ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение высшего профессионального образования
«Владимирский государственный университет»
Кафедра конструирования и технологии радиоэлектронных средств»

Методические указания по выполнению лабораторных работ по курсу

Системное программирование

ВЛАДИМИР 2007

Содержание **Стр.**

Лабораторная работа N1

Разработка программы «Калькулятор»

Цель работы: получить навыки по визуальному проектированию интерфейса приложения и созданию компактных исходных текстов программ.

1. Основные положения

Сокращение сроков и затрат при разработке приложений является важнейшей проблемой в современном программировании. В последнее время все более широкое распространение получают средства ускоренной разработки программ (Rapid Application Development). Данный класс инструментальных средств основан на использовании

- принципов объектно-ориентированного программирования,
- визуального конструирования форм и
- использования библиотек визуальных компонентов.

К инструментам RAD относятся такие системы программирования, как Visual Basic, Visual C++, C++ Builder, Power Builder, Delphi и др. Система программирования Delphi выгодно отличается от аналогичных продуктов тем, что основывается на ясном и гибком языке программирования (Object Pascal), позволяющем создавать приложения любой сложности.

Визуальное конструирование форм позволяет среде программирования автоматически добавлять в исходный текст программы необходимые заготовки (фрагменты кода), а библиотеки визуальных компонентов включают в себя набор готовых типовых решений по созданию интерфейса приложения. Таким образом, работа программиста сводится к созданию прототипа будущего окна программы (формы) и наполнению его компонентами, реализующими необходимые интерфейсные свойства. Компонент – это фрагмент программного обеспечения, с которым во время разработки приложения можно работать из среды проектирования как с «черным ящиком», он включает в себя программный код и все необходимые для его работы данные. При необходимости его можно создать средствами самой среды Delphi и включить в библиотеку визуальных компонентов. После размещения на форме очередного компонента среда программирования автоматически вставляет в программный код ссылку на компонент и корректирует файл описания формы, который после компиляции преобразуется в ресурсный файл операционной системы.

Все компоненты Delphi согласно принципам объектно-ориентированного программирования являются потомками базовых классов и являются специальными типами, которые содержат поля, методы и свойства. Поля – это скрытые в классе данные, методы – процедуры и функции, свойства – механизм, регулирующий доступ к полям.

Наиболее востребованными визуальными компонентами в Delphi являются

TForm - форма

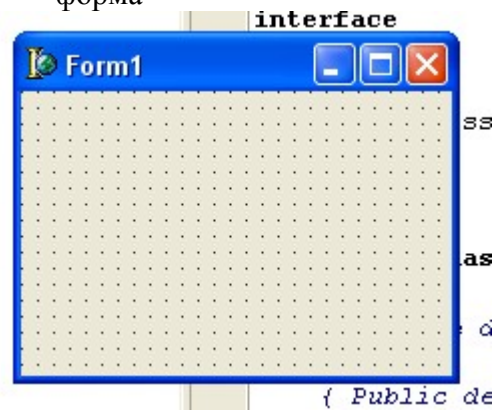


Рис. 1 Изображение заготовки формы (окно визуального редактора формы).

TPanel - панель

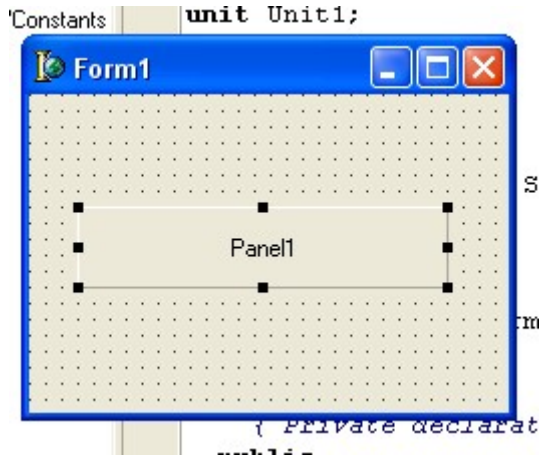


Рис. 2 Компонент TPanel, установленный на форму

TEdit - однострочное поле редактирования

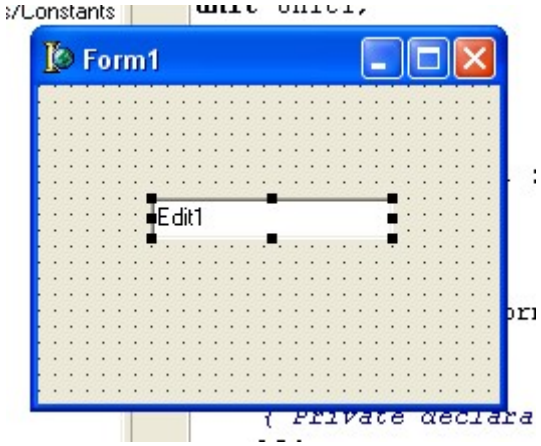


Рис 3. Компонент TEdit, установленный на форму

TButton (TSpeedButton или TBitBtn) - кнопка

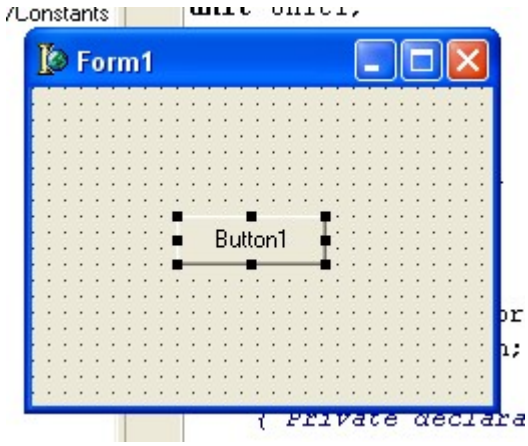


Рис 4. Компонент TButton, установленный на форму

Для всех компонентов можно выделить общие и специфические свойства (Property). К общим свойствам относятся

Name – имя компонента (при установке компонента на форму Delphi автоматически присваивает ему имя, совпадающее с именем класса без начальной буквы T и дополненное числовым суффиксом, через Инспектор объектов имя компонента можно изменять).

Tag - произвольное целое число, присваиваемое программистом (не используется Delphi, свойством можно распоряжаться по своему усмотрению)

Visible - видимость (Property Visible: Boolean; //true - показать)
Enabled - доступность (Property Enabled: Boolean; //true – сделать доступным)
Parent - родительский компонент (Property Parent: TWinControl; //родитель управляет видимым компонентом)
Top - положение верхней границы (Property Top: Integer //в пикселях)
Left - положение левой границы (Property Left: Integer //в пикселях)
Width - ширина компонента (Property Width: Integer //в пикселях)
Height - высота компонента (Property Height: Integer //в пикселях)
Align - выравнивание относительно границ его родителя (Property Align: TAlign,
возможные значения свойства:
alNone - не задано
alClient - вся область
alLeft - слева
alTop - сверху
alRight - справа
alBottom - снизу)
Font - шрифт надписи (Property Font: TFont)
Класс TFont содержит свойства
Name - имя (Property Name: TFontName
//например: 'Arial', 'MS Sans Serif', 'Courier New' , 'Symbol')
Color - цвет (Property Color: TColor
например: clWhite, clBlue, clNavy, clTeal, clRed, clBlack)
Style - стиль (Property Style: TFontStyles
возможные значения:
fsBold - полужирный
fsItalic - курсив
fsUnderline - подчеркнутый
fsStrikeOut - перечеркнутый)
Size - высота в пунктах (Property Size: Integer)
Height - высота в пикселях (Property Height: Integer)
Caption - текстовая строка для надписи на компоненте (Property Caption: TCaption).

К специфическим свойствам компонентов можно отнести следующие:
для компонента TEdit
Text - содержимое поля редактирования (Property Text: TCaption)
MaxLength - максимальное количество символов (Property MaxLength: Integer)
ReadOnly - разрешение (запрещение) редактирования (Property ReadOnly: Boolean)
Color - цвет фона компонента (Property Color: TColor)
AutoSelect - выделение текста при получении фокуса ввода (Property AutoSelect: Boolean)
для компонента TPanel
BevelInner - внутренняя кромка (Property BevelInner: TBevelCut
возможные значения
bvNone - нет эффекта объема
bvLowered - вдавленная кромка
bvRaised - выпуклая кромка
bcSpace - белая кромка)
BevelOuter - внешняя кромка
(аналогично BevelInner)
BevelKind - контрастность кромок (Property BevelKind: TBevelKind
возможные значения
bkNone - нет
bkTile - контрастные кромки
bkSort - пониженной контрастности кромки
bkFlat - белые кромки)
BevelWidth - ширина кромки (Property BevelWidth: Integer)
BorderWidth - толщина рамки (Property BorderWidth: Integer)
BorderStyle - стиль рамки
для компонента TForm
KeyPreview - обработка событий от клавиатуры (Property KeyPreview: Boolean)

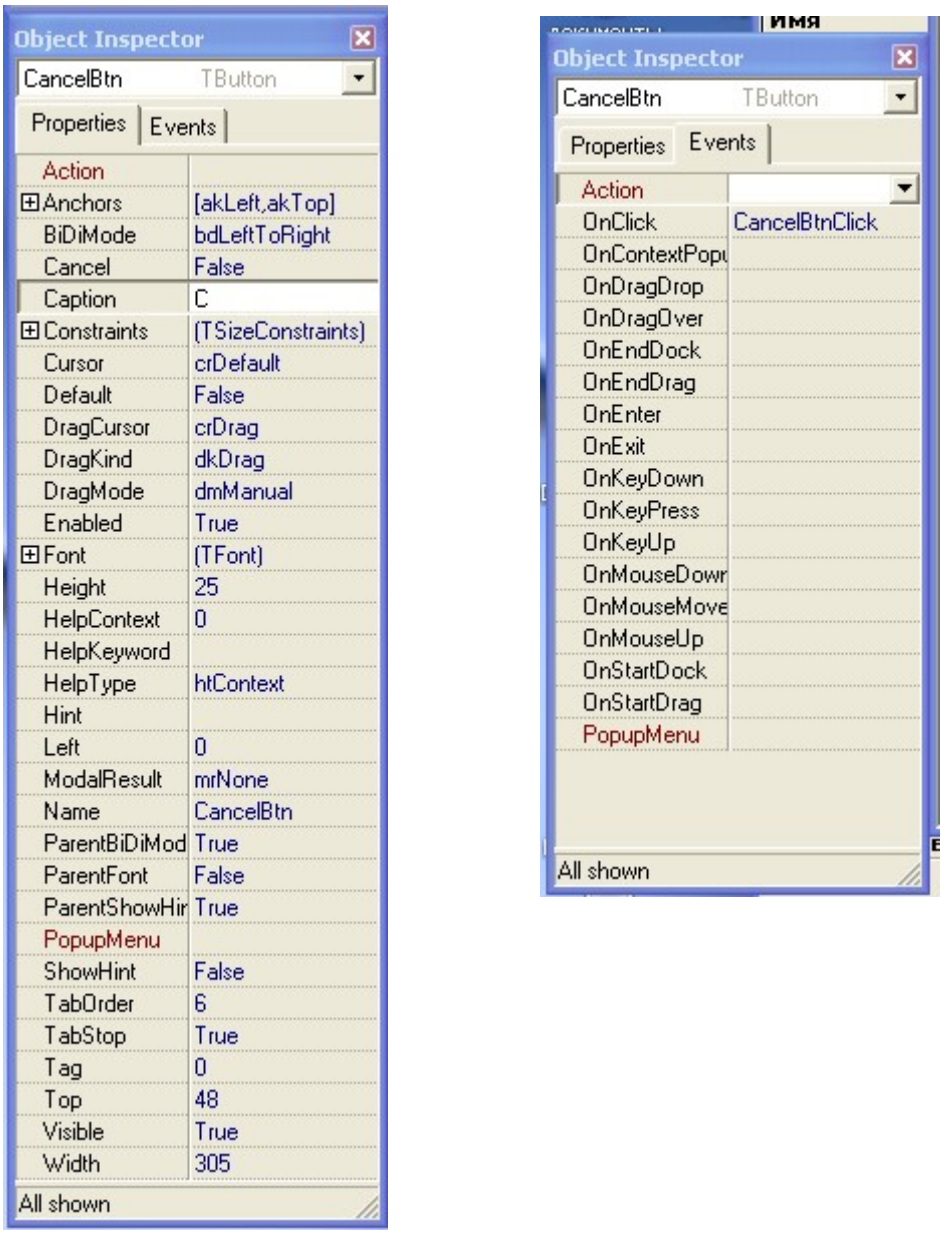


Рис 5. Инспектор объектов для компонента TButton (показаны все доступные свойства TButton и типовые отклики на события)

Обработчики событий компонентов (Events)
для компонента TButton
OnClick – возникает при щелчке на компоненте, происходит
анализ состояния переменной Sender: TObject.
для компонента TForm
OnKeyPress – возникает при нажатии клавиатуры

2. Выполнение работы

2.1. Задание на подготовку к работе

1. На своем профиле создать новую папку для хранения разрабатываемого приложения.
2. Запустить Delphi. Среда программирования создаст «пустой» проект.
3. Записать «пустой» проект в папку на своем профиле (пункт основного меню File->Save Project as)

2.2. Лабораторное задание

Задача лабораторной работы – разработать приложение «Калькулятор».

Программа «Калькулятор» должна выполнять арифметические действия (сложение, вычитание, умножение и деление) над целыми числами.

Рекомендуемый вариант интерфейса программы показан на рисунке Рис.6.

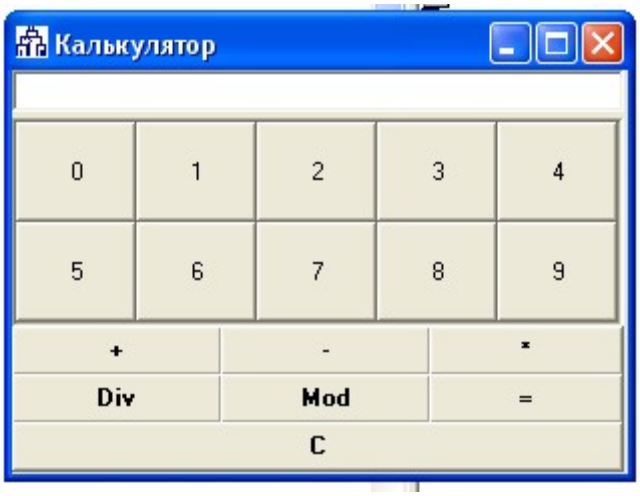


Рис.6 Окно запущенной на выполнение программы «Калькулятор»

3. Рекомендации при разработке приложения.

3.1.Используйте стандартные функции преобразования типов

StrToInt(идентификатор: String): Integer - перевод строковой переменной в целую

IntToStr(идентификатор: Integer): String - перевод целой переменной в строковую

3.2. Используйте оператор выбора по значению

case идентификатор **of**

значение 11 [12,13..]: действие 1;

значение 21 [22,23..]: действие 2;

. . . . . .

end;

3.3. При обработке щелчка мыши по цифровым кнопкам используйте обобщенный обработчик события (созданный только для одной кнопки!), связывание обработчика с другими кнопками осуществляется с помощью инспектора объектов. Внутри тела обработчика рационально использовать конструкцию (Sender as TButton).<необходимое свойство>

Пример программного кода, добавляющего в символьную строку свойства Text однострочного редактора Edit содержимое заголовка кнопки:

```
Edit.Text := Edit.Text + (Sender as TButton).Caption;
```

3.4. При обработке событий от клавиатуры в обработчике OnKeyPress в зависимости от состояния переменной Key (код нажатой клавиши) можно реализовать передачу в однострочный редактор Edit символа, соответствующего нажатой клавише или указание на арифметическое действие (возбуждается событие OnClick нажатой клавиши)

```
        case Key of
'0'..'9': Edit.Text := Edit.Text  + Key;
'+':      AddBtn.Click;
.       .       .       .       .       .
        End;
```

3.5. Для обработки событий от клавиатуры необходимо присвоить свойству KeyPreview формы значение true.

4. Содержание отчета

Отчет оформляется в форме Zip-архива, содержащего все файлы проекта (Project1.cfg, Project1.dof, Project1.dpr, Project1.exe, Project1.res, Unit1.dcu, Unit1.dfm, Unit1.pas). Zip-архив именуется по фамилии студента, номера группы и номера лабораторной работы.

5. Контрольные вопросы

1. Как сделать визуальный объект недоступным?
2. Как сделать визуальный объект невидимым?
3. Как изменить надпись на кнопке?
4. Как изменить цвет фона в однострочном поле редактирования?
5. Как убрать кнопку минимизации формы?
6. Как изменить положение компонента относительно границ своего родителя?
7. Как запретить редактирование текста в однострочном поле редактирования?
8. Как подчеркнуть надпись на кнопке?
9. Как вывести значение тега компонента в его надпись?
10. Как убрать заголовок панели?

Лабораторная работа N2

Разработка программы «Редактор списка строк»

Цель работы – получить навыки по визуальному проектированию интерфейса приложения в среде ускоренной разработки.

1. Основные положения.

За последнее время в информационных технологиях на первый план выдвигаются все более жесткие требования к скорости и качеству создания программ и, как следствие, использование сред ускоренной разработки приложений. Среда визуального проектирования берет на себя значительные объемы рутинной работы по подготовке приложений, создает предпосылки для коллективной разработки программ. Компонентный подход к созданию программ (в системе Delphi существует несколько сотен типовых компонентов) позволяет повторно использовать готовые разработки и во многих случаях значительно повышает эффективность труда программиста.

2. Выполнение работы

2.1. Задание на подготовку к работе

1. На своем профиле создать новую папку для хранения разрабатываемого приложения.
2. Запустить Delphi. Среда программирования создаст «пустой» проект.
3. Записать «пустой» проект в папку на своем профиле (пункт основного меню File->Save Project as).

2.2. Лабораторное задание

Задача лабораторной работы – разработать приложение, в котором обеспечиваются функции поддержания набора текстовых строк (компонент TListBox):
- считывание и запись в текстовый файл,
- добавление, изменение, удаление выбранных строк списка.

3. Рекомендации при разработке приложения.

3.1 Рекомендуемый интерфейс программы приведен на рис.1.

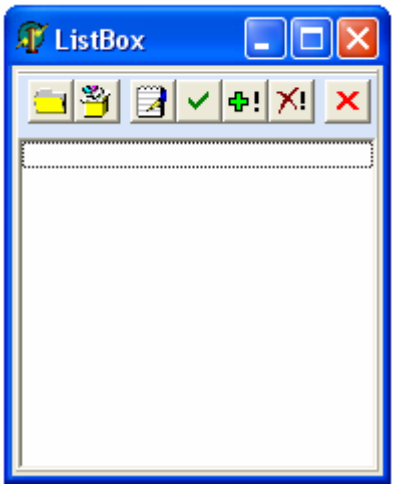


Рис.1 Окно запущенной на выполнение программы «Редактор списка строк».

3.2 Рекомендуемые визуальные компоненты:

TPanel	- панель
TToolBar	- панель инструментов
TToolButton	- кнопки панели инструментов
TImageList	- хранилище изображений
TOpenDialog	- диалог открытия файла
TSaveDialog	- диалог сохранения файла
TSplitter	- вешка разбивки
TEdit	- однострочное поле редактирования
TListBox	- список строк

3.3 Управление функциями редактора списка строк может быть обеспечено с помощью
- основного меню (компонент TMainMenu);
- всплывающего меню (компонент TPopupMenu);
- набора кнопок (компоненты TButton, TBitBtn и др.);
- набора связанных кнопок (компонент TSpeedButton);
- специализированного компонента панель инструментов TToolBar.

3.4 Минимально необходимый набор функций редактора:
- открытие файла с помощью диалогового компонента TOpenDialog и считывание содержимого файла в список строк TListBox (кнопка управления со всплывающей подсказкой в утопленном состоянии - рис. 2);

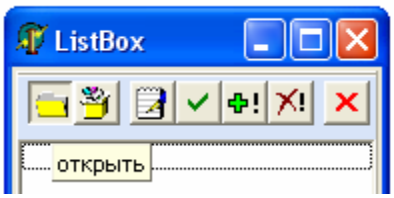


Рис.2 Вызов функции открытия файла.

- вызов окна редактирования строки TEdit (кнопка управления со всплывающей подсказкой в утопленном состоянии - рис. 3);

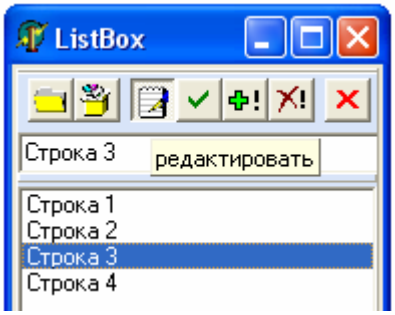


Рис.3 Вызов окна редактирования.

- отображение выбранной строки из списка в окне редактирования после щелчка курсором мыши по строке в компоненте TListBox (рис.3);

- сохранение отредактированной строки в списке (кнопка управления со всплывающей подсказкой в утопленном состоянии на рис. 4);

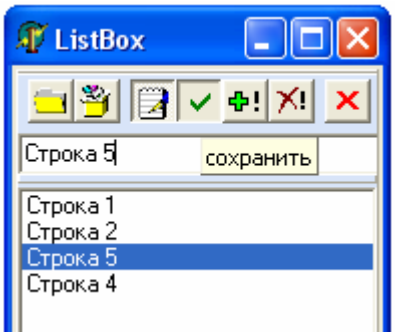


Рис.4 Сохранение отредактированной строки в списке.

- добавление новой строки в список (рис. 5);

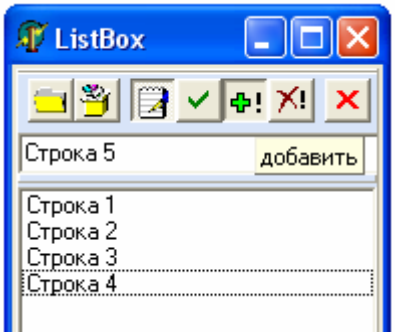


Рис.5 Добавление новой строки в список.

- удаление выбранной строки из списка (рис. 6);

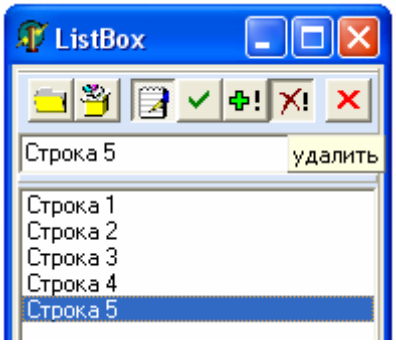


Рис.6 Удаление выбранной строки из списка.

- сохранение списка строк в файле (TSaveDialog) (рис.7).

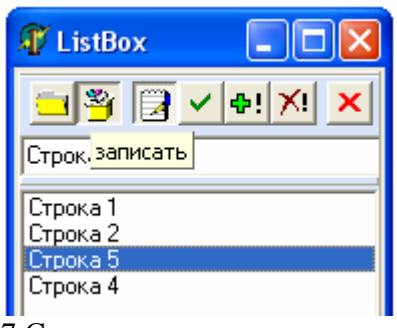
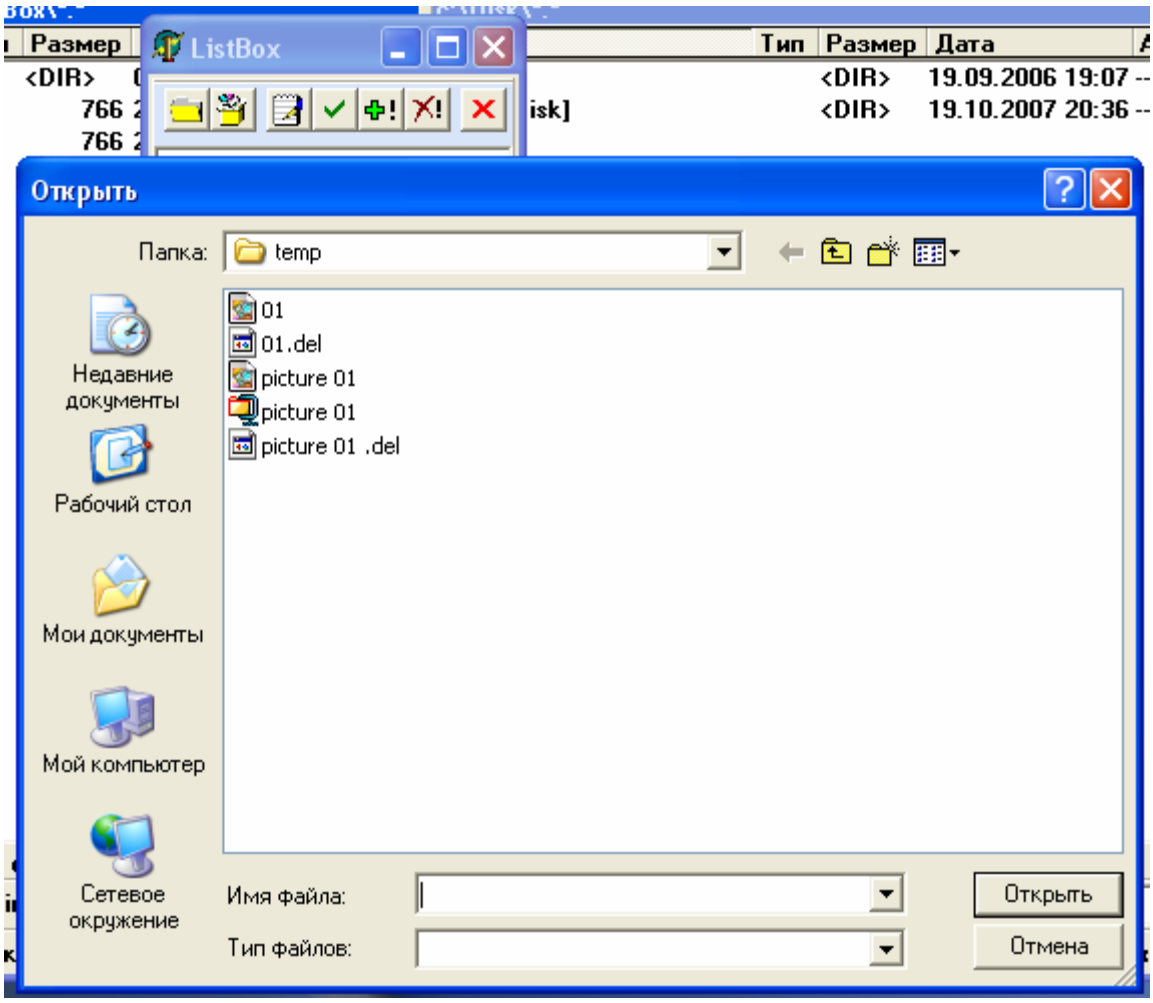


Рис.7 Сохранение списка строк в файле.

3.4 Вызов стандартного диалогового окна визуальным компонентом TOpenDialog:



3.6 Специфические свойства компонентов (property):

TListBox

Items: TStrings - набор строк, имеет методы

LoadFromFile() - считать из файла

SaveToFile() - записать в файл

Add() - добавить строку к списку

Delete() - удалить строку из списка

Insert() - вставить строку в список

Clear - очистить список

- свойство (только для чтения)

Count - количество строк списка

Sorted: Boolean - сортировка строк в алфавитном порядке

ItemIndex: Integer - индекс выделенной строки

Canvas: TCanvas - канва для программной прорисовки

TOpenDialog и TSaveDialog

FileName: String - путь поиска и имя выбранного файла

InitialDir: String - стартовый каталог

Filter: String - фильтр файлов, показываемых в диалоговом окне, устанавливается специальным редактором или программно, например

OpenDialog1.Filter:='Текстовые файлы*txt\Файлы Паскаля|*.pas';

Execute: Boolean - метод активизации диалогового окна

TImageList - контейнер для хранения графических изображений. При щелчке правой кнопкой мыши по значку компонента на форме во всплывающем меню имеется пункт ImageList Editor – открыть редактор для заполнения хранилища.

TToolButton разработан специально для TToolBar. При щелчке правой кнопкой мыши по компоненту TToolBar появляется контекстное меню, предлагающее выбрать New Button (новая кнопка) или New Separator (новый разделитель). Графические изображения кнопок TToolButton хранятся в контейнере TImageList. В свойство Images компонента TToolBar помещается имя хранилища графических изображений, а свойство ImageIndex компонента TToolButton определяет индекс связанного с кнопкой изображения.

TSplitter - предназначен для ручного (с помощью мыши) управления размерами контейнеров во время прогона программы. Устанавливается на форме между компонентами (панелями), размеры которых должны изменяться во время выполнения программы. Последовательность установки компонентов при использовании TSplitter:

- на пустую форму устанавливается один из разделяемых компонентов (Panel1),
- установленный компонент (Panel1) выравнивается по какой-либо границе формы (alLeft),
- с противоположной стороны компонента (правой) устанавливается разделитель (Splitter1)
- разделитель выравнивается по стороне контакта с компонентом (alLeft),
- в оставшуюся часть формы помещается второй разделяемый компонент (Panel2),
- второй разделяемый компонент (Panel2) выравнивается на всю свободную клиентскую область (alClient).

3.7 Необходимо обеспечить выполнение следующих особенностей:

- форма приложения (окно) полностью заполняется контейнером (панелью);
- при изменении размеров окна приложения соответствующим образом должны менять свои размеры компоненты, находящиеся внутри;
- внутри панели помещаются панель инструментов, список строк, разделитель и однострочное окно редактирования;
- в обычном состоянии окно редактирования невидимо, все поле окна занимают список строк и панель инструментов

4. Содержание отчета

Отчет оформляется в форме Zip-архива, содержащего все файлы проекта (Project1.cfg, Project1.dof, Project1.dpr, Project1.exe, Project1.res, Unit1.dcu, Unit1.dfm, Unit1.pas). Zip-архив именуется по фамилии студента, номера группы и номера лабораторной работы.

5. Контрольные вопросы

1. Можно ли хранить в компоненте список изображений TImageList рисунки с разными размерами?
2. Как в среде Delphi создать файл формата *.bmp с пиксельным изображением?
3. Как организовать замену рисунка на кнопке TSpeedButton по мере выполнения программы?
4. Как изменить стартовую позицию (папку) компонента TOpenDialog во время выполнения программы?
5. Как изменить фильтрацию файлов компонента TOpenDialog во время выполнения программы?
6. Как выделить группу связанных кнопок в компоненте TToolBar?
7. Можно ли на кнопках TToolButton одновременно выводить надпись и рисунок?
8. Как на компоненте TListBox рисовать пиксельные изображения?
9. Как на компоненте TListBox программным путем выделить строку?
10. Можно ли список строк компонента TListBox представить как одну строку?

Лабораторная работа N3

Модернизация программы «Графический редактор»

Цель работы: получить навыки в области использования графического инструментария Delphi.

1. Основные положения

Графический инструментарий Delphi основывается на использовании дескриптора графического устройства Windows и трех входящих в него инструментов: шрифта, пера и кисти. Для упрощения работы с графикой были введены четыре специализированных класса-надстройки:
TCanvas – для контекста (создает канву на которой можно рисовать пером, кистью и шрифтом),
TFont – для шрифта (создает объект-шрифт для любого графического устройства),
TPen – для пера (создает объект-перо для вычерчивания линий),
TBrush – для кисти (создает объект-кисть для заполнения внутреннего пространства замкнутых фигур).

Свойства класса TFont

property Charset: TFontCharSet;	Набор символов. Для русскоязычных программ это свойство обычно имеет значение OEFAULT_CHARSET или RUSSIAN_CHARSET. Используйте значение OEM_CHARSET для отображения текста MS-DOS
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: fpFixed задает моноширинный текст, при котором каждая буква имеет одинаковую ширину; fpVariable определяет пропорциональный текст, при котором ширина буквы зависит от ее начертания; fpDedault определяет ширину, принятую для текущего шрифта по умолчанию
property PixelPerInch: Integer;	Определяет количество пикселов экрана на один дюйм реальной длины. Это свойство не следует изменять, так как оно используется системой для обеспечения соответствия экранного шрифта шрифту принтера
property Size: Integer;	Высота шрифта в пунктах (у дюйма). Изменение этого свойства автоматически изменяет свойство Height, и наоборот
property Style: TFontStyles	Стиль шрифта. Может принимать значение как комбинацию следующих признаков:

fsBold (полужирный),
fsItalic (курсив),
fsUnderline (подчеркнутый),
fsStrikeOut (перечеркнутый)

Свойства класса TPen

property Color: TColor;	Цвет вычерчиваемых пером линий
property Handle: Integer;	Дескриптор пера. Используется при непосредственном обращении к API-функциям Windows
property Style: TPenStyle;	Определяет стиль линий (см. рис. 1). Учитывается только для толщины линий в 1 пиксель. Для толстых линий стиль всегда psSolid (сплошная)
property Width: Integer;	Толщина линий в пикселях экрана
property Mode: TPenMode;	Определяет способ взаимодействия линий с фоном: pmBIlack — линии всегда черные (Color и Style игнорируются); pmWhite — линии всегда белые (Color и Style игнорируются); pmNop — цвет фона не меняется (линии не видны); pmNot — инверсия цвета фона (Color и Style игнорируются); pmCopy — цвет линий определяется свойством Color пера; pmNotCopy — инверсия цвета пера (Style игнорируется); pmMergePenNot — комбинация цвета пера и инверсионного цвета фона; pmMaskPenNot — комбинация общих цветов для пера и инверсионного цвета фона (Style игнорируется); pmMergeNotPen — комбинация инверсионного цвета пера и фона; pmMaskNotPen — комбинация общих цветов для инверсионного цвета пера и фона (Style игнорируется); pmMerge — комбинация цветов пера и фона; pmNotMerge — инверсия цветов пера и фона (Style игнорируется); pmMask — общие цвета пера и фона; pmNotMask — инверсия общих цветов пера и фона; pmXor — объединение цветов пера и фона операцией X0R; pmNotXor — инверсия объединения цветов пера и фона операцией X0R.

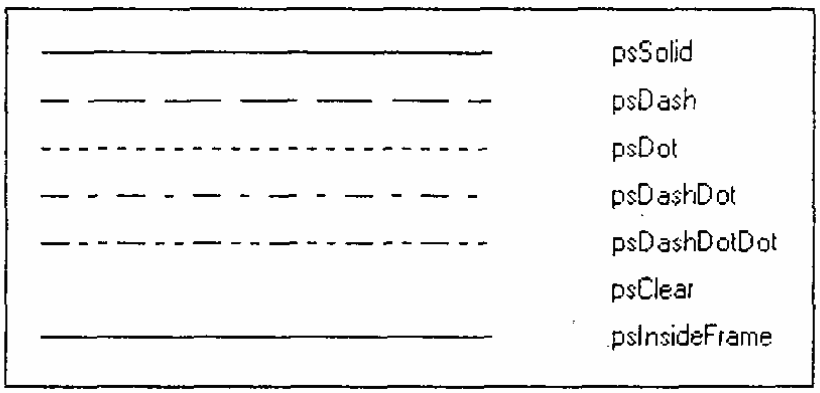


Рис 1. Стили пера

Свойства класса TBrush

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 (см. рис.2)

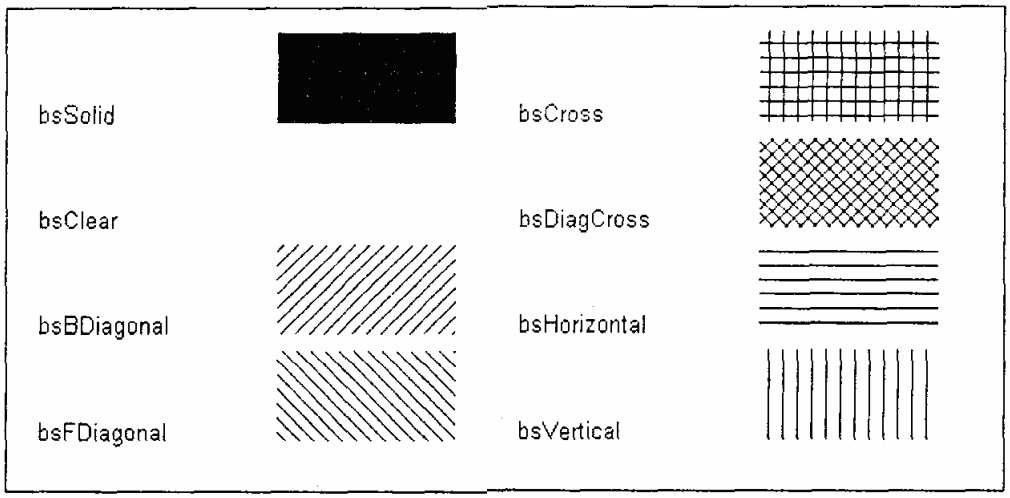


Рис. 2 Стили кисти

Свойства класса TCanvas

property Brush: TBrush;	Объект-кисть
property ClipRect: TRect;	Определяет текущие размеры области, нуждающейся в прорисовке
property CopyMode: TCopyMode;	Устанавливает способ взаимодействия растрового изображения с цветом фона
property Font: TFont;	Объект-шрифт
property Handle; Integer;	Дескриптор канвы. Используется при непосредственном обращении к API-функциям Windows
property LockCount: Integer;	Счетчик блокировок канвы. Увеличивается на единицу при каждом обращении к методу Lock и уменьшается на единицу при обращении к методу Unlock
property Pen: TPen;	Объект-перо
property PenPos: TPoint;	Определяет текущее положение пера в пикселах относительно левого верхнего угла канвы
property Pixels[X.Y: Integer]: TColor;	Массив пикселов канвы

Свойство CopyMode используется при копировании части одной канвы (источника) в другую (приемник) методом CopyRect и может иметь одно из следующих значений:

- cmBlackness — заполняет область рисования черным цветом;
- cmDestInvert — заполняет область рисования инверсным цветом фона;
- cmMergeCopy — объединяет изображение на канве и копируемое изображение операцией AND;

O cmMergePaint — объединяет изображение на канве и копируемое изображение операцией OR;

O cmNotSrcCopy — копирует на канву инверсное изображение источника;

O cmNotSrcErase — объединяет изображение на канве и копируемое изображение операцией OR и инвертирует полученное;

O cmPatCopy — копирует образец источника;

O cmPatInvert — комбинирует образец источника с изображением на канве с помощью операции X0R;

O cmPatPaint — комбинирует изображение источника с его образцом с помощью операции OR, затем полученное объединяется с изображением на канве также с помощью операции OR;

O cmSrcAnd — объединяет изображение источника и канвы с помощью операции AND;

O cmSrcCopy — копирует изображение источника на канву;

O crnSrcErase — инвертирует изображение на канве и объединяет результат с изображением источника операцией AND;

O cmSrcInvert — объединяет изображение на канве и источник операцией X0R;

O cmSrcPaint — объединяет изображение на канве и источник операцией OR;

O cmWhitness — заполняет область рисования белым цветом.

С помощью свойства Pixels все пикселы канвы представляются в виде двухмерного массива точек. Изменяя цвет пикселов, можно прорисовывать изображение по отдельным точкам.

Методы класса TCanvas

установка шрифта, пера и кисти по умолчанию

```
procedure Refresh;
```

рисование линий

```
procedure MoveTo(X. Y: Integer):
procedure LineTo(X. Y: Integer);
procedure Polyline(Points: array of Point);
```

рисование фигур

```
procedure Polygon(Points: array of Point);
procedure Rectangle(X1, Y1, X2, Y2: Integer);
procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
```

рисование эллипса

```
procedure Elllpse(Xl. Yl, X2.Y2: Integer);
procedure Arc(XI, Yl. X2. Y2. X3. Y3, X4. Y4: Integer);
procedure Chord(Xl. Yl, X2. Y2. X3. Y3. X4. Y4: Integer);
procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
```

вывод текстовых строк

```
function TextExtent(const Text: string): TSize;
function TextHeight(const Text: string): Integer;
procedure TextOut(X,Y: Integer; const Text: string);
procedure TextRect(Rect: TRect; X,Y: Integer; const Text: string);
function TextWidth(const Text: string): Integer;
```

копирование изображений

```
procedure BrushCopy (const Dest: TRect; Bitmap: TBitmap; const Source: TRect: Color: TColor);
procedure CopyRect (Dest: TRect: Canvas: TCanvas: Source: TRect);
```

прорисовка графического объекта

```
procedure Draw(X. Y: Integer: Graphic: TGraphic);
procedure StretchDraw(const Rect: TRect; Graphic: TGraphic );
procedure FloodFill(X, Y: Integer; Color: TColor; FillStyle: TFillStyle);
```

прорисовка прямоугольных областей

```
procedure DrawFocusRect (const Rect: TRect);
procedure FillRect (const Rect: TRect);
```

```
procedure FrameRect(cons Rect: TRect);
```

блокировка канвы в многопоточных приложениях

```
procedure Lock;
function TryLock: Boolean;
procedure Unlock.
```

Класс TCanvas инкапсулирует лишь часть приемов работы с графическими инструментами Windows, в нем ограниченно представлены возможности по текстовому выводу и прорисовке областей.

2. Выполнение работы

2.1. Задание на подготовку к работе

1. На своем профиле создать новую папку для хранения разрабатываемого приложения.
2. Скопировать в созданную папку проект Graphex (находится во внутренней папке Demos\Doc\graphex).
3. Запустить Delphi и открыть проект Graphex.

2.2. Лабораторное задание

Задача лабораторной работы – встроить в существующее приложение средства, позволяющие рисовать заданную фигуру.

3. Рекомендации при разработке приложения.

3.1 При запуске программы Graphex открывается окно графического редактора с набором элементов управления (рис 3.)

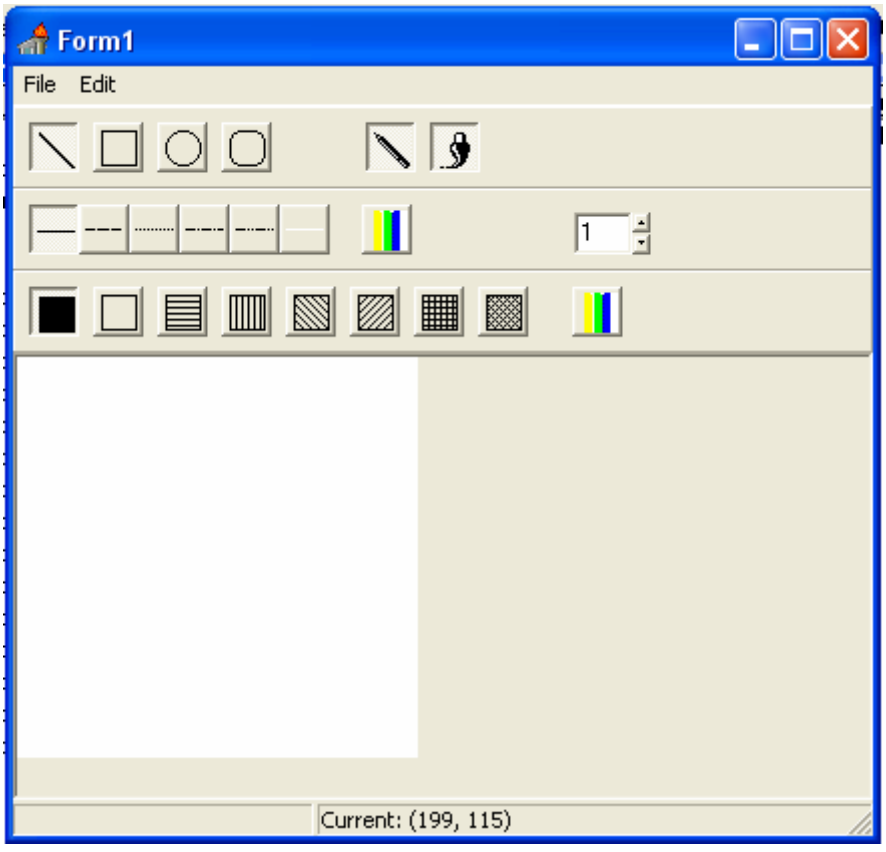


Рис.3 Окно графического редактора.

3.2 К имеющемуся набору кнопок для выбора фигур следует добавить новую с изображением заданной фигуры. Пиктограмму фигуры можно создать с помощью встроенного в Delphi графического редактора Image Editor (вызывается из основного меню кнопкой Tools).

3.3 Перечень фигур для задания представлен на рис. 4

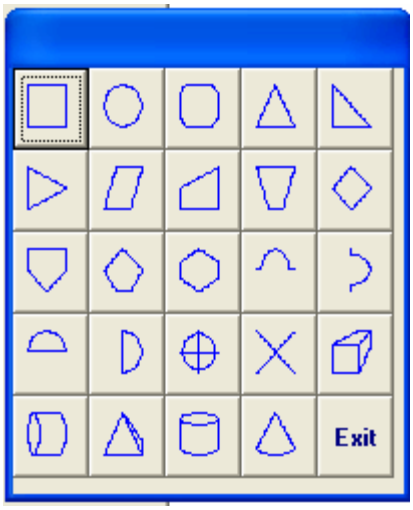


Рис.4 Варианты фигур для задания.

3.4. В графическом редакторе Graphex введен тип данных TDrawingTool, содержащий именованные константы фигур:

```
type
  TDrawingTool = (dtLine, dtRectangle, dtEllipse, dtRoundRect);
```

С этим типом данных связана переменная DrawingTool: TDrawingTool, в которой хранится текущий код рисуемой фигуры. Управление состоянием переменной DrawingTool осуществляется нажатием соответствующей кнопки выбора фигур, например

```
procedure TForm1.LineButtonClick(Sender: TObject);
begin
  DrawingTool := dtLine;
end;
```

Рисование фигур обеспечивает процедура

```
 DrawShape(TopLeft, BottomRight: TPoint; AMode: TPenMode);
```

в которой в зависимости от переменной DrawingTool вызываются соответствующие подпрограммы, например

```
  case DrawingTool of
     dtLine:
       begin
         Image.Canvas.MoveTo(TopLeft.X, TopLeft.Y);
         Image.Canvas.LineTo(BottomRight.X, BottomRight.Y);
       end;
.       .       .
```

4. Содержание отчета

Отчет оформляется в форме Zip-архива, содержащего все файлы проекта (Project1.cfg, Project1.dof, Project1.dpr, Project1.exe, Project1.res, Unit1.dcu, Unit1.dfm, Unit1.pas). Zip-архив именуется по фамилии студента, номера группы и номера лабораторной работы.

5. Контрольные вопросы

1. Какие способы получения графических изображений вы знаете?
2. Какой класс Delphi интерпретирует графический объект как растровое изображение?
3. Какой класс Delphi интерпретирует графический объект как значок?
4. В файлах с каким расширением рекомендуется хранить пиксельные рисунки?
5. Как очистить область рисования?
6. Можно ли рисовать непосредственно на форме без использования компонента TImage?
7. Какой класс Delphi предназначен для хранения координат точки?
8. Каким свойством определяется текущее положение пера?
9. Как определить положение пера после вывода надписи?
10. Можно ли изменять стиль надписи (наклонная, толстая и т.д.) при ее выводе в область рисования?

Список рекомендуемой литературы

1. Фаронов В.В. Delphi 6: учебный курс. СПб: Питер, 2002 – 512с.
2. Бабушкина И.А. Практикум по объектно-ориентированному программированию. – М.: «БИНОМ», 2004. – 366 с.
3. Архангельский А.Я. Object Pascal в Delphi. – М.: ЗАО «БИНОМ», 2002. – 384 с.
4. Компьютерный учебник: «Delphi. Среда быстрой разработки». - М.: «Alex Soft», 2002.
5. Озеров В. Delphi Советы программистов. - СПб: Символ-Плюс, 2004. – 976 с.