение цвета линий,
в) заливку областей текущей кистью,
г) изменение цвета кисти,
д) сохранение рисунка в графическом файле.

13. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента TpaintBox с помощью мыши эллипсы. Предусмотреть возможность:
    а) изменения толщины линий,
    б) изменение цвета линий,
    в) заливку областей текущей кистью,
    г) изменение цвета кисти,
    д) сохранение рисунка в графическом файле.
14. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен рисовать в канве компонента TpaintBox любой текст в указанном мышкой месте. Предусмотреть возможность:
    а) изменения типа, размера и цвета шрифта,
    б) сохранение рисунка в графическом файле.
15. Разработать программу простейшего графического редактора (аналога программы Paint системы Windows). Он должен помещать в канву компонента TpaintBox из графического файла произвольный рисунок и обеспечивать возможность:
    а) стирания произвольной области рисунка,
    б) изменение размеров стирки,
    в) сохранение рисунка в графическом файле.

ЛАБОРАТОРНАЯ РАБОТА 4. РАБОТА С БАЗАМИ ДАННЫХ

Цель лабораторной работы: научиться создавать простейшие программы для работы с базами данных.

4.1 Основные определения

База данных - это набор взаимосвязанных таблиц, которые могут храниться как в отдельных файлах (системы управления базами данных – Dbase, Paradox), так и в одном файле (система управления базами данных Access).

Таблица - это набор логически связанных полей (столбцов), количество которых постоянно, и записей (строк), количество которых изменяется.

Первыми в таблице идут поля *ключа записи*. Ключ определяет уникальность одной записи (строки). Сроки в таблицах реляционных баз данных образуют множество, они не имеют последовательных номеров и для их идентификации используется ключ.

Индексные поля определяют порядок сортировки строк таблицы. Индекс может состоять из одного поля, нескольких полей или быть выражением на основе данных полей таблицы. Индексы позволяют резко ускорить процесс поиска нужной записи таблицы путем использования бинарного поиска. Следует иметь в виду, что индексы требуют дополнительного места на диске, но все поиски информации в таблицах проводятся только с использованием индексов или ключей.

Домен – это диапазон или набор допустимых значений какого-либо поля.

Сессия – это в какой-то мере канал связи с базой данных. Приложение может иметь несколько изолированных сессий для доступа к данным. Сессия является владельцем курсоров и блокировок таблиц.

Курсор в контексте баз данных представляет собой область памяти, в которой хранится текущий набор строк таблицы и информация о закладках (bookmark). Все операции по изменению данных таблицы происходят с помощью курсора. Курсор позволяет существенно ускорить работу с таблицами, т.к. из таблицы читается не одна запись, а сразу 10-1000, все зависит от объема буферной памяти. Это существенно сокращает число обращений к диску.

Запрос – это оператор на языке SQL (Structured Query Language) - структурированных запросов. Результатом запроса является таблица.

Транзакция – это система взаимосвязанных действий по изменению базы данных. Эта группа действий или должна быть выполнена целиком или все действия должны быть отменены. Это так называемый откат назад.

Хранимая процедура – это набор SQL запросов, хранимых на сервере, и которые можно вызывать просто по имени. Клиент это тот, кто задает вопросы, а сервер это тот, кто на них отвечает.

4.2 Механизм взаимодействия программы на Delphi с базами данных

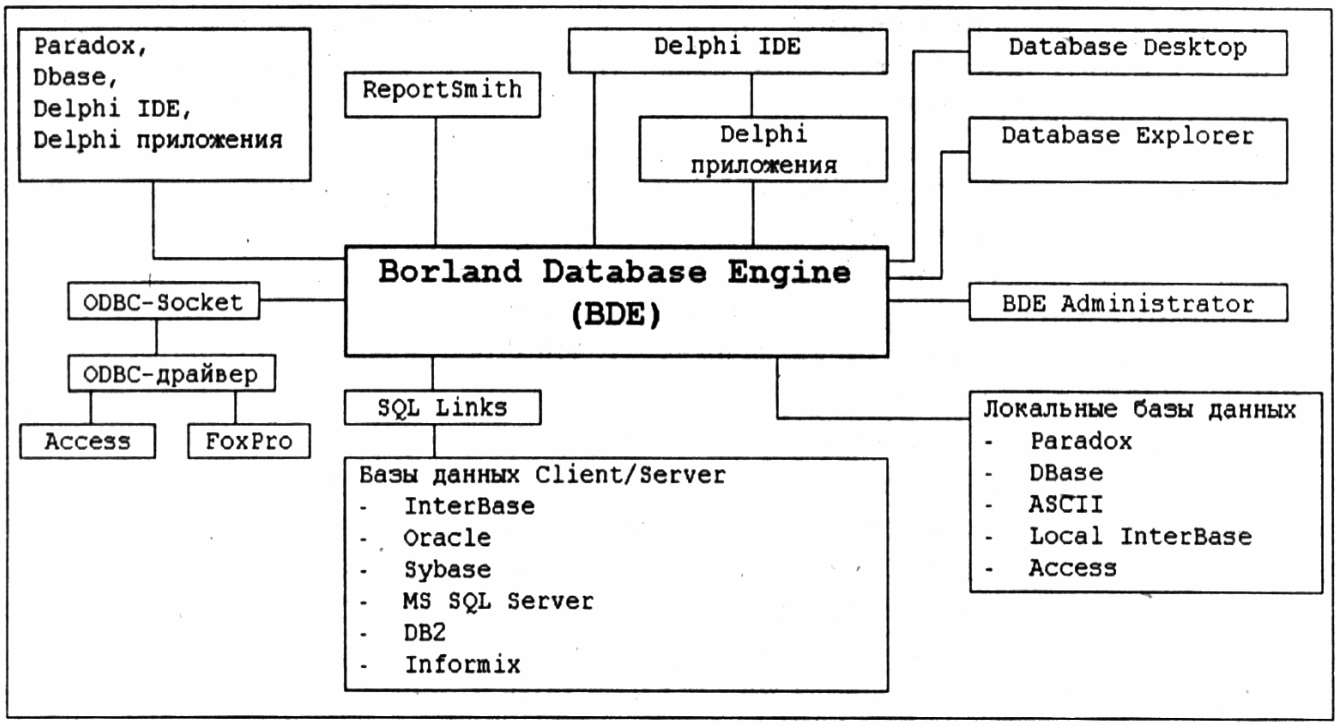


Рис.4.1, Функции BDE

Как видно из рисунка основу связи любого приложения с базой данных составляет BDE- процессор баз данных фирмы Borland (Inprise). BDE представляет собой набор DLL библиотек, драйверов, файлов конфигурации и дополнительных программ DataBase DeskTop, DataBase Explorer и BDE Administrator. Следует иметь в виду, что при поставке готовой программы написанной на Delphi для работы с базами данных заказчику придется поставлять и процессор BDE. Это легко сделать с помощью программы InstallShield Express, которая входит в состав программ, поставляемых вместе с Delphi. Однако, начиная с версии Delphi 5.0, в палитре компонентов есть страничка ADO, компоненты которой используют напрямую процессор баз данных фирмы MicroSoft – ODBC (Open Data Base Connection), который входит как составная часть в систему Windows. В данной работе мы не будем затрагивать создание клиент – серверных приложений и использование языка структурированных запросов SQL.

На рис.4.2 приведена схема взаимодействия компонентов для доступа, визуализации и управления данными. Непосредственно с базой данных связаны компоненты Ttable или Tquery. Компоненты TdataSource являются промежуточным буфером между ними и визуальными компонентами, которые выводят информацию на экран и позволяют интерактивно изменить ее.

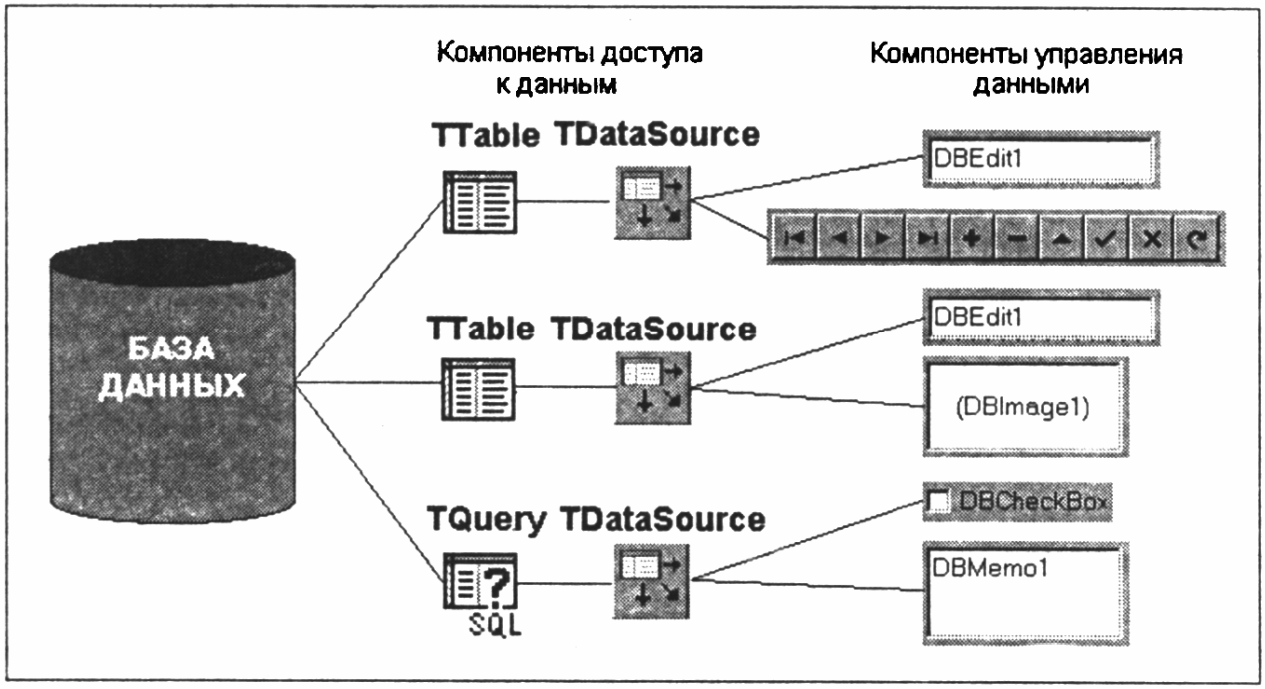
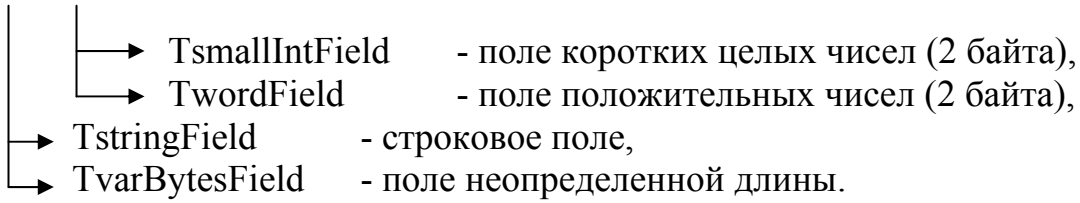


Рис.4.2 Взаимодействие компонентов доступа и управления данными

Иерархия объектов доступа к базам данных приведена ниже.

Tcomponent - компонент,
- Tsession - сессия,
- Tdatabase - база данных,
- TdataSource - источник данных,
- TbatchMove - перемащение данных
- TdataSet - абстрактный класс для реляционного набора данных,
  - TDBDateSet - класс для работы с базами данных через BDE,
    - Ttable - таблица,
    - Tquery - запрос,
    - TStoredProc - хранимая процедура,
- Tfield - поле
  - TblobField - двоичный массив данных,
    - TgraphicField - картинка,
    - TmemoField - многострочный текст,
  - TbooleanField - логическое поле,
  - TbytesField - поле с фиксированным числом байт,
  - TdateTimeField - поле даты и времени,
    - TdateField - поле даты,
    - TtimeField - поле времени,
  - TnumericField - числовые поля,
    - TfloatField - поле вещественного числа,
      - TBCDField - поле числа повышенной точности,
      - TcurrencyField – поле денежных значений,
  - TintegerField - поле целых чисел (4 байта),



Основу связи программы с базой данных составляет компонент TTable. Использование API BDE возможно через указатели на курсор и базу данных. Записи в наборе данных можно селектировать и ограничивать при помощи свойств Filter, Filtered и Filter0ptions, создающих фильтр, ограничивающий набор данных по значениям данных. Методы SetRangeStart, SetRangeEnd, SetRange, ApplyRange, EditRangeStart, EditRangeEnd создают специальный диапазон включаемых в набор данных записей, отбор в диапазон проводится по задаваемым граничным значениям любых полей набора данных. Поиск нужной записи можно осуществлять методами Lookup или Locate (достаточно просто и не очень быстро) или, используя существующие в таблице базы данных индексы, методом FindKey (сложнее, но очень быстро). От предков компонент унаследовал инструменты для работы с закладками (методы GetBookmark, FreeBootanark, GotoBookmark). Работа с полями осуществляется целой группой свойств и методов, среди которых особое место занимает свойство Fields, представляющее собой индексированный список всех полей набора данных. Использование индексов обеспечено свойствами indexName, indexFields, IndexFieldNames, IndexFiles. Свойства MasterSource, MasterField, IndexName дают возможность установить отношение "один ко многим" с другой таблицей. Механизм навигации по набору данных унаследован от класса TDataSet. Очень полезны в практическом использовании методы и свойства для работы с буфером вносимых изменений (свойства CachedUpdates, UpdateRecordTypes, методы ApplyUpdates, CancelUpdates, ConmitUpdate, RevertRecord). От класса TDataSet унаследован обширный набор методов - обработчиков событий, позволяющий решать практически любые задачи по управлению набором данных. Область применения компонента не зависит от типа приложения (одноуровневое или многоуровневое). В данном практикуме не приводятся описания всех классов объектов, используемых при работе с базами данных, это выходит за рамки возможностей методического пособия. В справочной системе Delphi можно найти подробное описание любого из классов. Остановимся хотя бы на основном классе TTable, свойства и методы которого перечислены в табл. 12.

4.3 Класс TTable

Свойства и методы класса TTable Таблица 12

Объявление	Тип	Описание
Свойства		
Property Defaultlndex: Boolean;	Pb	Управляет сортировкой данных. При значении True записи упорядочиваются по первичному ключу. При значении False упорядочивание не производится
Property Exclusive: Boolean;	Pb	Ограничивает доступ к таблице. При значении True с таблицей может работать только одно приложение. Это свойство важно при работе с данными в локальной сети
Property Exists: Boolean;	Pu, Ro	Значение True говорит о том, что связанная с компонентом таблица базы данных существует
Property IndexDefs: TIndexDefs;	Pb	Содержит информацию об индексах таблицы
Property IndexFieldCount: Integer;	Pu, Ro	Возвращает число полей в текущем индексе таблицы
Property IndexFieldNames: string;	Pb	Разделенный запятыми список названий полей, составляющих текущий индекс
Property IndexFields: [Index: Integer]: TField;	Pu	Индексированный список полей текущего индекса
Property IndexFiles: Tstrings;	Pb	Список индексных файлов для таблиц dBASE
Property IndexName: string;	Pb	Определяет вторичный индекс для таблицы
Property KeyExclusive: Boolean;	Pu	Управляет границами диапазона, задаваемого методом SetRange. При значении True крайние записи в диапазон не включаются
Property KeyFieldCount: Integer;	Pu	Содержит число полей ключа, используемых при поиске. При значении 0 используется только первое поле, при значении 1 используются два первых поля и т. д. По умолчанию устанавливается полное число полей ключа
Property MasterFields: string;	Pb	Список имен полей главной таблицы, разделенных запятой, используемых при создании отношения "один ко многим"

Property MasterSource: TDataSource;	Pb	Содержит имя компонента TDataSource, связанного с набором данных, который является главным в отношении "один ко многим"
Property Readonly: Boolean;	Pb	Включает и отключает режим "только для чтения". В некоторых случаях набор данных можно открыть только в этом режиме
Property StoreDefs: Boolean;	Pb	При значении True все сведения об индексах и структуре таблицы хранятся вместе с формой или модулем данных. В этом случае при создании набора данных одновременно создаются поля, индексы, ограничения
Property TableLevel: Integer;	Pu	Содержит уровень таблицы, используемый в драйвере BDE
Property TableName: TFileName;	Pb	Определяет имя таблицы
type TTableType = (ttDefault, ttParadox, ttDBase, ttASCII, ttFoxPro) ; property TableType: TTableType;	Pb	Определяет тип таблицы для стандартного драйвера BDE. ttDefault означает, что тип таблицы определяется по расширению файла
Методы		
Procedure AddIndex(const Name, Fields: string; Options: TIndexOptions);	Pu	Создает новый индекс. Параметр Name определяет имя нового индекса, параметр Fields — список полей индекса через запятую, параметр Options задает тип индекса
Procedure ApplyRange;	Pu	Включает в работу границы диапазона, заданные методами SetRangeStart, SetRangeEnd или EditRangeStart, EditRangeEnd
type TBatchMode = (batAppend, batUpdate, batAppendUpdate, batDelete, batCopy) ; function BatchMove(ASource: TBDEDataSet; AMode: TBatchMode): Longint;	Pu	Переносит записи из таблицы ASource в набор данных. Тип операции задается параметром AMode. Возвращает число обработанных записей

Продолжение табл. 12

Procedure CancelRange;	Pu	Удаляет текущий диапазон
Procedure CloseIndexFile(const IndexFileName: string);	Pu	Закрывает индексный файл для таблиц dBASE
Procedure CreateTable;	Pu	Создает новую таблицу, основываясь на данных о структуре таблицы, содержащихся в свойствах FieldDefs и IndexDefs. Если свойство FieldDefs пустое, используется свойство Fields. Структура и данные существующей таблицы перезаписываются
Procedure Deletelndex(const Name: string) ;	Pu	Удаляет вторичный индекс
Procedure DeleteTable;	Pu	Уничтожает таблицу базы данных. Набор данных должен быть закрыт
Procedure EditKey;	Pu	Переводит набор данных в режим редактирования буфера поиска. После использования этого метода можно изменять значения полей, используемые для поиска записей
Procedure EditRangeEnd;	Pu	Разрешает редактирование нижней границы диапазона
Procedure EditRangeStart;	Pu	Разрешает редактирование верхней границы диапазона
Procedure EmptyTable;	Pu	Удаляет все записи из набора данных
Function FindKey(const KeyValues: array of const): Boolean;	Pu	Проводит поиск записи, значения полей которой удовлетворяют условиям, заданным параметром KeyValues. Значения разделяются запятыми. Для поиска можно использовать только поля, входящие в текущий индекс. Для локальных стандартных таблиц BOE это поля, определяемые свойством IndexName. Для таблиц серверов SQL индекс можно задать свойствами indexName или IndexFieldNames. При успешном поиске функция возвращает True

Продолжение табл. 12

Procedure FindNearest(con5t KeyValues: array of const);	Pu	Проводит поиск записи, значения полей которой, заданные параметром KeyValues, в минимальной степени отличаются от требуемых в большую сторону. Значения для поиска разделяются запятыми. Для поиска можно использовать только поля, входящие в текущий индекс. Для локальных стандартных таблиц BOE это поля, определяемые свойством indexName. Для таблиц серверов SQL индекс можно задать свойствами IndexName или IndexFieldNames. При успешном поиске функция возвращает True
Procedure GetIndexNames(List: TStrings) ;	Pu	Возвращает список индексов таблицы
Procedure GotoCurrent(Table: TTable) ;	Pu	Синхронизирует курсор набора данных с курсором таблицы, заданной параметром Table
Function GotoKey: Boolean;	Pu	Устанавливает курсор на запись, соответствующую значениям полей, заданным при последнем применении методов SetKey или EditKey
Procedure GotoNearest;	Pu	Устанавливает курсор на запись, точно соответствующую значениям полей, заданным при последнем применении методов SetKey или EditKey, или ближайшую к ним по значениям в большую сторону
type TLockType = (ItReadLock, ItWriteLock) ; Procedure LockTable(LockType: TLockType) ;	Pu	Закрывает доступ к таблице Paradox или dBASE из других приложений
Procedure OpenIndexFile(const IndexFileName: string);	Pu	Открывает индексный файл таблицы dBASE
Procedure RenameTable(const NewTableName: string);	Pu	Переименовывает таблицу Paradox или dBASE

Окончание табл. 12

Procedure SetKey;	Pu	Очищает буфер поиска. После использования этого метода можно изменять значения полей, используемые для поиска записей
Procedure SetRange(const StartValues, EndValues: array of const);	Pu	Задает диапазон отбора записей. Параметр StartValues определяет значения полей для верхней границы диапазона. Параметр EndValues определяет значения полей для нижней границы диапазона. Значения диапазона задаются для полей текущего индекса
Procedure SetRangeEnd;	Pu	Задает нижнюю границу диапазона. После этого метода необходимо задать значения для полей текущего индекса, которые и будут нижней границей
Procedure SetRangeStart;	Pu	Задает верхнюю границу диапазона. После этого метода необходимо задать значения для полей текущего индекса, которые и будут верхней границей
type TLockType = (ItReadLock, ItWriteLock); Procedure UnlockTable(LockType: TLockType);	Pu	Разблокирует таблицу Paradox или dBASE для доступа из других приложений

В данной таблице Pu означает, что свойство или метод находится в общей секции Public, Pb – в секции публикаций Public, Ro – свойство доступно только для чтения.

Пример 1. Поиск в таблице строки с фамилией Иванов А.А.

```
With Table1 do Begin
    IndexFieldNames:=’FIO’; // определяем индексное поле
    SetKey;       // очищаем поле ключа поиска
    FieldByName(‘FIO’).asString:=’Иванов А.А.’; // задаем значение ключа
    If GotoKey then // найдена строка с фамилией Иванов А.А. в поле FIO
    Else          // не найдена искомая строка
    End;
```

Пример 2. Тот же поиск, но другим методом.

```
With Table1 do Begin
    IndexFieldNames:=’FIO’; // определяем индексное поле
    If FindKey([‘Иванов А.А.’]) then  // ищем Иванова А.А.
```

```
                    // найдена строка с фамилией Иванов А.А. в поле FIO
        else        // не найдена искомая строка
        end;
```

Пример 3. Увеличить в таблице оклады всех сотрудников в 2 раза.

```
Var I:integer;
……
With Table1 do Begin
        DiableControls;     // отключаем связь таблицы
                            // с визуальными компонентами
        First;          // Устанавливаем курсор в начало первой записи
        While Not Eof Do Begin // открываем цикл редактирования
                                // всех записей
            I:=FieldByName('OKLAD').asInteger; // узнаем старый оклад
            I:=I*2;         // увеличиваем оклад в 2 раза
            Edit;           // включаем режим редактирования таблицы
            FieldByName('OKLAD').asInteger:=I; // заносим новый оклад
            Post;           // записываем изменения в таблицу
            Next;           // переходим к следующей записи
            End;
        EnableControls;     // восстанавливаем связь таблицы с
                            // визуальными компонентами
        end;
```

Задания

При описании таблиц звездочками указаны ключевые поля.

1. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

   Таблица 1(основная), список сотрудников отдела. Таблица должна иметь следующие поля:

   1)* Номер сотрудника, char(2);
   2) Фамилия сотрудника, char(40);
   3) Дата его рождения, date;
   4) Код области рождения, char(2);
   5) Месячная зарплата, char(5).

   Таблица 2(зависимая), области Беларуси. Таблица должна иметь следующие поля:

   1)* Код области, char(2);
   2) Наименование области, char(20);

   На форму поместить по два компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полю кода области. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBEdit. Подсчитать месячный фонд зарплаты сотрудников отдела.

2. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

Таблица 1(основная), список сотрудников института. Таблица должна иметь следующие поля:
1)* Номер сотрудника, char(2);
2) Фамилия сотрудника, char(40);
3) Дата его рождения, date;
4) Код кафедры, char(2);
Таблица 2(зависимая), кафедры института. Таблица должна иметь следующие поля:
1)* Код кафедры, char(2);
2) Название кафедры, char(20);
На форму поместить по два компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полю кода кафедры. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBGrid. Построить зависимость частоты рождений сотрудников по месяцам.

3. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями три взаимосвязанные таблицы.

Таблица 1(основная), список международных телефонных разговоров. Таблица должна иметь следующие поля:
1)* Номер разговора, char(6);
2) Телефон звонившего, char(7);
3) Дата разговора, date;
4) Время начала разговора, Time;
5) Продолжительность разговора в минутах, Char(3);
6) Код города, куда звонили, char(4).
Таблица 2(зависимая), список телефонов. Таблица должна иметь следующие поля:
1)* Номер телефона, char(7);
2) Фамилия владельца, char(40);
3) Адрес владельца, char(60).
Таблица 3(зависимая), список городов. Таблица должна иметь следующие поля:
1)* Код города, char(4);
2) Название города, char(20).
На форму поместить по три компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям номера телефона и кода города. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBLabel. Подсчитать сумму оплаты за каждый разговор и общую сумму за все разговоры, введя произвольный поминутный тариф.

4. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями три взаимосвязанные таблицы.

Таблица 1(основная), список учета транспортных перевозок. Таблица должна иметь следующие поля:
1)* Номер перевозки, char(6);

2) Дата перевозки, date;
3) Код шофера, char(3);
4) Код машины, Char(3);
5) Вес груза в тоннах, char(2);
6) Длина пути в километрах., char(4).

Таблица 2(зависимая), список шоферов. Таблица должна иметь следующие поля:
1)* Код шофера, char(3);
2) Фамилия шофера, char(40);

Таблица 3(зависимая), список машин. Таблица должна иметь следующие поля:
1)* Код машины, char(4);
2) Марка машины, char(20).

На форму поместить по три компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода шофера и кода машины. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBGrid. Подсчитать сумму оплаты за каждую перевозку и общую сумму за все перевозки, введя произвольный тонно-километровый тариф.

5. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

Таблица 1(основная), список наличия книг в библиотеке. Таблица должна иметь следующие поля:
1)* Номер книги, char(6);
2) Авторы, char(50);
3) Название книги, char(50);
4) Код издательства, Char(3);
5) Количество страниц, char(4);
6) Цена, char(4),
7) Количество, char(3).

Таблица 2(зависимая), список издательств. Таблица должна иметь следующие поля:
1)* Код издательства, char(3);
2) Название издательства, char(40);

На форму поместить по два компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода издательства. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBGrid. Подсчитать стоимость книг каждого наименования и общую стоимость всех книг.

6. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

Таблица 1(основная), список наличия программного обеспечения в магазине. Таблица должна иметь следующие поля:
1)* Номер продукта, char(6);
2) Код фирмы изготовителя, char(3);

3) Наименование продукта, char(50);
4) Цена, Char(4);
5) Количество, char(4);
Таблица 2(зависимая), список фирм изготовителей. Таблица должна иметь следующие поля:
1)* Код фирмы, char(3);
2) Название фирмы, char(40);
На форму поместить по два компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода фирмы изготовителя. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBGrid. Подсчитать стоимость программ каждого наименования и общую стоимость всех продуктов.

7. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

Таблица 1(основная), список наличия товаров на складе. Таблица должна иметь следующие поля:
1)* Номер товара, char(6);
2) Код товара, char(3);
3) Количество, char(4);
4) Цена, char(4),
Таблица 2(зависимая), список товаров. Таблица должна иметь следующие поля:
1)* Код товара, char(3);
2) Название товара, char(40);
На форму поместить по два компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода товара. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBEdit. Подсчитать стоимость товара каждого наименования и общую стоимость всего товара.

8. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями три взаимосвязанные таблицы.

Таблица 1(основная), список проводок платежей в банке. Таблица должна иметь следующие поля:
1)* Номер платежа, char(6);
2) Дата платежа, date;
3) Номер счета плательщика, char(13);
4) Код банка плательщика, Char(3);
5) Номер счета получателя, char(13);
6) Код банка получателя, char(3);
7) Сумма платежа, char(7).
Таблица 2(зависимая), список счетов. Таблица должна иметь следующие поля:
1)* Номер счета, char(13);

2) Название организации, char(40);

Таблица 3(зависимая), список банков. Таблица должна иметь следующие поля:

1)* Код банка, char(4);

2) Название банка, char(20).

На форму поместить по три компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода шофера и кода машины. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBLabel. Подсчитать полную сумму платежей.

9. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

Таблица 1(основная), список поставленных на учет в ГАИ транспортных средств. Таблица должна иметь следующие поля:

1)* Учетный номер, char(8);

2) Дата постановки на учет, date;

3) Код машины, char(3);

4) Наименование владельца, Char(50);

5) Адрес владельца, char(50);

6) Номерной знак, char(7).

Таблица 2(зависимая), список машин. Таблица должна иметь следующие поля:

1)* Код машины, char(3);

2) Марка машины, char(40);

3) Вес машины в килограммах, char(5).

На форму поместить по два компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода шофера и кода машины. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBLabel. Подсчитать вес всех зарегистрированных машин.

10. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

Таблица 1(основная), список поставленных на продажу квартир. Таблица должна иметь следующие поля:

1)* Порядковый номер, char(6);

2) Дата постановки на продажу, date;

3) Код района, char(3);

4) Число комнат, Char(1);

5) Общая площадь в кв.метрах, char(4);

6) Наличие телефона, char(1)

7) Оценочная стоимость квартиры, char(7).

Таблица 2(зависимая), список районов. Таблица должна иметь следующие поля:

1)* Код района, char(3);

2) Название района, char(40);

На форму поместить по два компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода шофера и кода машины. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBEdit. Подсчитать полную стоимость всех квартир и среднюю стоимость одного квадратного метра жилья.

11. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

Таблица 1(основная), список комплектующих персональных компьютеров. Таблица должна иметь следующие поля:

1)* Порядковый номер, char(6);

2) Код изделия, char(3);

3) Цена, char(4);

4) Количество, Char(4);

Таблица 2(зависимая), список изделий. Таблица должна иметь следующие поля:

1)* Код изделия, char(3);

2) Наименование изделия, char(40);

3) Код фирмы изготовителя, char(3).

Таблица 3(зависимая), список фирм. Таблица должна иметь следующие поля:

1)* Код фирмы, char(4);

2) Название фирмы, char(20).

На форму поместить по три компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода изделия и кода фирмы изготовителя. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBGrid. Подсчитать количество изделий по каждой фирме.

12. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями четыре взаимосвязанные таблицы.

Таблица 1(основная), список пациентов врача. Таблица должна иметь следующие поля:

1)* Порядковый номер, char(6);

2) Код пациента, char(4);

3) Дата обращения, Date;

Таблица 2(зависимая), список пациентов. Таблица должна иметь следующие поля:

1)* Код пациента, char(4);

2) Фамилия пациента, char(40);

3) Возраст, char(3)

4) Адрес, char(40).

Таблица 3(зависимая), список признаков заболеваний. Таблица должна иметь следующие поля:

1)* Код признака, char(4);

2) Название признака, char(20).

Таблица 4(зависимая), список признаков заболеваний каждого пациента. Таблица должна иметь следующие поля:

1)* Код пациента, char(4);

2) Код признака, char(4);

На форму поместить по четыре компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода пациента и кода признака. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBGrid. Подсчитать среднее количество признаков заболевания для одного пациента. Для работы с четвертой таблицей использовать фильтрацию таблиц.

13. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями две взаимосвязанные таблицы.

Таблица 1(основная), список почтовых отправлений. Таблица должна иметь следующие поля:

1)* Порядковый номер, char(6);

2) Дата отправки, Date;

3) Время отправки, Time;

4) Код города назначения, Char(4);

5) Адрес получателя, char(50);

6) Фамилия получателя, char(40);

7) Адрес отправителя, char(50);

8) Фамилия отправителя, char(40);

9) Вес отправления в граммах, char(4);

Таблица 2(зависимая), список городов. Таблица должна иметь следующие поля:

1)* Код города, char(3);

2) Название города, char(40);

На форму поместить по два компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям кода изделия и кода фирмы изготовителя. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBGrid. Подсчитать количество отправлений в каждый город.

14. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями четыре взаимосвязанные таблицы.

Таблица 1(основная), список музыкальных компакт-дисков. Таблица должна иметь следующие поля:

1)* Порядковый номер, char(6);

2) Название компакт-диска, char(50);

3) Код фирмы изготовителя, char(3);

4) Количество, Char(4);

5) Цена, char(4).

Таблица 2(зависимая), список фирм изготовителей дисков. Таблица должна иметь следующие поля:

1)* Код фирмы, char(3);
2) Название фирмы, char(40);
Таблица 3(зависимая), содержание компакт-дисков. Таблица должна иметь следующие поля:
1)* Порядковый номер, char(4);
2)* Номер произведения на диске, char(2)
3) Код исполнителя, char(3)
4) Название произведения, char(50)
5) Длительность звучания в минутах, char(2).
Таблица 4(зависимая), список исполнителей. Таблица должна иметь следующие поля:
1)* Код исполнителя, char(3);
2) Фамилия исполнителя, char(40)
На форму поместить по четыре компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям порядкового номера, кода фирмы и кода исполнителя. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBLabel. Подсчитать количество произведений для каждого исполнителя.

15. С помощью программы DataBase DeskTop создать и частично заполнить несколькими записями три взаимосвязанные таблицы.

Таблица 1(основная), список выставленных на продажу автомобилей. Таблица должна иметь следующие поля:
1)* Порядковый номер, char(6);
2) Код марки машины, char(3);
3) Год выпуска, char(4);
4) Пробег в километрах, Char(4);
5) Цена, char(6).
Таблица 2(зависимая), список марок машин. Таблица должна иметь следующие поля:
1)* Код марки машины, char(3);
2) Название марки, char(40);
3) Код страны изготовителя, char(3).
Таблица 3(зависимая), список стран, производителей машин. Таблица должна иметь следующие поля:
1)* Код страны, char(3);
2) Название страны, char(40)
На форму поместить по три компонента Ttable и TdataSource для этих таблиц. Связать эти таблицы по полям порядкового номера, кода фирмы и кода исполнителя. Для навигации по основной таблицы использовать компонент TDBNavigator. Для визуализации таблиц использовать компонент TDBLabel. Подсчитать стоимость машин каждой марки, выставленных на продажу.

Литература

1. Закалюкин А.Б., Колосов С.В., Навроцкий А.А. и др. Программирование в среде Delphi: Лабораторный практикум для студентов всех специальностей // Мн.: БГУИР, 1998.
2. Дарахвелидзе П.Г., Марков Е.П., Delphi – среда визуального программирования // С.Пб.: BHV- Санкт-Петербург, 1996.
3. Даразвелидзе П.Г., Марков Е.П., Delphi 4 – среда визуального программирования // С.Пб.: BHV- Санкт-Петербург, 1999.
4. Фаронов В.В., Delphi 4, учебный курс // М.: Нолидх, 1999.
5. Эбнер М., Delphi 5, руководство разработчика // Киев: BHV, 2000.

Св. план 2001, поз. 10(вед.)

Учебное издание

Колосов Станислав Васильевич

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В СРЕДЕ DELPHI

Лабораторный практикум для студентов всех специальностей

Редактор Е.Н.Батурчик

Подписано в печать 26.10.2001 Формат 60x84 1/16
Бумага Печать офсетная.
Усл. печ. л. 2,5 Уч.-изд.л. Тираж 500 экз.
Заказ

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Лицензия ЛП №156 от 05.02.2001
Лицензия ЛВ №509 от 03.08.2001
220013, Минск, П.Бровки, 6