МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ

ПО РАБОТЕ С ГРАФИЧЕСКИМИ СРЕДСТВАМИ DELPHI

Донецк 2007

УДК 681.3.06(071)

Методическое пособие по работе с графическими средствами Delphi/
Сост.: Л. В. Незамова, Е. В. Гомжина. – Донецк, ДонНТУ, 2007. – 40 с.

В методическом пособии приведены справочные сведения по графическим средствам Delphi, рассмотрена технология визуального программирования, приведены задания и примеры решения типовых задач.

Методическое пособие можно использовать в качестве основы курсов лекций, в которых изучаются соответствующие программные продукты. Пособие предназначено для студентов, аспирантов, преподавателей.

Авторы: Л.В. Незамова, ассистент
Е.В. Гомжина, ассистент

Рецензент

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Delphi – мощная система визуального объектно-ориентированного программирования. С ее помощью, даже начинающие программисты, могут быстро создавать оконные интерфейсы, удовлетворяющие стандартам Windows. Спектр отраслей, в которых возможно применение Delphi, достаточно широк: инженерные, офисные, торговые и другие.

Очень удобная методика создания различных типов приложений, способы вывода готовых и подготовки собственных графических изображений, приемы демонстрации анимационных изображений, а так же методика построения диаграмм и графиков с использованием специальных компонентов. Разработчики Delphi уделили большое внимание возможностям работы с деловой графикой: простота и удобство ее использования напрямую сказывается на простоте и удобстве созданных приложений.

1. СОЗДАНИЕ МЕНЮ

В приложениях, написанных на Delphi, могут быть реализованы меню двух основных видов:

• Главное меню. Такое меню принадлежит форме и отображается вместе с ней под ее панелью заголовка. Если в приложении несколько форм, то для удобства можно объединять меню всех активных форм в одном.

• Контекстное меню. Такое меню предусмотрено почти у всех компонентов — элементов управления Windows. Оно возникает (всплывает) при нажатии правой кнопки мыши на этом компоненте.

1.1 Главное меню приложения - MainMenu

Многие приложения Delphi имеют главное меню, содержащее перечень допустимых действий. Меню являются иерархическими структурами, состоящими из пунктов меню. Пункты главного меню называют элементами верхнего уровня, так как с каждым из них может быть связано всплывающее подменю. Каждый пункт может быть выбран. Это может произойти вследствие щелчка кнопкой мыши, нажатия соответствующих клавиш на клавиатуре или вызова процедуры в программе. На нижнем уровне лежат команды меню — пункты, с выбором которых должна быть связана та или иная реакция приложения. Команды объединяются в подменю. Подменю — это пункты, выбор которых означает показ или свертывание списка входящих в него команд и подменю.

В Delphi для создания главного меню используется невизуальный компонент MainMenu. Компонент расположен на странице Standard Палитры Компонентов.

Принципы создания и работы с меню в Delphi очень просты. Каждому пункту меню соответствует свой компонент класса TMenuItem. Вы добавляете к меню новые пункты (а к форме — новые компоненты) либо во время разработки (при помощи Конструктора меню), либо во время исполнения.

Для того, чтобы создать главное меню на этапе конструирования необходимо поместить компонент MainMenu на форму и вызвать конструктор меню. Щелкните дважды левой кнопкой мышки по MainMenu и на экране появится конструктор меню. С помощью конструктора можно набрать элементы меню (рис.1.1). Можно использовать Инспектор Объектов, для задания свойств каждого элемента меню.

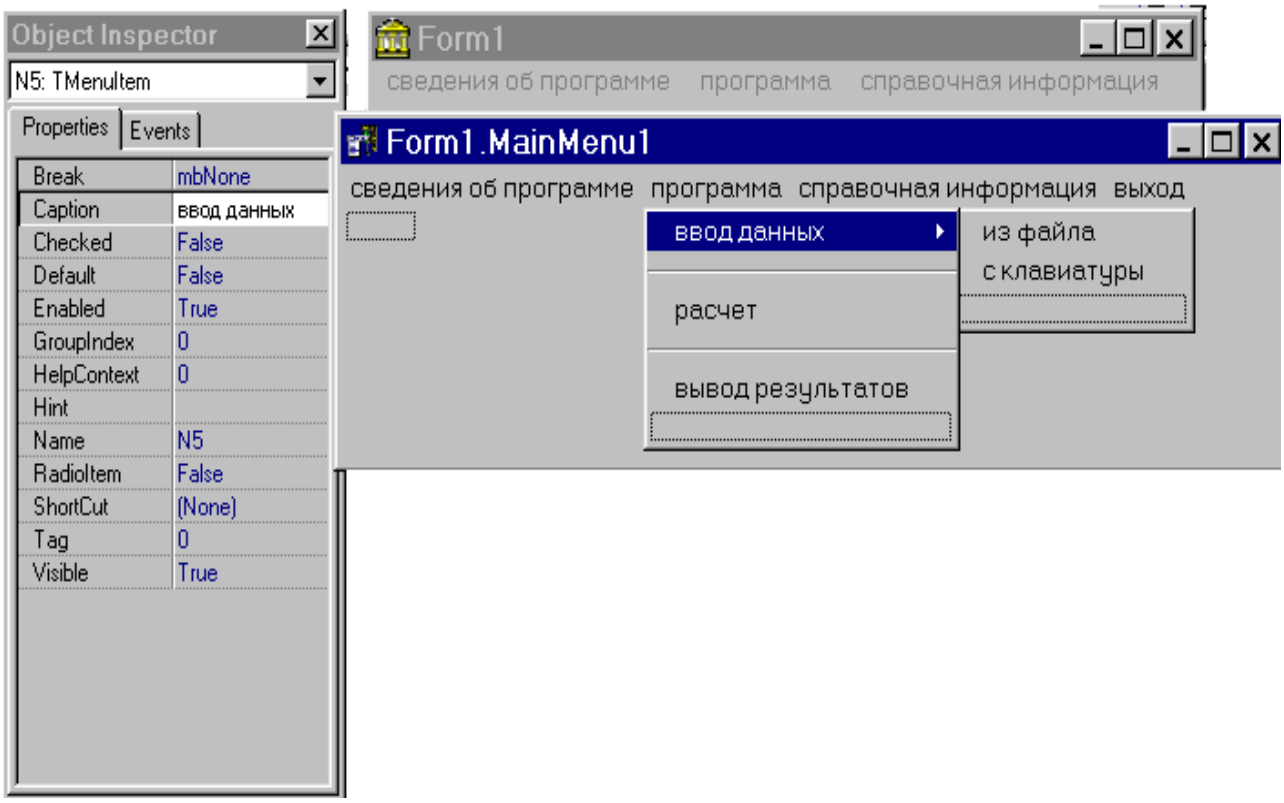


Рис. 1.1 – Окно конструктора меню

Элементы меню могут представлять собой подменю, разделительную линию или команду. Если элемент представляет подменю, свойство Items у него должно содержать пункты этого подменю. Если свойство Caption элемента содержит знак минус (-), то этот элемент является разделительной чертой. Во всех остальных случаях элемент меню является командой. При выборе такого пункта меню для описывающего его компонента инициируется событие OnClick, в обработчике которого и нужно предусмотреть соответствующие действия.

Для того чтобы удалить или вставить элемент меню, добавить подменю подключите контекстное меню этого элемента и выберите соответствующий пункт (рис. 1.2)

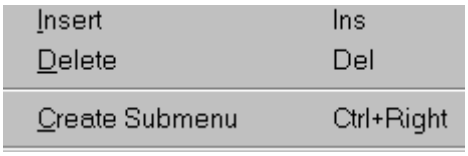


Рис. 1.2 – Контекстное меню

Основные свойства меню представлены в таблице 1.1.

Таблица 1.1 – Основные свойства MainMenu

Название	Обозначение
Caption	Название элемента меню.
Checked	По умолчанию False True – элемент меню помечается “галочкой”
Default	По умолчанию False True – элемент меню выделяется полужирным шрифтом
Enabled	По умолчанию True False – элемент недоступен и выделяется тусклым цветом
ShortCut	Определяет комбинацию “горячих клавиш” для быстрого выбора элемента меню

1.2 Контекстное меню – PopupMenu

Для создания контекстного меню используется невизуальный компонент PopupMenu. Компонент расположен на странице Standard Палитры Компонентов.

Этот компонент описывает всплывающее меню. В отличие от главного, собственное меню такого типа может быть почти у каждого оконного элемента управления на форме (кроме переключателей), а также у самой формы.

Контекстное меню, так же как и главное меню можно создать с помощью конструктора меню. Всплывающее меню обычно связывают с нажатием правой кнопки мыши по оконному элементу. Чтобы установить связь между оконным

элементом и компонентом PoupMenu у оконного компонента используют свойство PoupMenu, в котором необходимо указать имя соответствующего компонента-меню.

Основные свойства контекстного меню представлены в таблице 1.2.

Таблица 1.2 – Основные свойства контекстного меню

Название	Обозначение
Alignment	Определяет расположение контекстного меню
AutoPopup	По умолчанию True – контекстное меню появляется при нажатии правой кнопкой мышки по объекту

1.3 Задание

Создать форму. Разработать главное меню. Меню верхнего уровня должно содержать пункты: ЗАСТАВКА, вкладки для последующих трех лабораторных работ, СПРАВОЧНЫЕ ДАННЫЕ, ВЫХОД.

2. ГРАФИЧЕСКИЕ ИНСТРУМЕНТЫ

Разработка приложений в Delphi предполагает включение в программные продукты графиков, диаграмм, рисунков.

В стандартном графическом интерфейсе MicroSoft Windows GDI (Graphics Device Interface включает набор программ, воспроизводящих графику) основой для рисования служит HDC (Handle Device context) – дескриптор контекста устройства – и связанные с ним шрифт(Font), перо(Pen) и кисть(Brush). В состав библиотеки компонентов Delphi входят графические надстройки, назначением которых является обеспечение удобного доступа ко всем свойствам указанных выше инструментов.

Шрифт, перо и кисть, как правило, используются не самостоятельно, а в составе специального класса TCanvas. Этот класс объединяет в себе ”холст”, рабочие инструменты (перо, кисть, шрифт), а также набор функций по рисованию геометрических фигур. Свойство типа TCanvas называют канвой. Канва входит в качестве свойства во многие компоненты, в частности, TImage. Изображение на канву компонента TImage можно переносить из файла с помощью свойства Picture. При этом можно загружать растровые изображения (bmp, pcx), пиктограммы в виде иконок (файлы ico), сжатые в формате jpeg и jpg изображения.

2.1 Специализированные классы-надстройки

- *Класс TCanvas – Холст*

Этот класс объединяет в себе и "холст" и "рабочие инструменты" (перо, кисть, шрифт) и набор функций по рисованию типовых геометрических фигур.

Для обозначения этого класса можно использовать термин "канва". Канва не является компонентом, но она входит в качестве свойства во многие другие компоненты, которые должны уметь нарисовать себя и отобразить какую-либо информацию (ниже приводятся компоненты, которые будут полезны при выполнении лабораторных работ).

Компонент Timage (Страница Палитры компонентов Additional) замыкает всю иерархию графических объектов VCL. Он предназначен для показа на форме изображения: битовой карты (TBitmap), метафайла (TMetafile), значка (TIcon).

Компонент Tshape (страница Палитры компонентов Additional) предназначен только для того, чтобы нарисовать и показать на своей поверхности простейшие геометрические фигуры. Рисуемая фигура — одна из предопределенных в множестве:

TShapeType=(stRectangle,stSquare,stRoundRect,stRoundSquare,stEllipse,stCircle)

(прямоугольник, квадрат, скругленный прямоугольник, скругленный квадрат, эллипс, окружность). Размеры фигуры соответствуют размерам компонента. Если задан квадрат или круг, а ширина и высота компонента не равны между собой, то сторона или диаметр фигуры будут равны длине меньшей стороны компонента.

Рассмотрим координатную плоскость холста. Холст состоит из отдельных точек — пикселов, в пикселах измеряются координаты. Пиксел – это наименьший элемент поверхности рисунка, с которым можно манипулировать. Положение пиксела характеризуется его горизонтальной (X) и вертикальной (Y) координатами. Система координат имеет началом левый верхний угол холста и имеет координаты (0, 0). Координаты возрастают сверху вниз и слева направо. Значения координат правой нижней точки холста зависят от размера холста. Размер холста можно получить, обратившись к свойствам Height и Width (области иллюстрации Image) или к свойствам формы: ClientHeight и ClientWidth. На рис.2.1 представлены координаты точек холста.

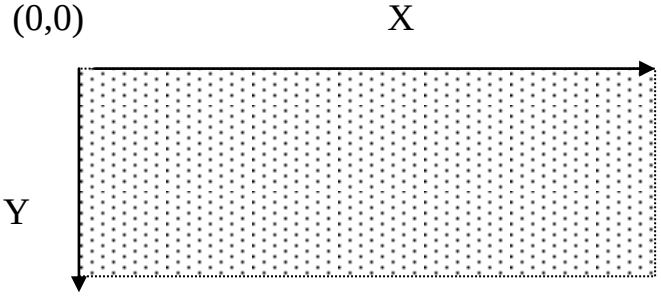


Рис.2.1 - Координаты точек холста

Пример1. Начертить прямоугольник в окне программы (рис.2.2)

Программный код -> Canvas.Rectangle(50,10,150,50)

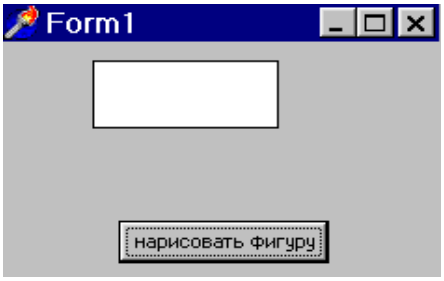


Рис.2.2 – Прямоугольник

- *Класс TFont – Шрифт*

Шрифт, который используется для вывода текста, определяется значением свойства Font соответствующего объекта Canvas. Свойство Font представляет собой объект типа TFont. В табл. 2.1 перечислены свойства объекта TFont, позволяющие задать характеристики шрифта, используемого методами TextOut и TextRect для вывода текста.

Таблица 2.1 – Свойства класса TFont

Color	Цвет шрифта
Height	Высота шрифта в пикселях экрана
Name	Имя шрифта. По умолчанию MS Sans Serif
Size	Высота шрифта в пунктах. Пункт— это единица измерения размера шрифта, используемая в полиграфии. Один пункт равен 1/72 дюйма. Изменение этого свойства автоматически изменяет свойство Height и наоборот.
Style	Стиль шрифта. Может быть: нормальным, полужирным, курсивным, подчеркнутым, перечеркнутым. Стиль задается при помощи следующих констант: fsBold (полужирный), fsltalic (курсив), fsUnderline (подчеркнутый), fsStrikeOut (перечеркнутый).

Для вывода текста на поверхность графического объекта используется метод TextOut. Инструкция вызова метода TextOut в общем виде выглядит следующим образом:

Объект.Canvas.TextOut(x, y, Текст)

где: объект — имя объекта, на поверхность которого выводится текст; x, y — координаты точки графической поверхности, от которой выполняется вывод текста; Текст — переменная или константа символьного типа, значение которой определяет выводимый методом текст.

При работе с текстом, можно использовать следующие функции:

TextHeight - Возвращает высоту строки Text в пикселах;

TextWidth - Возвращает ширину строки Text в пикселах;

TextRect - Производит вывод текста с отсечением. Как и в TextOut, строка Text выводится с позиции (X,Y); при этом часть текста, лежащая вне пределов прямоугольника Rect, отсекается и не будет видна.

Пример2. Вывести текст на форме в заданных координатах 10,10 (рис.2.3)

Программный код -> Canvas.TextOut(10,10,'вывод текста с позиции 10,10 ')

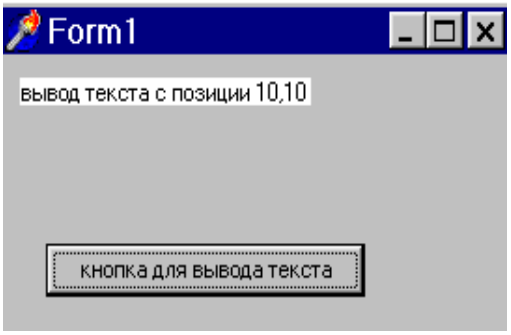


Рис.2.3 – Вывод текста

Пример3. Вывести текст на форме в заданных координатах 10,10. Шрифт полужирный, размер 15 пунктов. (рис.2.4)

Программный код -> Canvas.Font.Size:=15;

```
Canvas.Font.Style:=[fsBold];
Canvas.TextOut(10,10,'вывод текста с позиции 10,10 ');
```

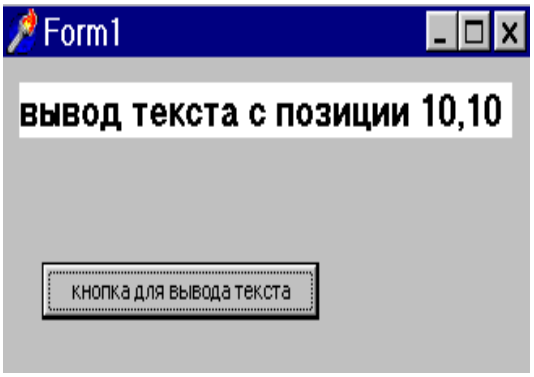


Рис.2.4 – Вывод текста

- *Класс TPen – Карандаш*

Класс TPen задает характеристики карандаша, при помощи которого создаются изображения различных линий или контуров.

Вид линии, которую оставляет карандаш на поверхности холста, определяют свойства объекта TPen, которые перечислены в табл. 2.2.

Таблица 2.2 – Свойства класса TPen

Color	Цвет линии, рисуемой карандашом (см. табл.2.3). По умолчанию цвет черный.
Mode	Режим отображения (см. табл.2.4)
Style	Стиль линии (см. табл.2.5).
Width	Толщина рисуемой линии. По умолчанию толщина линии 1 пиксель

Таблица 2.3 – Кодировка наиболее распространенных цветов

clBlack	Черный
clMaroon	Каштановый
clGreen	Зеленый
clOlive	Оливковый
clNavy	Темно-синий
clPurple	Розовый

clTeal	Зелено-голубой
clGray	Серый
clSilver	Серебристый
ClRed	Красный
clLime	Салатный
ClBlue	Синий
clFuchsia	Ярко-розовый
clAqua	Бирюзовый
clWhite	Белый

Свойство Mode определяет, как будет формироваться цвет точек линии в зависимости от цвета точек холста, через которые эта линия прочерчивается. По умолчанию вся линия вычерчивается цветом, определяемым значением свойства Pen.Color. Однако можно задать инверсный цвет линии по отношению к цвету фона. Это гарантирует, что независимо от цвета фона все участки линии будут видны, даже в том случае, если цвет линии и цвет фона совпадают.

Таблица 2.4 – Значение свойства Pen.Mode

pmBlack	Черный, не зависит от значения свойства Pen.Color
pmWhite	Белый, не зависит от значения свойства Pen.Color
pmCopy	Цвет линии определяется значением свойства Pen.Color
pmNotCopy	Цвет линии является инверсным по отношению к значению свойства Pen.Color
pmNot	Цвет точки линии определяется как инверсный по отношению к цвету точки холста, в которую выводится точка линии

Таблица 2.5 – Стиль линии

psSolid	Сплошная линия
psDash	Штриховая линия
psDot	Пунктирная линия
psDashDot	Штрихпунктирная линия
psDashDotDot	Штрихпунктирная линия с двумя пунктирами
psClear	Невидимая линия

- *Класс Brush – Кисть*

Кисть используется методами, обеспечивающими вычерчивание замкнутых областей, например геометрических фигур, для заливки (закрашивания) этих областей.

Кисть, как объект, обладает двумя свойствами (см. табл.2.6).

Таблица 2.6 – Стиль линии

Color	Цвет закрашивания замкнутой области	
Style	Стиль (тип) заполнения области	
	BsSolid	Сплошная заливка
	BsClear	Область не закрашивается
	bsHorizontal	Горизонтальная штриховка
	BsVertical	Вертикальная штриховка
	bsFDiagonal	Диагональная штриховка с наклоном линий вперед
	bsBDiagonal	Диагональная штриховка с наклоном линий назад
	BsCross	Горизонтально-вертикальная штриховка, в клетку
	bsDiagCross	Диагональная штриховка, в клетку

Область внутри контура может быть закрашена или заштрихована. В первом случае область полностью перекрывает фон, а во втором — сквозь не-заштрихованные участки области будет виден фон.

На рисунке 2.5 приведены примеры стилей заполнения областей

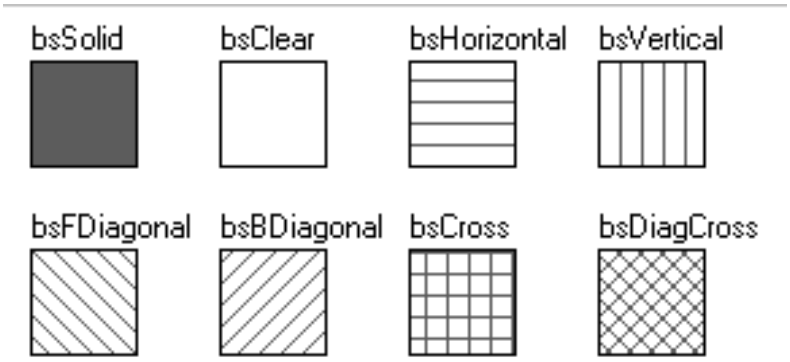


Рис. 2.5 – Стили заполнения областей

2.2 Создание простых графических изображений

В классе TCanvas определено много различных методов, предназначенных для рисования всевозможных геометрических фигур. Все геометрические фигуры можно условно разбить на контурные (т.е. не имеющие внутренней закраски) и закрашенные (имеющие внутреннюю закраску). При рисовании контурных фигур используется только карандаш Pen с установленными в нем характеристиками (цвет линии, толщина и т.д.). Если фигура является закрашенной (например эллипс, многоугольник), ее внутренность закрашивается кистью Brush с установленными в ней характеристиками (цвет, орнамент и т.д.). Тексты изображаются в соответствии с характеристиками (начертания, размер и т.д.), заданными в шрифте Font. Значения свойств Pen, Brush, Font на этапе выполнения программы можно изменять, но эти установки должны быть выполнены до того, как они будут использованы в процессе рисования. Вычерчивание графических изображений на поверхности компонента осуществляется применением соответствующих методов к свойству Canvas этого компонента.

- Прямоугольник / прямоугольник со скругленными углами

Прямоугольник вычерчивается методом Rectangle, инструкция вызова которого в общем виде выглядит следующим образом:

Объект.Canvas.Rectangle(x1, y1,x2, y2)

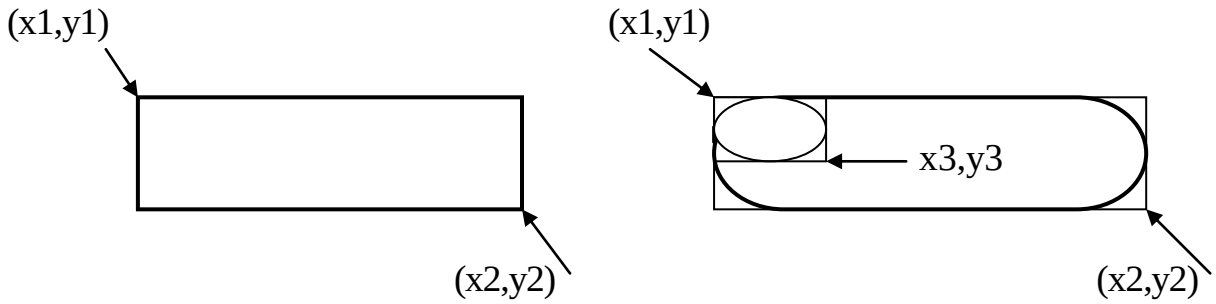
где: объект — имя объекта (компонента), на поверхности которого выполняется вычерчивание; x1, y1 и x2, y2 — координаты левого верхнего и правого нижнего углов прямоугольника.

Метод RoundRec тоже вычерчивает прямоугольник, но со скругленными углами. Инструкция вызова метода RoundRec выглядит так:

Объект.Canvas.RoundRec(x1,y1,x2, y2, x3, y3)

где: x1, y1, x2, y2 -- параметры, определяющие положение углов прямоугольника, в который вписывается прямоугольник со скругленными углами; x3 и y3 — размер эллипса, одна четверть которого используется для вычерчивания скругленного угла.

Пример4. Начертить прямоугольник и прямоугольник со скругленными углами (рис.2.6).



Canvas.Rectangle(10,10,150,50);

Cavas.RoundRect(10,10,150,50,40,40);

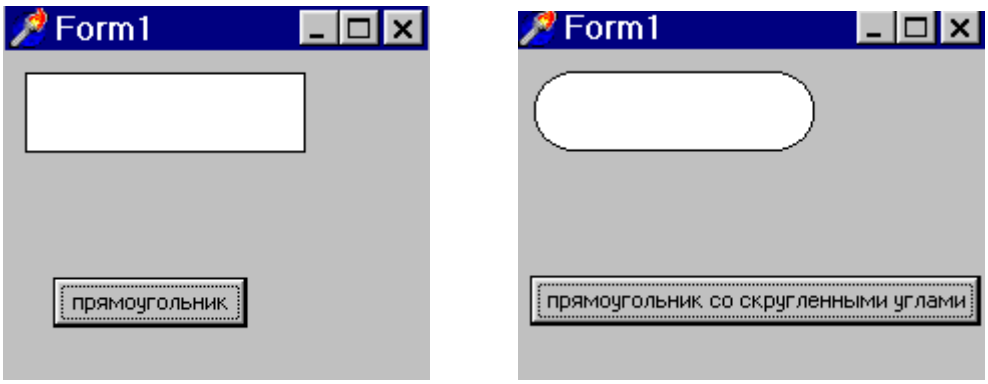


Рис.2.6 – Прямоугольник и прямоугольник со скругленными углами

Вид линии контура (цвет, ширина и стиль) определяется значениями свойства Реп, а цвет и стиль заливки области внутри прямоугольника — значениями свойства Brush поверхности (canvas), на которой прямоугольник вычерчивается.

- Окружность и эллипс

Метод Ellipse вычерчивает эллипс или окружность, в зависимости от значений параметров. Инструкция вызова метода в общем виде выглядит следующим образом:

Объект.Canvas.Ellipse(x1,y1,x2,y2)

где: объект — имя объекта (компонента), на поверхности которого выполняется вычерчивание; x1, y1, x2, y2 — координаты прямоугольника, внутри которого вычерчивается эллипс или, если прямоугольник является квадратом, окружность.

Пример5. Начертить круг и эллипс в окне программы (рис.2.7)

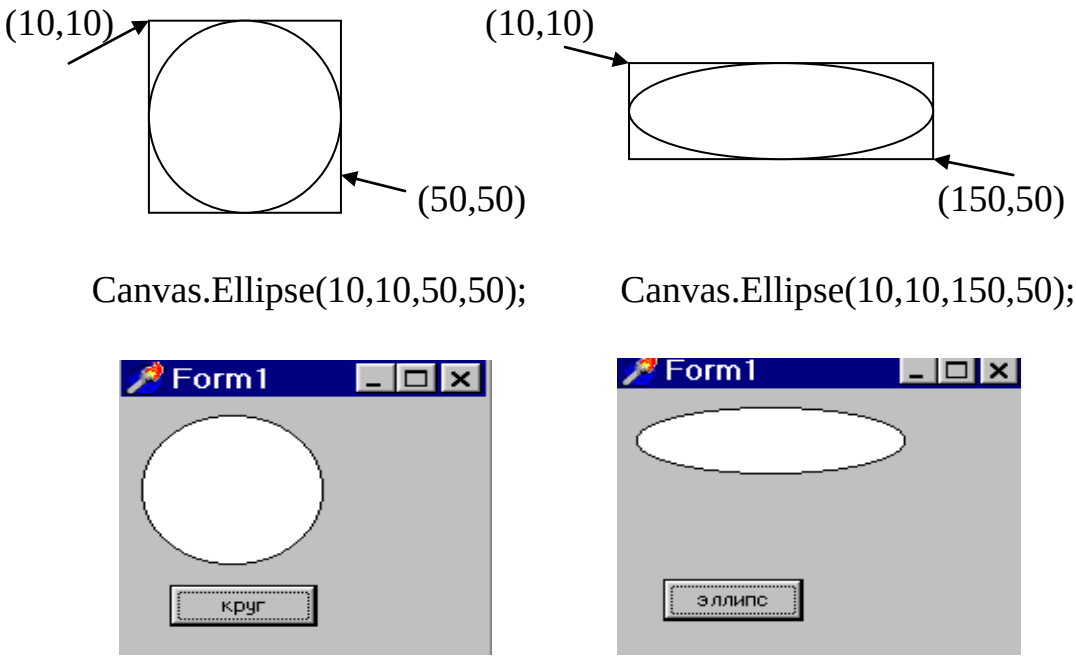


Рис.2.7 – Круг и эллипс

Цвет, толщина и стиль линии эллипса определяются значениями свойства Реп, а цвет и стиль заливки области внутри эллипса — значениями свойства Brush поверхности (Canvas), на которую выполняется вывод.

- Дуга

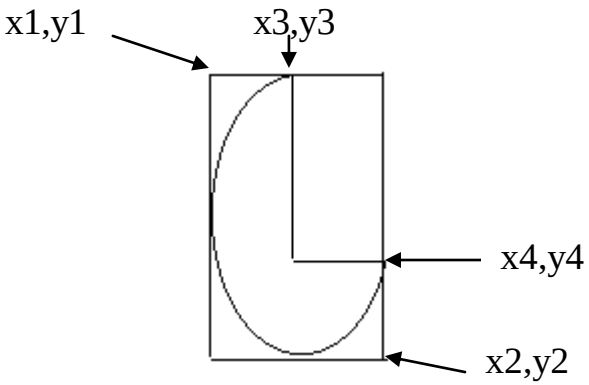
Вычерчивание дуги выполняет метод Arc, инструкция вызова которого в общем виде выглядит следующим образом:

Объект.Canvas.Arc(x1,y1,x2,y2,x3,y3,x4,y4)

где: x1, y1, x2, y2 — параметры, определяющие эллипс (окружность), частью которого является вычерчиваемая дуга; x3, y3 — параметры, определяющие начальную точку дуги; x4, y4 — параметры, определяющие конечную точку дуги.

Начальная (конечная) точка — это точка пересечения границы эллипса и прямой, проведенной из центра эллипса в точку с координатами x3 и y3 (x4, y4). Дуга вычерчивается против часовой стрелки от начальной точки к конечной. Цвет, толщина и стиль линии, которой вычерчивается дуга, определяются значениями свойства Реп поверхности (Canvas), на которую выполняется вывод.

Ïðèìåð 6. Начертить дугу в окне программы (рис.2.8)



Canvas.Arc(10,10,100,150,50,10,100,100)

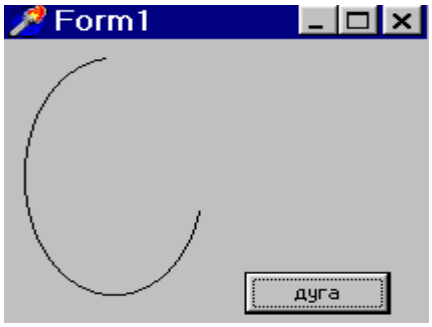


Рис. 2.8 – Дуга

- Сектор

Метод Pie вычерчивает сектор эллипса или круга. Инструкция вызова метода в общем виде выглядит следующим образом:

Объект. Canvas.Pie(x1,y1,x2,y2,x3,y3,x4,y4)

где: x1, y1, x2, y2 — параметры, определяющие эллипс (окружность), частью которого является сектор; x3, y3, x4, y4 — параметры, определяющие координаты конечных точек прямых, являющихся границами сектора.

Начальные точки прямых совпадают с центром эллипса (окружности). Сектор вырезается против часовой стрелки от прямой, заданной точкой с координатами (x3, уз), к прямой, заданной точкой с координатами (x4, y4)

Пример 7. Начертить сектор в окне программы (рис.2.9)

Canvas.Pie(10,10,100,150,50,10,100,100)

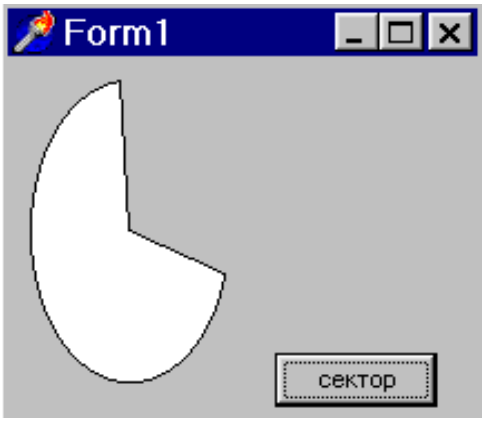


Рис. 2.9 – Сектор

- Замкнутая фигура

Метод Chord чертит сегмент эллипса в охватывающем прямоугольнике (x1,y1) ->(x2,y2). Инструкция вызова метода в общем виде выглядит следующим образом:

Объект. Canvas.Chord(x1,y1,x2,y2,x3,y3,x4,y4)

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,y4). Дуга сегмента чертится против часовой стрелки, а начальная и конечная точки дуги соединяются прямой.

Пример 8. Начертить замкнутую фигуру (рис.2.10)

Canvas.Chord(10,10,100,150,50,10,100,100)

Рис. 2.10 – Замкнутая фигура

- Многоугольник

Метод Polygon вычерчивает пером многоугольник по точкам, заданным в массиве Points. Каждый элемент массива представляет собой запись, поля (x,y) которой содержат координаты одной вершины многоугольника. Метод Polygon вычерчивает многоугольник, последовательно соединяя прямыми линиями точки, координаты которых находятся в массиве: первую со второй, вторую с третьей, третью с четвертой и т. д. Конечная точка соединяется с начальной и многоугольник заполняется кистью. Для вычерчивания без заполнения пользуются методом Polyline.

Цвет и стиль границы многоугольника определяются значениями свойства Реп, а цвет и стиль заливки области, ограниченной линией границы, — значениями свойства Brush, причем область закрашивается с использованием текущего цвета и стиля кисти.

Пример 9. Начертить треугольник (с заполнением) в окне программы (рис.2.11)

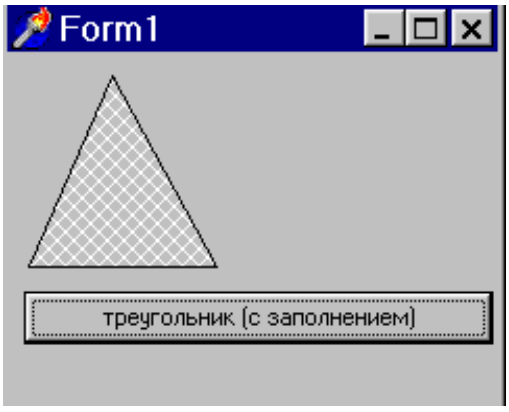


Рис.2.11 – Треугольник (с заполнением)

Программный код

```
var
ugol: array[1..3] of TPoint;
// координаты трех вершин треугольника
begin
canvas.Brush.Style:=bsDiagCross;
ugol[1].x:=50; ugol[1].y:=10;
ugol[2].x:=100;ugol[2].y:=100;
ugol[3].x:=10; ugol[3].y:=100;
Canvas.Polygon(ugol);
end;
```

- Ломаная линия

Метод Polyline вычерчивает пером ломаную линию по точкам, заданным в массиве Points. Каждый элемент массива представляет собой запись, поля x и y которой содержат координаты точки перегиба ломаной. Метод Polyline вычерчивает ломаную линию, последовательно соединяя прямыми точки, координаты которых находятся в массиве: первую со второй, вторую с третьей, третью с четвертой и т. д.

Метод Polyline можно использовать для вычерчивания замкнутых контуров. Для этого надо, чтобы первый и последний элементы массива содержали координаты одной и той же точки.

Пример 10. Начертить треугольник в окне программы (рис.2.12)

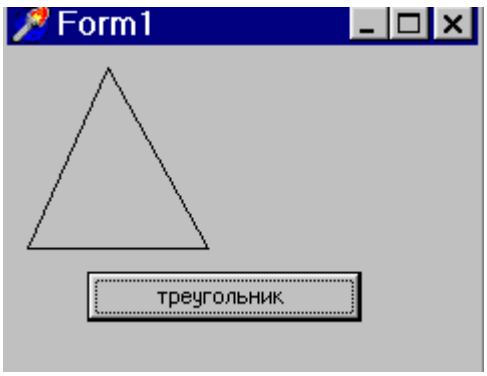


Рис.2.12 – Треугольник

Программный код

```
var

ugol: array[1..4] of TPoint;

// координаты трех вершин треугольника
begin
ugol[1].x:=50; ugol[1].y:=10;
ugol[2].x:=100;ugol[2].y:=100;
ugol[3].x:=10; ugol[3].y:=100;
ugol[4].x:=50; ugol[4].y:=10;
Canvas.Polyline(ugol);
end;
```

- Линия

Вычерчивание прямой линии осуществляет метод LineTo, инструкция вызова которого в общем виде выглядит следующим образом:

Компонент.Canvas.LineTo(x,y)

Метод LineTo вычерчивает прямую линию от текущей позиции карандаша в точку с координатами, указанными при вызове метода. Начальную точку линии можно задать, переместив карандаш в нужную точку графической поверхности. Сделать это можно при помощи метода MoveTo, указав в качестве параметров координаты нового положения карандаша.

Компонент.Canvas.Moveto(x,y)

Процедура MoveTo(X, Y: Integer) перемещает текущее положение пера в точку (X,Y) без вычерчивания линии. Вид линии (цвет, толщина и стиль) определяется значениями свойств объекта Pen графической поверхности, на которой вычерчивается линия.

Пример 11. Начертить координатную сетку в окне программы (рис.2.13)

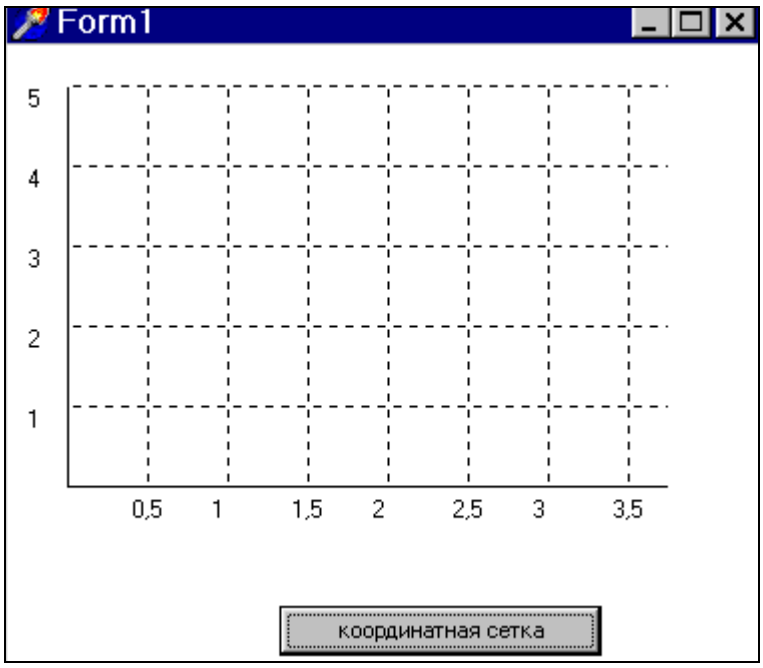


Рис.2.13 – Координатная сетка

Программный код

```
procedure TForm1.Button1Click(Sender: TObject);
var
x0,y0:integer; // координаты начала координатных осей
dx,dy:integer; // шаг координатной сетки (в пикселах)
x,y:integer;
h,w:integer; // высота и ширина области вывода координатной сетки
lx,ly:real; // метки (оцифровка) линий сетки по X и Y
dlx,dly:real; // шаг меток (оцифровки) линий сетки по X и Y
begin
x0:=30;
y0:=220;
// оси начинаются в точке (30,220)
dx:=40; dy:=40; // шар координатной сетки 40 пикселов
dlx:=0.5; // шаг меток оси X
dly:=1.0; // шаг меток оси Y
h:=200; w:=300;
```

```
with Canvas do begin
Pen.Style:=psSolid;// линия сплошная
MoveTo(x0,y0); LineTo(x0,y0-h); // ось X
MoveTo(x0,y0); LineTo(x0+w, y0); // ось Y
x:=x0+dx;
lx:=dlx;
repeat
TextOut(x-8,y0+5,FloatToStr(lx)); // значения по оси У
Pen.Style:=psDot;// линия пунктирная
MoveTo(x,y0-3);LineTo(x,y0-h); //линии сетки параллельные оси x
lx:=lx+dlx;
x:=x+dx;
until (x>x0+w);
y:=y0-dy;
ly:=dly;
repeat
TextOut(x0-20,y,FloatToStr(ly)); // значения по оси У
Pen.Style:=psDot;// линия пунктирная
MoveTo(x0+3,y); LineTo(x0+w,y); //линии сетки параллельные оси x
y:=y-dy;
ly:=ly+dly;
until (y<y0-h);
end;
end;
```

- Рисование на канве по пикселам

Поверхности, на которую программа может осуществлять вывод графики, соответствует объект Canvas. Свойство Pixels, представляющее собой двумерный массив типа TColor, содержит информацию о цвете каждой точки графической поверхности. Используя свойство Pixels, можно задать требуемый цвет для любой точки графической поверхности, т. е. "нарисовать" точку.

Например, инструкция

Form1.Canvas.Pixels[10,10]:=clRed

окрашивает точку поверхности формы в красный цвет.

Размерность массива pixels определяется размером графической поверхности. Размер графической поверхности формы (рабочей области, которую также называют клиентской) задается значениями свойств ClientWidth и ClientHeight, а размер графической поверхности компонента Image — значениями свойств Width и Height. Левой верхней точке рабочей области формы соответствует элемент Pixels [0,0], а правой нижней - Pixels[ClientWidth - 1,ClientHeight - 1].

Свойство Pixels можно использовать для построения графиков. График строится, как правило, на основе вычислений по формуле. Границы диапазона изменения аргумента функции являются исходными данными. Диапазон изменения значения функции может быть вычислен. На основании этих данных можно вычислить масштаб, позволяющий построить график таким образом, чтобы он занимал всю область формы, предназначенную для вывода графика.

Например, если некоторая функция *f(x)* может принимать значения от нуля до 1000, и для вывода ее графика используется область формы высотой в 250 пикселов, то масштаб оси Y вычисляется по формуле: т = 250/1000. Таким образом, значению f(x) = 70 будет соответствовать точка с координатой Y =233. Значение координаты *Y* вычислено по формуле

$$Y= h -f(x) * т = 250 – 70*(250/1000),$$

где *h* - высота области построения графика.

Точное значение выражения 250 - 70х(250/1000) равно 232,5. Но т. к. индексом свойства pixels, которое используется для вывода точки на поверхность Canvas, может быть только целое значение, то число 232,5 округляется к ближайшему целому, которым является число 233.

Пример 12. Построить график функции $y=2*\sin(x)*e^{x/5}$ по пикселам в окне программы (рис.2.14)

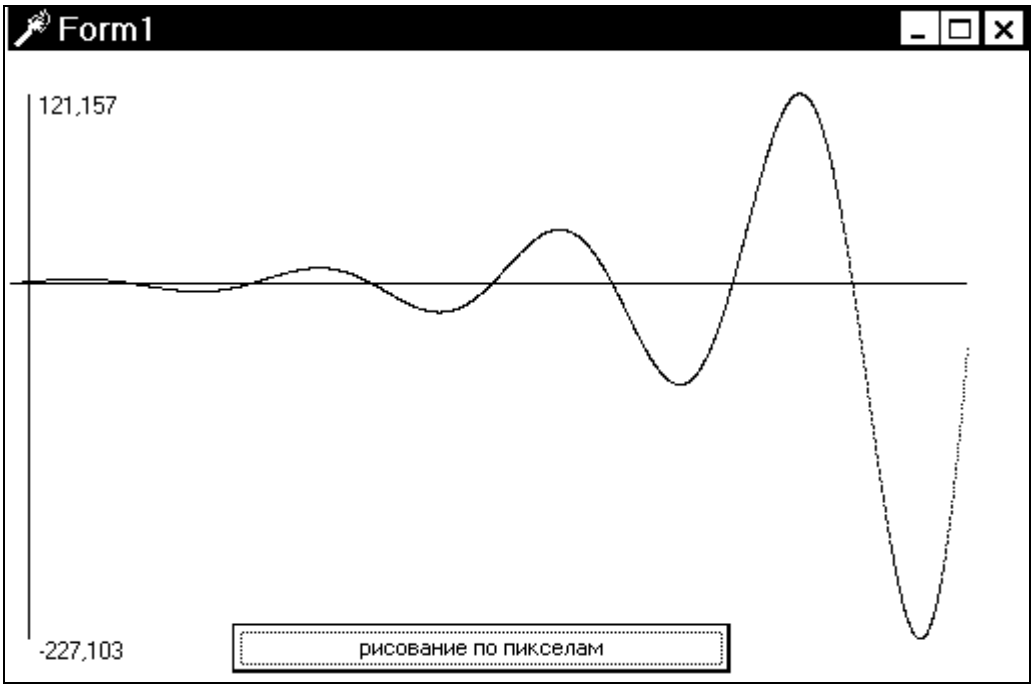


Рис.2.14 – Рисование по пикселам

Программный код

```
// Функция, график которой надо построить

Function f(x:real):real;
begin
f:=2*Sin(x)*exp(x/5) ;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
x1,x2:integer; // границы изменения аргумента функции
y1,y2:real; // границы изменения значения функции
x:real; // аргумент функции
y:real; // значение функции в точке x
dx:real; // приращение аргумента
l,b:integer; // левый нижний угол области вывода графика
w,h:integer; // ширина и высота области вывода графика
```

```
mx,my:real; // масштаб по осям X и Y
x0,y0:integer; // точка — начало координат
begin
l:=10; // X — координата левого верхнего угла
b:=ClientHeight-20;//У — координата левого верхнего угла
h:=ClientHeight-40; // высота
w:=Width-40; // ширина
x1:=0; // нижняя граница диапазона аргумента
x2:=25; // верхняя граница диапазона аргумента
dx:=0.01; // шаг аргумента
// определение максимального и минимального значения
// функции на отрезке [x1,x2]
y1:=f(x1); // минимум
y2:=f(x1); //максимум
x:=x1;
repeat
y:=f(x);
if y<y1 then y1:=y;
if y>y2 then y2:=y;
x:=x+dx;
until (x>=x2);
// определим масштаб
my:=h/abs(y2-y1); // масштаб по оси Y
mx:=w/abs(x2-x1); // масштаб по оси X
x0:=1;
y0:=b-Abs(Round(y1*my));
with Canvas do
begin
// оси
MoveTo(l,b);LineTo(l,b-h);
```

```
MoveTo(x0,y0);LineTo(x0+w,y0);
TextOut(l+5,b-h,FloatToStrF(y2,ffGeneral,6,3));
TextOut(l+5,b,FloatToStrF(y1,ffGeneral,6,3));
// построение графика
x:=x1;
repeat
y:=f(x);
Pixels[x0+Round(x*mx),y0-Round(y*my)]:=clRed;
x:=x+dx;
until (x >= x2);
end;
end;
```

2.3 Задание1

В пункте 1.3 Вы создали форму с соответствующими вкладками. Использую графические возможности Delphi оформите вкладку ЗАСТАВКА, на этой вкладке сообщите сведения об авторе. На вкладке для первой лабораторной работы постойте график функции по пикселям (см. табл. 2.7)

Таблица 2.7 – Построить график функции

Вариант	Функция	Интервал	График (цвет)	Оси (цвет)
1	x^2-sin(5x)	[-5; 5]	зеленый	синий
2	2^x-4x	[0; 10]	красный	черный
3	$(x-1)^2$-0.5e^x	[1; 6]	каштановый	оливковый
4	cos(x^2)-6	[-7; 7]	оливковый	зеленый
5	x^2-cos(πx)	[-10; -2]	розовый	красный
6	x·tg(x)	[0; 10]	синий	черный

7	x^2-cos(x)	[-20; 0]	зеленый	серый
8	sin(x)/(x^2+1)	[1; 5]	салатный	каштановый
9	40/(1+x^2)	[-20; 20]	бирюзовый	красный
10	(2x+0.5) sin(x)	[-20; 0]	красный	ярко-розовый
11	(x+1) cos ^{2}(x)	[-20; 0]	оливковый	черный
12	x-sin(x)	[-10; -4]	серебристый	синий
13	(x+x^3)$^{1/2}$	[-20; 0]	черный	каштановый
14	ctg (5+x^2)	[-5;-1]	темно-синий	салатный
15	(sin2x)/x^2	[1; 10]	синий	красный
16	(x+1) sin(x)	[-10; 10]	каштановый	зеленый
17	(0.2x^2+1)$^{1/2}$	[-5; 5]	черный	темно-синий
18	(cos x)/x	[-5; 5]	салатный	бирюзовый
19	(1+x^3)$^{1/2}$	[-10; 5]	красный	зелено-голубой
20	2-x-ln(x)	[5; 10]	ярко-розовый	синий
21	x-sin x-0.45	[-10; 5]	зеленый	серебристый
22	3x-cos x-1	[-5; 0]	розовый	каштановый
23	x ln(x)-100	[5; 10]	черный	зеленый
24	x^2-cos x^2-6	[0; 4]	оливковый	каштановый
25	ln(x)+(x+1)^3	[1; 10]	темно-синий	красный

К графику построенному в задании 2.2 построить график аппроксимирующей функции по уравнению линии тренда и сделать прогноз на ближайщие два периода. Значения переменных x и y вывести через компонент Memo.

2.4 Задание 2

Найти графически решение уравнения. На вкладке для второй лабораторной работы постойте график функции пером (см. табл. 2.8)

Таблица 2.8 – Уравнения

Вариант	Уравнение
1	x^2-5 = cos(x)
2	sin(x+6) = 5/cos ^{2}(x)
3	1.8 x^2 = sin10x
4	2 = xe ^{x}
5	x/cos(x) = sin(x)
6	x^2 = cos x^2
7	e $^{-x}$ = 2-x^2
8	ln(x) = $(x+1)^3$
9	3x = cos(x)-1
10	$(x+1)^2$ = 4/(x+1)
11	0.5e^x = -$(x-1)^2$
12	sin(x)/(x+1) = 0
13	ln(x) = 2-x
14	ctg (x^2) = sin(x)
15	x^2-6 = cos(x^2)
16	$(x+6)^2$ = 5/x
17	sin(x) = (x^2+1)
18	(x+2.8) = (x^2 +1.2)$^{1/2}$
19	ln(x+2) = 1/x

20	$\sin(3x) = 1/x^2$
21	$(x^2+1)^{1/2} = 4-x$
22	$(x+1) = \sin(x)$
23	$\cos(x) = x+2$
24	$(x+x^3) = \cos(x^2)$
25	$\sin(x) = 2x+0.5$

2.5 TChart – Компонент для построения диаграмм и графиков

TChart - наиболее важный компонент в TeeChart библиотеке. Вы можете создавать с помощью этого компонента диаграммы и графики, как во время проектирования формы так и во время выполнения.

TeeChart Библиотека включает наиболее типичные диаграммы. Каждый тип имеет определенное назначение и особенности. Имея ряд данных, Вы можете создавать различные типы диаграмм. Каждый компонент может включать несколько серий(рядов данных). Если вы хотите отобразить график, то каждая серия будет соответствовать одной кривой на графике. На рисунках 2.17 и 2.18 приведены примеры графиков, использущих в первом случае один ряд данных и два ряда данных во втором случае.

Пример 13. Построить круговую диаграмму с помощью компонента Chart.

Для предварительного проектирования графика выполним следующие действия:

- Расположим компонент Chart (страница Additional палитры компонентов) на форме;
- Для перехода в редактор диаграмм щелкните 2 раза левой кнопкой мышки по установленному на форме компоненту Chart. На вкладке Chart, на вкладке Series щелкните на кнопке Add, чтобы добавить серию (ряд данных). В появившемся диалоговом окне можно выбрать тип диаграммы или графика (см. рис.2.15, рис.2.16)

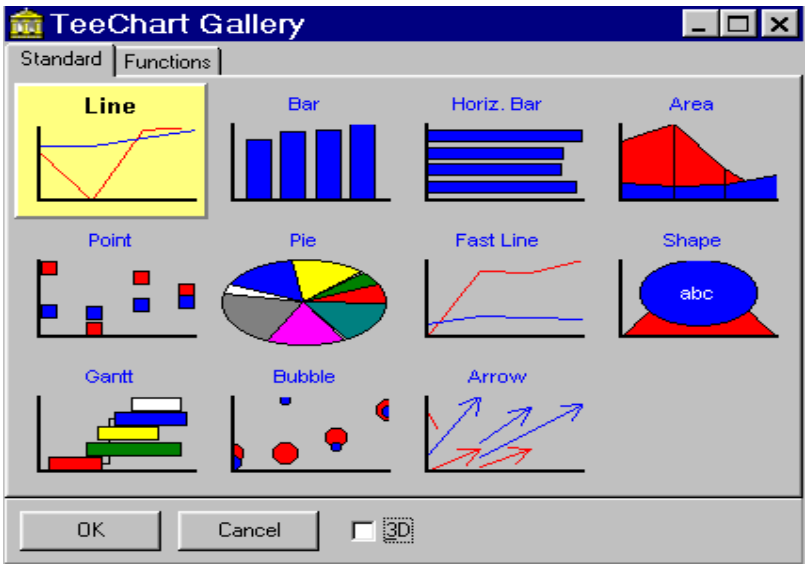


Рис.2.15 – Графики и диаграмы (плоские)

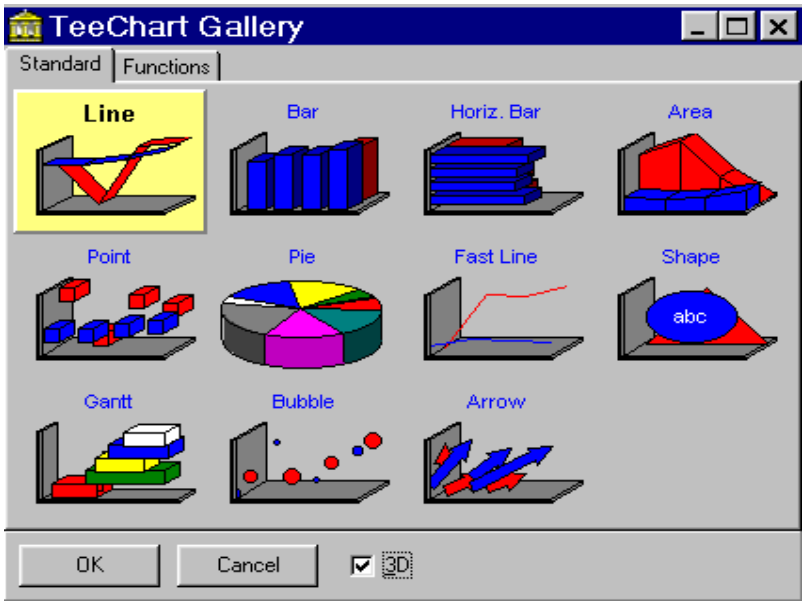


Рис.2.16 – Графики и диаграмы (трехмерные)

Выберем, например Pie – круговая диаграмма. Вкладка Titels - позволяет задать заголовок диаграммы, дизайн заголовка. Вкладка Legend – позволяет задать параметры отображения легенды. Значения вкладка Panel задают вид панели, на которой отображается диаграмма.

- В редакторе диаграмм, на вкладке Series можно выбрать дополнительные характеристики отображения серии. На вкладке Format для круговой диаграммы можно включить опцию Circled Pie, которая обеспечит при любом размере компонента Chart отображение диаграмы в виде круга. На

вкладке Marks кнопки группы Style определяют, что будет написано на ярлычках, относящихся к отдельным сегментам диаграммы: Value – значение, Percent – проценты, Label – названия данных и т.д. На вкладке General – установлен шаблон процентов, обеспечивающий отображение значений с заданной точностью (например, один знак после запятой).

- Можно задать программный код для соответствующей кноки или процедуру загрузки данных можно включить в событие OnCreate формы.

```
procedure TForm1.FormCreate(Sender: TObject);
const
nz1=15;
nz2=25;
nz3=31;
nz4=26;
begin
 with series1 do begin clear;
 add(nz1,'i - квартал', clred);
 add(nz2,'ii - квартал', clblue);
 add(nz3,'iii - квартал', clgreen);
 add(nz4,'iv - квартал', clwhite);
 end; end;
```

В результате выполненных действий получим график, представленный на рисунке 2.17

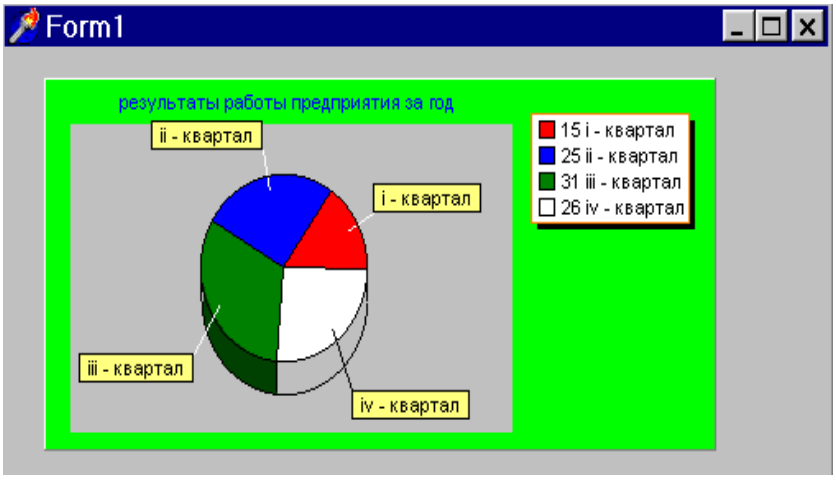


Рис.2.17 – Результат работы программы

В работе для задания отображаемых значений использованы методы серий Series:

Clear – очищает серию от занесенных ранее данных;

Add – добавляет в диаграмму новую точку:

Add(Const AValue:double, Const ALabel:String, AColor:Tcolor),

где AValue соответствует добавляемому значению,

ALabel – название, которое будет отображаться на диаграмме и в легенде,

AColor – определяет цвет.

Пример 14. Построить с помощью компонента Chart графики двух функций cos(x) и sin(x) на интервале [1,10].

- Выполним действия аналогичные примеру 13, только зададим две серии (два ряда данных->два графика) и выберем тип диараммы Line. На вкладках Axis и Walls можно задать координатные характеристики осей и трехмерных граней графика.

Программный код:

```
procedure TForm1.FormCreate(Sender: TObject);
var
i:integer;
begin
series1.Clear;
series2.Clear;
for i:=1 to 10 do
begin
series1.AddXY(i, cos(i), ' ' ,clred);
series2.AddXY(i, sin(i), ' ' ,clblue);
end;
end;
```

В результате выполненных действий будут получены графики, представленные на рисунке 2.18

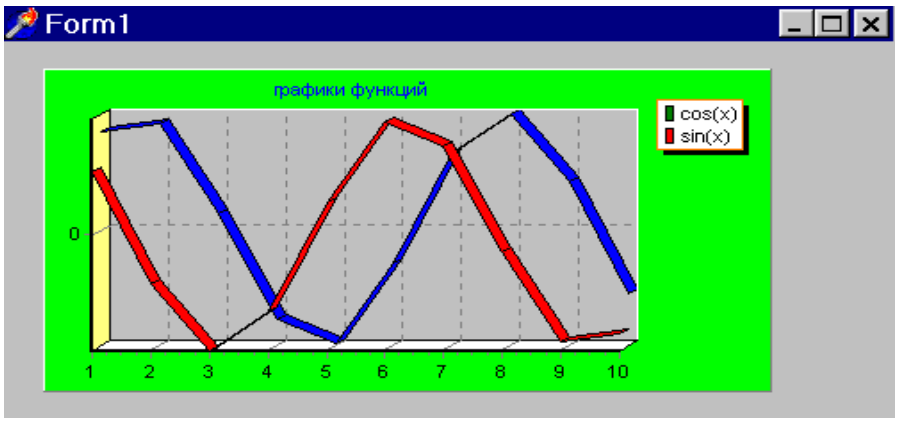


Рис.2.18 – Графики функций

В программе использован метод AddXY, который позволяет добавить новую точку в график функции.

AddXY(Const AXValue, AYValue:double, Const ALabel:String, AColor:Tcolor),

где AXValue,AYValue соответствует добавляемому значению (x и y),

ALabel – название, которое будет отображаться на диаграмме и в легенде,

AColor – определяет цвет.

Основные свойства: компонента Chart приведены в таблице 2.9

Таблица 2.9 – Основные свойства: компонента Chart

AllowPanning	Определяет возможность пользователя прокручивать наблюдаемую часть графика во время выполнения, нажимая правую кнопку мыши: pmNone – прокрутка запрещена; pmHorizontal – разрешена прокрутка только в горизонтальном направлении; pmVertical – только в вертикальном pmBoth – в обоих направлениях
AllowZoom	Позволяет пользователю изменять во время выполнения масштаб изображения, вырезая фрагменты диаграммы или графика курсором мыши.
Title	Определяет заголовок диаграммы
Foot	Определяет подпись под диаграммой. По умолчанию отсутствует. Текст подписи определяется подсвойством Text

Frame	Определяет рамку вокруг диаграммы
Legend	Легенда диаграммы – список обозначений
MarginLeft, MarginRight MarginTop MarginButtom	Значения левого, правого, верхнего и нижнего полей
ButtomAxis LeftAxis RightAxis	Эти свойства определяют характеристики соответственно ниж-ней, левой и правой осей. Задание этих свойств имеет смысл для графиков и некоторых других типов диаграмм
LeftWall ButtomWall BackWall	Эти свойства определяют характеристики соответственно левой, нижней и задней граней области трехмерного отображения графика
SeriesList	Список серий данных, отображаемых в компоненте
View3d	Разрешает или запрещает трехмерное отображение диаграммы
View3dOptions	Характеристики трехмерного отображения
Chart3DPersent	Масштаб трехмерности (толщина диаграммы, ширина лент графика)

2.6 Задание 3

На вкладке для третьей лабораторной работы постойте график функции любого производственного процесса с помощью Chart:

а) Данные заданы в виде одномерного или многомерного массива произвольных значений (входные данные представлены компонентами Memo, Stringgrid)

б) Построить график функции на интервале (см.табл 2.7), шаг по оси X выбрать произвольно

в) На форме расположить кнопку для преобразования одного типа графика в другой

СПИСОК ЛИТЕРАТУРЫ

1. Архангельский А.Я. Object Pascal в Delphi.- М.: Бином, 2002, 384 с.
2. Дарахвелидзе П.Г., Марков Е.П. Delphi – среда визуального программирования. – СПб.: BHV – Санкт-Петербург, 1996, 352 с
3. Джон Манчо, Дэвид Р. Фолкнер. Delphi: Пер. с англ. – М.: БИНОМ, 1995, 464с.
4. Фаронов В. В. Delphi 3. Учебный курс. – М.: Нолидж, 1998, 400 с.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ

ПО РАБОТЕ С ГРАФИЧЕСКИМИ СРЕДСТВАМИ DELPHI

(учебно-методическое пособие для студентов всех специальностей)

Составители: Незамова Лариса Викторовна
Гомжина Елена Васильевна

Донецкий национальный технический университет
83000, г. Донецк-00, ул. Артема, 58